FileVault

Generated by Doxygen 1.15.0

1 FileVault **1**

1.1 Features . . . 1
1.1.1 Documentation link . . . 1
1.1.2 Modern Encryption . . . 1
1.1.3 Key Derivation . . . 2
1.1.4 Compression . . . 2
1.1.5 Additional Features . . . 2
1.2 Quick Start . . . 2
1.2.1 Windows (MSVC) . . . 2
1.2.2 Windows (MinGW/MSYS2) . . . 2
1.2.3 Linux . . . 2
1.2.4 macOS . . . 2
1.3 Usage . . . 2
1.3.1 Basic Encryption . . . 2
1.3.2 Mode Presets . . . 3
1.3.3 Asymmetric Encryption . . . 3
1.3.4 Post-Quantum Cryptography (PQC) . . . 3
1.3.5 Steganography . . . 3
1.3.6 Archive . . . 3
1.3.7 Hash . . . 3
1.3.8 Benchmark . . . 3
1.3.9 List & Info . . . 3
1.4 Building . . . 4
1.4.1 Requirements . . . 4
1.4.2 Supported Platforms . . . 4
1.4.3 Manual Build . . . 4
1.5 Algorithm Comparison . . . 4
1.5.1 Symmetric (AEAD - Authenticated Encryption) . . . 4
1.5.2 Asymmetric & Post-Quantum . . . 4
1.5.3 Key Derivation . . . 5
1.5.4 Benchmark Results (1 MB data) . . . 5
1.6 Testing . . . 5
1.6.1 Test Categories . . . 5
1.7 Project Structure . . . 6
1.8 Configuration . . . 6
1.8.1 Security Levels . . . 6
1.8.2 Config File . . . 6
1.9 Documentation . . . 6
1.10 Contributing . . . 6
1.11 License . . . 6
1.12 Acknowledgments . . . 6

2 Directory Hierarchy **7**

2.1 Directories . . . 7

3 Namespace Index **17**

3.1 Namespace List . . . 17

4 Hierarchical Index **19**

4.1 Class Hierarchy . . . 19

5 Class Index **21**

5.1 Class List . . . 21

6 File Index **25**

6.1 File List . . . 25

7 Directory Documentation **29**

7.1 include/filevault/algorithms Directory Reference . . . 29

7.2 src/algorithms Directory Reference . . . 29

7.3 include/filevault/archive Directory Reference . . . 30

7.4 src/archive Directory Reference . . . 30

7.5 include/filevault/algorithms/asymmetric Directory Reference . . . 31

7.6 src/algorithms/asymmetric Directory Reference . . . 31

7.7 include/filevault/algorithms/classical Directory Reference . . . 32

7.8 src/algorithms/classical Directory Reference . . . 32

7.9 include/filevault/cli Directory Reference . . . 33

7.10 src/cli Directory Reference . . . 34

7.11 include/filevault/cli/commands Directory Reference . . . 34

7.12 src/cli/commands Directory Reference . . . 35

7.13 include/filevault/compression Directory Reference . . . 36

7.14 src/compression Directory Reference . . . 36

7.15 include/filevault/core Directory Reference . . . 37

7.16 src/core Directory Reference . . . 37

7.17 include/filevault Directory Reference . . . 38

7.18 include/filevault/format Directory Reference . . . 38

7.19 src/format Directory Reference . . . 39

7.20 include Directory Reference . . . 39

7.21 include/filevault/algorithms/pqc Directory Reference . . . 40

7.22 src/algorithms/pqc Directory Reference . . . 40

7.23 src Directory Reference . . . 41

7.24 include/filevault/steganography Directory Reference . . . 41

7.25 src/steganography Directory Reference . . . 42

7.26 include/filevault/algorithms/symmetric Directory Reference . . . 42

7.27 src/algorithms/symmetric Directory Reference . . . 43

7.28 include/filevault/utils Directory Reference . . . 44

7.29 src/utils Directory Reference . . . 45

8 Namespace Documentation **47**

8.1 filevault Namespace Reference . . . 47
8.2 filevault::algorithms Namespace Reference . . . 47
8.3 filevault::algorithms::asymmetric Namespace Reference . . . 47
8.3.1 Enumeration Type Documentation . . . 48
8.3.1.1 ECCurve . . . 48
8.3.2 Function Documentation . . . 48
8.3.2.1 get_botan_curve_name() . . . 48
8.3.2.2 get_curve_key_size() . . . 49
8.4 filevault::algorithms::classical Namespace Reference . . . 49
8.5 filevault::algorithms::pqc Namespace Reference . . . 49
8.5.1 Function Documentation . . . 50
8.5.1.1 get_dilithium_mode() . . . 50
8.5.1.2 get_kyber_mode() . . . 50
8.6 filevault::algorithms::symmetric Namespace Reference . . . 50
8.6.1 Function Documentation . . . 51
8.6.1.1 add_pkcs7_padding() . . . 51
8.6.1.2 remove_pkcs7_padding() . . . 51
8.7 filevault::archive Namespace Reference . . . 51
8.8 filevault::cli Namespace Reference . . . 52
8.9 filevault::cli::commands Namespace Reference . . . 52
8.10 filevault::compression Namespace Reference . . . 53
8.11 filevault::core Namespace Reference . . . 53
8.11.1 Typedef Documentation . . . 55
8.11.1.1 StreamProgressCallback . . . 55
8.11.2 Enumeration Type Documentation . . . 55
8.11.2.1 AlgorithmID . . . 55
8.11.2.2 AlgorithmType . . . 57
8.11.2.3 CompressionID . . . 58
8.11.2.4 CompressionType . . . 58
8.11.2.5 HashType . . . 59
8.11.2.6 KDFID . . . 59
8.11.2.7 KDFType . . . 59
8.11.2.8 PasswordStrength . . . 60
8.11.2.9 SecurityLevel . . . 60
8.11.2.10 UserMode . . . 60
8.11.3 Variable Documentation . . . 61
8.11.3.1 FILE_FORMAT_MAGIC . . . 61
8.11.3.2 FILE_FORMAT_VERSION_MAJOR . . . 61
8.11.3.3 FILE_FORMAT_VERSION_MINOR . . . 61

8.11.3.4 STREAM_MAGIC . . . 61
8.11.3.5 STREAM_VERSION . . . 61
8.12 filevault::core::presets Namespace Reference . . . 61
8.12.1 Variable Documentation . . . 61
8.12.1.1 ADVANCED . . . 61
8.12.1.2 BASIC . . . 61
8.12.1.3 STANDARD . . . 62
8.13 filevault::format Namespace Reference . . . 62
8.14 filevault::steganography Namespace Reference . . . 62
8.15 filevault::utils Namespace Reference . . . 62
8.15.1 Function Documentation . . . 63
8.15.1.1 is_terminal() . . . 63
8.15.2 Variable Documentation . . . 63
8.15.2.1 mingw_data . . . 63

9 Class Documentation **65**

9.1 filevault::algorithms::symmetric::AES_CBC Class Reference . . . 65
9.1.1 Detailed Description . . . 66
9.1.2 Constructor & Destructor Documentation . . . 66
9.1.2.1 AES_CBC() . . . 66
9.1.2.2 ~AES_CBC() . . . 66
9.1.3 Member Function Documentation . . . 67
9.1.3.1 block_size() . . . 67
9.1.3.2 decrypt() . . . 67
9.1.3.3 encrypt() . . . 68
9.1.3.4 is_suitable_for() . . . 69
9.1.3.5 iv_size() . . . 69
9.1.3.6 key_size() . . . 69
9.1.3.7 name() . . . 70
9.1.3.8 type() . . . 70
9.1.4 Member Data Documentation . . . 70
9.1.4.1 botan_name_ . . . 70
9.1.4.2 key_bits_ . . . 70
9.1.4.3 type_ . . . 70
9.2 filevault::algorithms::symmetric::AES_CFB Class Reference . . . 70
9.2.1 Detailed Description . . . 72
9.2.2 Constructor & Destructor Documentation . . . 72
9.2.2.1 AES_CFB() . . . 72
9.2.2.2 ~AES_CFB() . . . 72
9.2.3 Member Function Documentation . . . 72
9.2.3.1 block_size() . . . 72
9.2.3.2 decrypt() . . . 72

9.2.3.3 encrypt() . . . 73
9.2.3.4 is_authenticated() . . . 74
9.2.3.5 is_suitable_for() . . . 74
9.2.3.6 iv_size() . . . 74
9.2.3.7 key_size() . . . 75
9.2.3.8 name() . . . 75
9.2.3.9 requires_padding() . . . 75
9.2.3.10 type() . . . 75
9.2.4 Member Data Documentation . . . 75
9.2.4.1 botan_name_ . . . 75
9.2.4.2 key_bits_ . . . 75
9.2.4.3 type_ . . . 75
9.3 filevault::algorithms::symmetric::AES_CTR Class Reference . . . 76
9.3.1 Detailed Description . . . 77
9.3.2 Constructor & Destructor Documentation . . . 77
9.3.2.1 AES_CTR() . . . 77
9.3.2.2 ∼AES_CTR() . . . 77
9.3.3 Member Function Documentation . . . 77
9.3.3.1 decrypt() . . . 77
9.3.3.2 encrypt() . . . 78
9.3.3.3 is_suitable_for() . . . 79
9.3.3.4 key_size() . . . 79
9.3.3.5 name() . . . 79
9.3.3.6 nonce_size() . . . 80
9.3.3.7 type() . . . 80
9.3.4 Member Data Documentation . . . 80
9.3.4.1 botan_name_ . . . 80
9.3.4.2 key_bits_ . . . 80
9.3.4.3 type_ . . . 80
9.4 filevault::algorithms::symmetric::AES_ECB Class Reference . . . 80
9.4.1 Detailed Description . . . 82
9.4.2 Constructor & Destructor Documentation . . . 82
9.4.2.1 AES_ECB() . . . 82
9.4.2.2 ∼AES_ECB() . . . 82
9.4.3 Member Function Documentation . . . 82
9.4.3.1 block_size() . . . 82
9.4.3.2 decrypt() . . . 83
9.4.3.3 encrypt() . . . 84
9.4.3.4 is_authenticated() . . . 85
9.4.3.5 is_suitable_for() . . . 85
9.4.3.6 key_size() . . . 85
9.4.3.7 name() . . . 86

9.4.3.8 requires_padding() . . . 86
9.4.3.9 type() . . . 86
9.4.4 Member Data Documentation . . . 86
9.4.4.1 botan_name_ . . . 86
9.4.4.2 key_bits_ . . . 86
9.4.4.3 type_ . . . 86
9.5 filevault::algorithms::symmetric::AES_GCM Class Reference . . . 86
9.5.1 Detailed Description . . . 88
9.5.2 Constructor & Destructor Documentation . . . 88
9.5.2.1 AES_GCM() . . . 88
9.5.2.2 ∼AES_GCM() . . . 88
9.5.3 Member Function Documentation . . . 88
9.5.3.1 decrypt() . . . 88
9.5.3.2 encrypt() . . . 89
9.5.3.3 is_suitable_for() . . . 90
9.5.3.4 key_size() . . . 90
9.5.3.5 name() . . . 91
9.5.3.6 nonce_size() . . . 91
9.5.3.7 tag_size() . . . 91
9.5.3.8 type() . . . 91
9.5.4 Member Data Documentation . . . 91
9.5.4.1 botan_name_ . . . 91
9.5.4.2 key_bits_ . . . 91
9.5.4.3 type_ . . . 92
9.6 filevault::algorithms::symmetric::AES_OFB Class Reference . . . 92
9.6.1 Detailed Description . . . 93
9.6.2 Constructor & Destructor Documentation . . . 93
9.6.2.1 AES_OFB() . . . 93
9.6.2.2 ∼AES_OFB() . . . 93
9.6.3 Member Function Documentation . . . 94
9.6.3.1 block_size() . . . 94
9.6.3.2 decrypt() . . . 94
9.6.3.3 encrypt() . . . 94
9.6.3.4 is_authenticated() . . . 95
9.6.3.5 is_suitable_for() . . . 95
9.6.3.6 iv_size() . . . 95
9.6.3.7 key_size() . . . 96
9.6.3.8 name() . . . 96
9.6.3.9 requires_padding() . . . 96
9.6.3.10 type() . . . 96
9.6.4 Member Data Documentation . . . 96
9.6.4.1 botan_name_ . . . 96

9.6.4.2 key_bits_ . . . 96
9.6.4.3 type_ . . . 96
9.7 filevault::algorithms::symmetric::AES_XTS Class Reference . . . 97
9.7.1 Detailed Description . . . 98
9.7.2 Constructor & Destructor Documentation . . . 98
9.7.2.1 AES_XTS() . . . 98
9.7.2.2 ∼AES_XTS() . . . 98
9.7.3 Member Function Documentation . . . 98
9.7.3.1 block_size() . . . 98
9.7.3.2 decrypt() . . . 99
9.7.3.3 encrypt() . . . 100
9.7.3.4 is_authenticated() . . . 101
9.7.3.5 is_suitable_for() . . . 101
9.7.3.6 key_size() . . . 101
9.7.3.7 name() . . . 102
9.7.3.8 requires_padding() . . . 102
9.7.3.9 tweak_size() . . . 102
9.7.3.10 type() . . . 102
9.7.4 Member Data Documentation . . . 103
9.7.4.1 botan_name_ . . . 103
9.7.4.2 key_bits_ . . . 103
9.7.4.3 type_ . . . 103
9.8 filevault::cli::Application Class Reference . . . 103
9.8.1 Detailed Description . . . 103
9.8.2 Constructor & Destructor Documentation . . . 103
9.8.2.1 Application() . . . 103
9.8.2.2 ∼Application() . . . 103
9.8.3 Member Function Documentation . . . 104
9.8.3.1 initialize() . . . 104
9.8.3.2 register_commands() . . . 104
9.8.3.3 run() . . . 104
9.8.3.4 setup_logging() . . . 105
9.8.4 Member Data Documentation . . . 105
9.8.4.1 app_ . . . 105
9.8.4.2 commands_ . . . 105
9.8.4.3 engine_ . . . 105
9.8.4.4 log_level_ . . . 105
9.8.4.5 verbose_ . . . 106
9.9 filevault::cli::commands::ArchiveCommand Class Reference . . . 106
9.9.1 Detailed Description . . . 107
9.9.2 Constructor & Destructor Documentation . . . 107
9.9.2.1 ArchiveCommand() . . . 107

9.9.3 Member Function Documentation . . . 107
9.9.3.1 description() . . . 107
9.9.3.2 do_create() . . . 108
9.9.3.3 do_extract() . . . 108
9.9.3.4 execute() . . . 109
9.9.3.5 name() . . . 110
9.9.3.6 setup() . . . 111
9.9.4 Member Data Documentation . . . 111
9.9.4.1 algorithm_ . . . 111
9.9.4.2 compression_ . . . 111
9.9.4.3 engine_ . . . 112
9.9.4.4 extract_ . . . 112
9.9.4.5 extract_dir_ . . . 112
9.9.4.6 input_files_ . . . 112
9.9.4.7 kdf_ . . . 112
9.9.4.8 output_file_ . . . 112
9.9.4.9 password_ . . . 112
9.9.4.10 security_level_ . . . 112
9.9.4.11 verbose_ . . . 112
9.10 filevault::archive::ArchiveFormat Class Reference . . . 112
9.10.1 Detailed Description . . . 113
9.10.2 Member Function Documentation . . . 113
9.10.2.1 create_archive() . . . 113
9.10.2.2 extract_archive() . . . 113
9.10.2.3 list_files() . . . 114
9.10.2.4 read_string() . . . 115
9.10.2.5 read_uint32() . . . 115
9.10.2.6 read_uint64() . . . 115
9.10.2.7 write_uint32() . . . 115
9.10.2.8 write_uint64() . . . 116
9.10.3 Member Data Documentation . . . 116
9.10.3.1 MAGIC . . . 116
9.10.3.2 VERSION . . . 116
9.11 filevault::core::Argon2Params Struct Reference . . . 116
9.11.1 Detailed Description . . . 116
9.11.2 Member Function Documentation . . . 116
9.11.2.1 deserialize() . . . 116
9.11.2.2 serialize() . . . 117
9.11.3 Member Data Documentation . . . 117
9.11.3.1 iterations . . . 117
9.11.3.2 memory_kb . . . 117
9.11.3.3 parallelism . . . 117

9.12 filevault::algorithms::symmetric::ARIA_GCM Class Reference . . . 117
9.12.1 Detailed Description . . . 118
9.12.2 Constructor & Destructor Documentation . . . 119
9.12.2.1 ARIA_GCM() . . . 119
9.12.2.2 ~ARIA_GCM() . . . 119
9.12.3 Member Function Documentation . . . 119
9.12.3.1 decrypt() . . . 119
9.12.3.2 encrypt() . . . 120
9.12.3.3 is_suitable_for() . . . 121
9.12.3.4 key_size() . . . 121
9.12.3.5 name() . . . 122
9.12.3.6 nonce_size() . . . 122
9.12.3.7 tag_size() . . . 122
9.12.3.8 type() . . . 122
9.12.4 Member Data Documentation . . . 123
9.12.4.1 botan_name_ . . . 123
9.12.4.2 key_bits_ . . . 123
9.12.4.3 type_ . . . 123
9.13 filevault::cli::AsymmetricBenchmarkResult Struct Reference . . . 123
9.13.1 Member Data Documentation . . . 123
9.13.1.1 algorithm . . . 123
9.13.1.2 decrypt_ms . . . 123
9.13.1.3 encrypt_ms . . . 123
9.13.1.4 keygen_ms . . . 123
9.13.1.5 success . . . 123
9.14 filevault::cli::BenchmarkCommand Class Reference . . . 124
9.14.1 Constructor & Destructor Documentation . . . 125
9.14.1.1 BenchmarkCommand() . . . 125
9.14.2 Member Function Documentation . . . 125
9.14.2.1 benchmark_algorithm() . . . 125
9.14.2.2 benchmark_asymmetric() . . . 126
9.14.2.3 benchmark_asymmetric_algorithm() . . . 126
9.14.2.4 benchmark_compression() . . . 127
9.14.2.5 benchmark_hash() . . . 128
9.14.2.6 benchmark_hybrid_algorithm() . . . 128
9.14.2.7 benchmark_kdf() . . . 129
9.14.2.8 benchmark_pqc() . . . 129
9.14.2.9 benchmark_pqc_algorithm() . . . 130
9.14.2.10 benchmark_signature_algorithm() . . . 131
9.14.2.11 benchmark_symmetric() . . . 131
9.14.2.12 description() . . . 131
9.14.2.13 execute() . . . 132

9.14.2.14 get_platform_info() . . . 133
9.14.2.15 name() . . . 133
9.14.2.16 save_json_output() . . . 133
9.14.2.17 save_log_output() . . . 133
9.14.2.18 setup() . . . 133
9.14.3 Member Data Documentation . . . 134
9.14.3.1 algorithm_ . . . 134
9.14.3.2 all_ . . . 134
9.14.3.3 asymmetric_only_ . . . 134
9.14.3.4 compression_only_ . . . 134
9.14.3.5 data_size_ . . . 134
9.14.3.6 engine_ . . . 134
9.14.3.7 hash_only_ . . . 134
9.14.3.8 iterations_ . . . 134
9.14.3.9 json_output_ . . . 135
9.14.3.10 kdf_only_ . . . 135
9.14.3.11 output_file_ . . . 135
9.14.3.12 pqc_only_ . . . 135
9.14.3.13 symmetric_only_ . . . 135
9.15 filevault::cli::BenchmarkResult Struct Reference . . . 135
9.15.1 Member Data Documentation . . . 135
9.15.1.1 algorithm . . . 135
9.15.1.2 decrypt_mbps . . . 135
9.15.1.3 decrypt_ms . . . 135
9.15.1.4 encrypt_mbps . . . 135
9.15.1.5 encrypt_ms . . . 135
9.15.1.6 success . . . 136
9.16 filevault::utils::BlockProgressBar Class Reference . . . 136
9.16.1 Detailed Description . . . 136
9.16.2 Constructor & Destructor Documentation . . . 136
9.16.2.1 BlockProgressBar() . . . 136
9.16.3 Member Function Documentation . . . 136
9.16.3.1 mark_as_completed() . . . 136
9.16.3.2 set_option_text() . . . 136
9.16.3.3 set_progress() . . . 136
9.16.4 Member Data Documentation . . . 136
9.16.4.1 bar_ . . . 136
9.17 filevault::compression::Bzip2Compressor Class Reference . . . 137
9.17.1 Detailed Description . . . 137
9.17.2 Member Function Documentation . . . 138
9.17.2.1 compress() . . . 138
9.17.2.2 decompress() . . . 138

9.17.2.3 name() . . . 138
9.18 filevault::algorithms::classical::Caesar Class Reference . . . 138
9.18.1 Detailed Description . . . 140
9.18.2 Constructor & Destructor Documentation . . . 140
9.18.2.1 Caesar() [1/2] . . . 140
9.18.2.2 Caesar() [2/2] . . . 141
9.18.3 Member Function Documentation . . . 141
9.18.3.1 brute_force() [1/2] . . . 141
9.18.3.2 brute_force() [2/2] . . . 141
9.18.3.3 decrypt() [1/2] . . . 141
9.18.3.4 decrypt() [2/2] . . . 142
9.18.3.5 encrypt() [1/2] . . . 142
9.18.3.6 encrypt() [2/2] . . . 143
9.18.3.7 frequency_analysis() . . . 143
9.18.3.8 is_suitable_for() [1/2] . . . 143
9.18.3.9 is_suitable_for() [2/2] . . . 143
9.18.3.10 key_size() [1/2] . . . 143
9.18.3.11 key_size() [2/2] . . . 143
9.18.3.12 name() [1/2] . . . 143
9.18.3.13 name() [2/2] . . . 144
9.18.3.14 shift_char() [1/2] . . . 144
9.18.3.15 shift_char() [2/2] . . . 144
9.18.3.16 type() [1/2] . . . 144
9.18.3.17 type() [2/2] . . . 144
9.18.4 Member Data Documentation . . . 144
9.18.4.1 shift_ . . . 144
9.19 filevault::algorithms::symmetric::Camellia_GCM Class Reference . . . 145
9.19.1 Detailed Description . . . 146
9.19.2 Constructor & Destructor Documentation . . . 146
9.19.2.1 Camellia_GCM() . . . 146
9.19.2.2 ~Camellia_GCM() . . . 146
9.19.3 Member Function Documentation . . . 146
9.19.3.1 decrypt() . . . 146
9.19.3.2 encrypt() . . . 147
9.19.3.3 is_suitable_for() . . . 148
9.19.3.4 key_size() . . . 148
9.19.3.5 name() . . . 149
9.19.3.6 nonce_size() . . . 149
9.19.3.7 tag_size() . . . 149
9.19.3.8 type() . . . 149
9.19.4 Member Data Documentation . . . 150
9.19.4.1 botan_name_ . . . 150

9.19.4.2 key_bits_ . . . 150
9.19.4.3 type_ . . . 150
9.20 filevault::algorithms::symmetric::ChaCha20Poly1305 Class Reference . . . 150
9.20.1 Detailed Description . . . 151
9.20.2 Constructor & Destructor Documentation . . . 151
9.20.2.1 ChaCha20Poly1305() . . . 151
9.20.2.2 ~ChaCha20Poly1305() . . . 152
9.20.3 Member Function Documentation . . . 152
9.20.3.1 decrypt() . . . 152
9.20.3.2 encrypt() . . . 152
9.20.3.3 is_suitable_for() . . . 153
9.20.3.4 key_size() . . . 153
9.20.3.5 name() . . . 154
9.20.3.6 nonce_size() . . . 154
9.20.3.7 tag_size() . . . 154
9.20.3.8 type() . . . 155
9.21 filevault::core::ChunkInfo Struct Reference . . . 155
9.21.1 Detailed Description . . . 155
9.21.2 Member Data Documentation . . . 155
9.21.2.1 bytes_processed . . . 155
9.21.2.2 chunk_index . . . 155
9.21.2.3 chunk_size . . . 155
9.21.2.4 total_bytes . . . 155
9.21.2.5 total_chunks . . . 156
9.22 filevault::cli::CompressCommand Class Reference . . . 156
9.22.1 Detailed Description . . . 157
9.22.2 Constructor & Destructor Documentation . . . 157
9.22.2.1 CompressCommand() . . . 157
9.22.2.2 ~CompressCommand() . . . 157
9.22.3 Member Function Documentation . . . 157
9.22.3.1 description() . . . 157
9.22.3.2 detect_algorithm() . . . 158
9.22.3.3 do_compress() . . . 158
9.22.3.4 do_decompress() . . . 159
9.22.3.5 execute() . . . 159
9.22.3.6 generate_output_path() . . . 160
9.22.3.7 name() . . . 160
9.22.3.8 setup() . . . 161
9.22.4 Member Data Documentation . . . 161
9.22.4.1 algorithm_ . . . 161
9.22.4.2 auto_detect_ . . . 161
9.22.4.3 benchmark_ . . . 161

9.22.4.4 decompress_ . . . 161
9.22.4.5 input_file_ . . . 162
9.22.4.6 level_ . . . 162
9.22.4.7 output_file_ . . . 162
9.22.4.8 subcommand_ . . . 162
9.22.4.9 verbose_ . . . 162
9.23 filevault::compression::CompressionResult Struct Reference . . . 162
9.23.1 Detailed Description . . . 162
9.23.2 Member Data Documentation . . . 162
9.23.2.1 compressed_size . . . 162
9.23.2.2 compression_ratio . . . 162
9.23.2.3 data . . . 162
9.23.2.4 error_message . . . 162
9.23.2.5 original_size . . . 163
9.23.2.6 processing_time_ms . . . 163
9.23.2.7 success . . . 163
9.24 filevault::compression::CompressionService Class Reference . . . 163
9.24.1 Detailed Description . . . 163
9.24.2 Member Function Documentation . . . 163
9.24.2.1 create() . . . 163
9.24.2.2 get_algorithm_name() . . . 164
9.24.2.3 parse_algorithm() . . . 164
9.25 filevault::utils::Config Class Reference . . . 165
9.25.1 Detailed Description . . . 166
9.25.2 Member Function Documentation . . . 166
9.25.2.1 from_json() . . . 166
9.25.2.2 get() . . . 166
9.25.2.3 get_compression_level() . . . 166
9.25.2.4 get_config_path() . . . 167
9.25.2.5 get_default() . . . 167
9.25.2.6 get_default_algorithm() . . . 167
9.25.2.7 get_default_compression() . . . 167
9.25.2.8 get_default_kdf() . . . 167
9.25.2.9 get_default_mode() . . . 167
9.25.2.10 get_show_progress() . . . 167
9.25.2.11 get_verbose() . . . 167
9.25.2.12 load() . . . 167
9.25.2.13 reset() . . . 168
9.25.2.14 save() . . . 168
9.25.2.15 set() . . . 169
9.25.2.16 set_compression_level() . . . 169
9.25.2.17 set_default_algorithm() . . . 169

9.25.2.18 set_default_compression() . . . 169
9.25.2.19 set_default_kdf() . . . 169
9.25.2.20 set_default_mode() . . . 169
9.25.2.21 set_show_progress() . . . 169
9.25.2.22 set_verbose() . . . 169
9.25.2.23 to_json() . . . 170
9.25.3 Member Data Documentation . . . 170
9.25.3.1 compression_level_ . . . 170
9.25.3.2 default_algorithm_ . . . 170
9.25.3.3 default_compression_ . . . 170
9.25.3.4 default_kdf_ . . . 170
9.25.3.5 default_mode_ . . . 170
9.25.3.6 show_progress_ . . . 170
9.25.3.7 verbose_ . . . 170
9.26 filevault::cli::ConfigCommand Class Reference . . . 170
9.26.1 Detailed Description . . . 172
9.26.2 Constructor & Destructor Documentation . . . 172
9.26.2.1 ConfigCommand() . . . 172
9.26.3 Member Function Documentation . . . 172
9.26.3.1 description() . . . 172
9.26.3.2 execute() . . . 172
9.26.3.3 name() . . . 173
9.26.3.4 reset_config() . . . 173
9.26.3.5 set_value() . . . 174
9.26.3.6 setup() . . . 174
9.26.3.7 show_config() . . . 174
9.26.4 Member Data Documentation . . . 175
9.26.4.1 key_ . . . 175
9.26.4.2 subcommand_ . . . 175
9.26.4.3 value_ . . . 175
9.27 filevault::utils::Console Class Reference . . . 175
9.27.1 Detailed Description . . . 176
9.27.2 Member Function Documentation . . . 176
9.27.2.1 debug() . . . 176
9.27.2.2 error() . . . 176
9.27.2.3 header() . . . 177
9.27.2.4 info() . . . 178
9.27.2.5 separator() . . . 179
9.27.2.6 success() . . . 180
9.27.2.7 warning() . . . 181
9.28 filevault::core::CryptoEngine Class Reference . . . 182
9.28.1 Detailed Description . . . 182

9.28.2 Constructor & Destructor Documentation . . . 183
9.28.2.1 CryptoEngine() [1/2] . . . 183
9.28.2.2 ~CryptoEngine() . . . 183
9.28.2.3 CryptoEngine() [2/2] . . . 183
9.28.3 Member Function Documentation . . . 183
9.28.3.1 algorithm_name() . . . 183
9.28.3.2 derive_key() . . . 184
9.28.3.3 generate_nonce() . . . 184
9.28.3.4 generate_salt() . . . 185
9.28.3.5 get_algorithm() . . . 185
9.28.3.6 initialize() . . . 185
9.28.3.7 kdf_name() . . . 186
9.28.3.8 operator=() . . . 186
9.28.3.9 parse_algorithm() . . . 187
9.28.3.10 parse_kdf() . . . 187
9.28.3.11 parse_security_level() . . . 187
9.28.3.12 register_algorithm() . . . 187
9.28.3.13 security_level_name() . . . 187
9.28.4 Member Data Documentation . . . 187
9.28.4.1 algorithms_ . . . 187
9.29 filevault::core::CryptoResult Struct Reference . . . 188
9.29.1 Detailed Description . . . 188
9.29.2 Member Data Documentation . . . 188
9.29.2.1 algorithm_used . . . 188
9.29.2.2 data . . . 188
9.29.2.3 error_message . . . 188
9.29.2.4 final_size . . . 188
9.29.2.5 nonce . . . 188
9.29.2.6 original_size . . . 188
9.29.2.7 processing_time_ms . . . 188
9.29.2.8 salt . . . 188
9.29.2.9 success . . . 189
9.29.2.10 tag . . . 189
9.30 filevault::utils::CryptoUtils Class Reference . . . 189
9.30.1 Detailed Description . . . 189
9.30.2 Member Function Documentation . . . 189
9.30.2.1 format_bytes() . . . 189
9.30.2.2 hex_decode() . . . 189
9.30.2.3 hex_encode() . . . 190
9.31 filevault::cli::DecompressCommand Class Reference . . . 190
9.31.1 Detailed Description . . . 191
9.31.2 Member Function Documentation . . . 191

9.31.2.1 description() . . . 191
9.31.2.2 detect_algorithm() . . . 191
9.31.2.3 execute() . . . 192
9.31.2.4 generate_output_path() . . . 193
9.31.2.5 name() . . . 193
9.31.2.6 setup() . . . 193
9.31.3 Member Data Documentation . . . 194
9.31.3.1 algorithm_ . . . 194
9.31.3.2 auto_detect_ . . . 194
9.31.3.3 benchmark_ . . . 194
9.31.3.4 input_file_ . . . 194
9.31.3.5 output_file_ . . . 194
9.31.3.6 subcommand_ . . . 194
9.31.3.7 verbose_ . . . 194
9.32 filevault::cli::DecryptCommand Class Reference . . . 195
9.32.1 Constructor & Destructor Documentation . . . 196
9.32.1.1 DecryptCommand() . . . 196
9.32.2 Member Function Documentation . . . 196
9.32.2.1 description() . . . 196
9.32.2.2 execute() . . . 196
9.32.2.3 name() . . . 197
9.32.2.4 setup() . . . 198
9.32.3 Member Data Documentation . . . 198
9.32.3.1 engine_ . . . 198
9.32.3.2 input_file_ . . . 199
9.32.3.3 no_progress_ . . . 199
9.32.3.4 output_file_ . . . 199
9.32.3.5 password_ . . . 199
9.32.3.6 verbose_ . . . 199
9.33 filevault::algorithms::pqc::Dilithium Class Reference . . . 199
9.33.1 Detailed Description . . . 200
9.33.2 Member Enumeration Documentation . . . 200
9.33.2.1 Variant . . . 200
9.33.3 Constructor & Destructor Documentation . . . 200
9.33.3.1 Dilithium() . . . 200
9.33.4 Member Function Documentation . . . 200
9.33.4.1 generate_keypair() . . . 200
9.33.4.2 name() . . . 201
9.33.4.3 private_key_size() . . . 201
9.33.4.4 public_key_size() . . . 201
9.33.4.5 sign() . . . 201
9.33.4.6 signature_size() . . . 202

9.33.4.7 verify() . . . 202
9.33.5 Member Data Documentation . . . 203
9.33.5.1 botan_name_ . . . 203
9.33.5.2 variant_ . . . 203
9.34 filevault::cli::commands::DumpCommand Class Reference . . . 203
9.34.1 Detailed Description . . . 204
9.34.2 Constructor & Destructor Documentation . . . 204
9.34.2.1 DumpCommand() . . . 204
9.34.3 Member Function Documentation . . . 205
9.34.3.1 description() . . . 205
9.34.3.2 execute() . . . 205
9.34.3.3 name() . . . 205
9.34.3.4 setup() . . . 205
9.34.4 Member Data Documentation . . . 206
9.34.4.1 description_ . . . 206
9.34.4.2 file_path_ . . . 206
9.34.4.3 format_ . . . 206
9.34.4.4 max_bytes_ . . . 206
9.34.4.5 name_ . . . 206
9.34.4.6 show_ascii_ . . . 206
9.34.4.7 show_offset_ . . . 206
9.35 filevault::algorithms::asymmetric::ECCHybrid Class Reference . . . 206
9.35.1 Detailed Description . . . 208
9.35.2 Constructor & Destructor Documentation . . . 208
9.35.2.1 ECCHybrid() . . . 208
9.35.2.2 ~ECCHybrid() . . . 209
9.35.3 Member Function Documentation . . . 209
9.35.3.1 decrypt() . . . 209
9.35.3.2 encrypt() . . . 209
9.35.3.3 generate_key_pair() . . . 209
9.35.3.4 is_suitable_for() . . . 210
9.35.3.5 key_size() . . . 210
9.35.3.6 name() . . . 210
9.35.3.7 type() . . . 210
9.35.4 Member Data Documentation . . . 211
9.35.4.1 botan_curve_name_ . . . 211
9.35.4.2 curve_ . . . 211
9.35.4.3 ecdh_ . . . 211
9.35.4.4 type_ . . . 211
9.36 filevault::algorithms::asymmetric::ECCKeyPair Struct Reference . . . 211
9.36.1 Detailed Description . . . 211
9.36.2 Member Data Documentation . . . 211

9.36.2.1 curve . . . 211
9.36.2.2 curve_name . . . 211
9.36.2.3 private_key . . . 211
9.36.2.4 public_key . . . 211
9.37 filevault::algorithms::asymmetric::ECDH Class Reference . . . 212
9.37.1 Detailed Description . . . 212
9.37.2 Constructor & Destructor Documentation . . . 212
9.37.2.1 ECDH() . . . 212
9.37.2.2 ~ECDH() . . . 213
9.37.3 Member Function Documentation . . . 213
9.37.3.1 curve_name() . . . 213
9.37.3.2 derive_shared_secret() . . . 213
9.37.3.3 generate_key_pair() . . . 213
9.37.3.4 key_size() . . . 213
9.37.3.5 name() . . . 214
9.37.4 Member Data Documentation . . . 214
9.37.4.1 botan_curve_name_ . . . 214
9.37.4.2 curve_ . . . 214
9.38 filevault::algorithms::asymmetric::ECDHResult Struct Reference . . . 214
9.38.1 Detailed Description . . . 214
9.38.2 Member Data Documentation . . . 214
9.38.2.1 error_message . . . 214
9.38.2.2 shared_secret . . . 214
9.38.2.3 success . . . 215
9.39 filevault::algorithms::asymmetric::ECDSA Class Reference . . . 215
9.39.1 Detailed Description . . . 215
9.39.2 Constructor & Destructor Documentation . . . 215
9.39.2.1 ECDSA() . . . 215
9.39.2.2 ~ECDSA() . . . 216
9.39.3 Member Function Documentation . . . 216
9.39.3.1 curve_name() . . . 216
9.39.3.2 generate_key_pair() . . . 216
9.39.3.3 key_size() . . . 216
9.39.3.4 name() . . . 217
9.39.3.5 sign() . . . 217
9.39.3.6 signature_size() . . . 217
9.39.3.7 verify() . . . 217
9.39.4 Member Data Documentation . . . 218
9.39.4.1 botan_curve_name_ . . . 218
9.39.4.2 curve_ . . . 218
9.40 filevault::algorithms::asymmetric::ECDSASignResult Struct Reference . . . 218
9.40.1 Detailed Description . . . 218

9.40.2 Member Data Documentation . . . 218
9.40.2.1 error_message . . . 218
9.40.2.2 signature . . . 218
9.40.2.3 success . . . 218
9.41 filevault::cli::EncryptCommand Class Reference . . . 218
9.41.1 Detailed Description . . . 220
9.41.2 Constructor & Destructor Documentation . . . 220
9.41.2.1 EncryptCommand() . . . 220
9.41.3 Member Function Documentation . . . 220
9.41.3.1 description() . . . 220
9.41.3.2 execute() . . . 220
9.41.3.3 name() . . . 222
9.41.3.4 setup() . . . 222
9.41.4 Member Data Documentation . . . 223
9.41.4.1 algorithm_ . . . 223
9.41.4.2 compression_level_ . . . 223
9.41.4.3 compression_type_ . . . 223
9.41.4.4 engine_ . . . 223
9.41.4.5 force_weak_password_ . . . 223
9.41.4.6 input_file_ . . . 223
9.41.4.7 kdf_ . . . 223
9.41.4.8 mode_ . . . 223
9.41.4.9 no_progress_ . . . 223
9.41.4.10 output_file_ . . . 223
9.41.4.11 password_ . . . 223
9.41.4.12 security_level_ . . . 223
9.41.4.13 verbose_ . . . 223
9.42 filevault::core::EncryptionConfig Struct Reference . . . 224
9.42.1 Detailed Description . . . 224
9.42.2 Member Function Documentation . . . 224
9.42.2.1 apply_security_level() . . . 224
9.42.2.2 apply_user_mode() . . . 225
9.42.3 Member Data Documentation . . . 225
9.42.3.1 algorithm . . . 225
9.42.3.2 associated_data . . . 225
9.42.3.3 comment . . . 225
9.42.3.4 compression . . . 225
9.42.3.5 compression_level . . . 225
9.42.3.6 include_metadata . . . 226
9.42.3.7 kdf . . . 226
9.42.3.8 kdf_iterations . . . 226
9.42.3.9 kdf_memory_kb . . . 226

9.42.3.10 kdf_parallelism . . . 226
9.42.3.11 level . . . 226
9.42.3.12 mode . . . 226
9.42.3.13 nonce . . . 226
9.42.3.14 salt . . . 226
9.42.3.15 show_progress . . . 226
9.42.3.16 tag . . . 226
9.42.3.17 verbose . . . 226
9.43 filevault::archive::FileEntry Struct Reference . . . 226
9.43.1 Detailed Description . . . 227
9.43.2 Member Function Documentation . . . 227
9.43.2.1 deserialize() . . . 227
9.43.2.2 serialize() . . . 227
9.43.3 Member Data Documentation . . . 227
9.43.3.1 file_size . . . 227
9.43.3.2 filename . . . 227
9.43.3.3 modified_time . . . 227
9.43.3.4 offset . . . 227
9.43.3.5 permissions . . . 227
9.44 filevault::core::FileFormatHandler Class Reference . . . 228
9.44.1 Detailed Description . . . 228
9.44.2 Member Function Documentation . . . 228
9.44.2.1 create_header() . . . 228
9.44.2.2 from_algorithm_id() . . . 229
9.44.2.3 from_compression_id() . . . 229
9.44.2.4 from_kdf_id() . . . 230
9.44.2.5 is_legacy_format() . . . 230
9.44.2.6 read_file() . . . 230
9.44.2.7 read_legacy_file() . . . 231
9.44.2.8 to_algorithm_id() . . . 231
9.44.2.9 to_compression_id() . . . 231
9.44.2.10 to_kdf_id() . . . 231
9.44.2.11 write_file() . . . 232
9.45 filevault::core::FileHeader Struct Reference . . . 232
9.45.1 Detailed Description . . . 233
9.45.2 Member Function Documentation . . . 233
9.45.2.1 deserialize() . . . 233
9.45.2.2 is_valid() . . . 234
9.45.2.3 serialize() . . . 234
9.45.2.4 size() . . . 234
9.45.3 Member Data Documentation . . . 234
9.45.3.1 algorithm . . . 234

9.45.3.2 compressed . . . 235
9.45.3.3 compression . . . 235
9.45.3.4 kdf . . . 235
9.45.3.5 kdf_params . . . 235
9.45.3.6 magic . . . 235
9.45.3.7 nonce . . . 235
9.45.3.8 reserved . . . 235
9.45.3.9 salt . . . 235
9.45.3.10 version_major . . . 235
9.45.3.11 version_minor . . . 235
9.46 filevault::format::FileHeader Class Reference . . . 235
9.46.1 Detailed Description . . . 237
9.46.2 Constructor & Destructor Documentation . . . 237
9.46.2.1 FileHeader() . . . 237
9.46.3 Member Function Documentation . . . 237
9.46.3.1 algorithm() . . . 237
9.46.3.2 deserialize() . . . 238
9.46.3.3 encrypted_size() . . . 238
9.46.3.4 flags() . . . 238
9.46.3.5 is_compressed() . . . 238
9.46.3.6 kdf() . . . 238
9.46.3.7 nonce() . . . 239
9.46.3.8 original_size() . . . 239
9.46.3.9 read_uint16() . . . 239
9.46.3.10 read_uint32() . . . 239
9.46.3.11 read_uint64() . . . 240
9.46.3.12 salt() . . . 240
9.46.3.13 security_level() . . . 240
9.46.3.14 serialize() . . . 240
9.46.3.15 set_algorithm() . . . 241
9.46.3.16 set_compressed() . . . 241
9.46.3.17 set_encrypted_size() . . . 241
9.46.3.18 set_kdf() . . . 241
9.46.3.19 set_nonce() . . . 242
9.46.3.20 set_original_size() . . . 242
9.46.3.21 set_salt() . . . 242
9.46.3.22 set_security_level() . . . 242
9.46.3.23 set_tag() . . . 242
9.46.3.24 set_timestamp() . . . 243
9.46.3.25 tag() . . . 243
9.46.3.26 timestamp() . . . 243
9.46.3.27 total_size() . . . 243

9.46.3.28 validate() . . . 243
9.46.3.29 write_uint16() . . . 244
9.46.3.30 write_uint32() . . . 244
9.46.3.31 write_uint64() . . . 244
9.46.4 Member Data Documentation . . . 245
9.46.4.1 algorithm_ . . . 245
9.46.4.2 encrypted_size_ . . . 245
9.46.4.3 FLAG_COMPRESSED . . . 245
9.46.4.4 FLAG_METADATA . . . 245
9.46.4.5 flags_ . . . 245
9.46.4.6 kdf_ . . . 245
9.46.4.7 MAGIC . . . 245
9.46.4.8 MIN_HEADER_SIZE . . . 245
9.46.4.9 nonce_ . . . 245
9.46.4.10 original_size_ . . . 245
9.46.4.11 reserved_ . . . 245
9.46.4.12 salt_ . . . 245
9.46.4.13 security_level_ . . . 245
9.46.4.14 tag_ . . . 245
9.46.4.15 timestamp_ . . . 245
9.46.4.16 VERSION_MAJOR . . . 246
9.46.4.17 VERSION_MINOR . . . 246
9.47 filevault::cli::InfoCommand::FileInfo Struct Reference . . . 246
9.47.1 Detailed Description . . . 246
9.47.2 Member Data Documentation . . . 246
9.47.2.1 algorithm . . . 246
9.47.2.2 compressed . . . 246
9.47.2.3 compression . . . 246
9.47.2.4 data_size . . . 246
9.47.2.5 file_size . . . 246
9.47.2.6 has_header . . . 247
9.47.2.7 header_size . . . 247
9.47.2.8 kdf . . . 247
9.47.2.9 nonce_size . . . 247
9.47.2.10 salt_size . . . 247
9.47.2.11 tag_size . . . 247
9.47.2.12 version . . . 247
9.48 filevault::utils::FileIO Class Reference . . . 247
9.48.1 Detailed Description . . . 247
9.48.2 Member Function Documentation . . . 247
9.48.2.1 file_exists() . . . 247
9.48.2.2 file_size() . . . 248

9.48.2.3 read_file() . . . 248
9.48.2.4 write_file() . . . 249
9.49 filevault::cli::HashCommand Class Reference . . . 250
9.49.1 Detailed Description . . . 251
9.49.2 Constructor & Destructor Documentation . . . 251
9.49.2.1 HashCommand() . . . 251
9.49.3 Member Function Documentation . . . 251
9.49.3.1 calculate_file_hash() . . . 251
9.49.3.2 calculate_file_hmac() . . . 252
9.49.3.3 description() . . . 252
9.49.3.4 execute() . . . 252
9.49.3.5 get_botan_algorithm_name() . . . 253
9.49.3.6 is_secure_algorithm() . . . 253
9.49.3.7 name() . . . 254
9.49.3.8 setup() . . . 254
9.49.3.9 verify_mode() . . . 255
9.49.4 Member Data Documentation . . . 255
9.49.4.1 algorithm_ . . . 255
9.49.4.2 benchmark_ . . . 255
9.49.4.3 engine_ . . . 255
9.49.4.4 hmac_key_ . . . 255
9.49.4.5 input_file_ . . . 255
9.49.4.6 no_filename_ . . . 255
9.49.4.7 output_file_ . . . 255
9.49.4.8 output_format_ . . . 256
9.49.4.9 uppercase_ . . . 256
9.49.4.10 verbose_ . . . 256
9.49.4.11 verify_hash_ . . . 256
9.50 filevault::core::HashConfig Struct Reference . . . 256
9.50.1 Detailed Description . . . 256
9.50.2 Member Data Documentation . . . 256
9.50.2.1 algorithm . . . 256
9.50.2.2 expected_hash . . . 256
9.50.2.3 hmac_key . . . 256
9.50.2.4 hmac_mode . . . 256
9.50.2.5 include_filename . . . 256
9.50.2.6 uppercase . . . 257
9.50.2.7 verify_mode . . . 257
9.51 filevault::algorithms::classical::HillCipher Class Reference . . . 257
9.51.1 Detailed Description . . . 258
9.51.2 Member Typedef Documentation . . . 258
9.51.2.1 Matrix2x2 . . . 258

9.51.3 Member Function Documentation . . . 258
9.51.3.1 decrypt() . . . 258
9.51.3.2 decrypt_block() . . . 259
9.51.3.3 determinant() . . . 260
9.51.3.4 encrypt() . . . 260
9.51.3.5 encrypt_block() . . . 261
9.51.3.6 invert_matrix() . . . 261
9.51.3.7 is_suitable_for() . . . 262
9.51.3.8 is_valid_key() . . . 262
9.51.3.9 key_size() . . . 262
9.51.3.10 mod_inverse() . . . 262
9.51.3.11 name() . . . 263
9.51.3.12 parse_key() . . . 263
9.51.3.13 type() . . . 263
9.52 filevault::cli::ICommand Class Reference . . . 264
9.52.1 Detailed Description . . . 266
9.52.2 Constructor & Destructor Documentation . . . 266
9.52.2.1 ~ICommand() . . . 266
9.52.3 Member Function Documentation . . . 266
9.52.3.1 description() . . . 266
9.52.3.2 execute() . . . 266
9.52.3.3 name() . . . 266
9.52.3.4 setup() . . . 266
9.53 filevault::compression::ICompressor Class Reference . . . 267
9.53.1 Detailed Description . . . 267
9.53.2 Constructor & Destructor Documentation . . . 267
9.53.2.1 ~ICompressor() . . . 267
9.53.3 Member Function Documentation . . . 267
9.53.3.1 compress() . . . 267
9.53.3.2 decompress() . . . 268
9.53.3.3 name() . . . 268
9.54 filevault::core::ICryptoAlgorithm Class Reference . . . 268
9.54.1 Detailed Description . . . 270
9.54.2 Constructor & Destructor Documentation . . . 270
9.54.2.1 ~ICryptoAlgorithm() . . . 270
9.54.3 Member Function Documentation . . . 270
9.54.3.1 decrypt() . . . 270
9.54.3.2 encrypt() . . . 270
9.54.3.3 is_suitable_for() . . . 271
9.54.3.4 key_size() . . . 271
9.54.3.5 name() . . . 271
9.54.3.6 type() . . . 272

9.55 filevault::cli::InfoCommand Class Reference . . . 272
9.55.1 Detailed Description . . . 273
9.55.2 Constructor & Destructor Documentation . . . 273
9.55.2.1 InfoCommand() . . . 273
9.55.3 Member Function Documentation . . . 274
9.55.3.1 description() . . . 274
9.55.3.2 display_info() . . . 274
9.55.3.3 execute() . . . 274
9.55.3.4 name() . . . 275
9.55.3.5 parse_file() . . . 275
9.55.3.6 setup() . . . 276
9.55.4 Member Data Documentation . . . 276
9.55.4.1 engine_ . . . 276
9.55.4.2 input_file_ . . . 276
9.55.4.3 verbose_ . . . 277
9.56 filevault::cli::KeygenCommand Class Reference . . . 277
9.56.1 Detailed Description . . . 278
9.56.2 Constructor & Destructor Documentation . . . 278
9.56.2.1 KeygenCommand() . . . 278
9.56.3 Member Function Documentation . . . 278
9.56.3.1 description() . . . 278
9.56.3.2 execute() . . . 278
9.56.3.3 name() . . . 279
9.56.3.4 setup() . . . 280
9.56.4 Member Data Documentation . . . 280
9.56.4.1 algorithm_ . . . 280
9.56.4.2 engine_ . . . 280
9.56.4.3 force_ . . . 280
9.56.4.4 output_prefix_ . . . 280
9.56.4.5 verbose_ . . . 280
9.57 filevault::cli::commands::KeyInfoCommand Class Reference . . . 281
9.57.1 Detailed Description . . . 282
9.57.2 Constructor & Destructor Documentation . . . 282
9.57.2.1 KeyInfoCommand() . . . 282
9.57.3 Member Function Documentation . . . 282
9.57.3.1 description() . . . 282
9.57.3.2 execute() . . . 282
9.57.3.3 name() . . . 283
9.57.3.4 setup() . . . 283
9.57.4 Member Data Documentation . . . 283
9.57.4.1 check_pair_ . . . 283
9.57.4.2 description_ . . . 283

9.57.4.3 engine_ . . . 283
9.57.4.4 key_path_ . . . 283
9.57.4.5 name_ . . . 284
9.57.4.6 pair_key_path_ . . . 284
9.57.4.7 show_public_ . . . 284
9.58 filevault::algorithms::pqc::Kyber Class Reference . . . 284
9.58.1 Detailed Description . . . 285
9.58.2 Member Enumeration Documentation . . . 286
9.58.2.1 Variant . . . 286
9.58.3 Constructor & Destructor Documentation . . . 286
9.58.3.1 Kyber() . . . 286
9.58.3.2 ~Kyber() . . . 286
9.58.4 Member Function Documentation . . . 286
9.58.4.1 ciphertext_size() . . . 286
9.58.4.2 decrypt() . . . 286
9.58.4.3 encrypt() . . . 287
9.58.4.4 generate_keypair() . . . 288
9.58.4.5 is_suitable_for() . . . 288
9.58.4.6 key_size() . . . 288
9.58.4.7 name() . . . 288
9.58.4.8 private_key_size() . . . 289
9.58.4.9 public_key_size() . . . 289
9.58.4.10 shared_secret_size() . . . 289
9.58.4.11 type() . . . 289
9.58.5 Member Data Documentation . . . 289
9.58.5.1 botan_name_ . . . 289
9.58.5.2 type_ . . . 289
9.58.5.3 variant_ . . . 289
9.59 filevault::algorithms::pqc::KyberHybrid Class Reference . . . 289
9.59.1 Detailed Description . . . 291
9.59.2 Constructor & Destructor Documentation . . . 291
9.59.2.1 KyberHybrid() . . . 291
9.59.2.2 ~KyberHybrid() . . . 291
9.59.3 Member Function Documentation . . . 291
9.59.3.1 decrypt() . . . 291
9.59.3.2 encrypt() . . . 292
9.59.3.3 generate_keypair() . . . 293
9.59.3.4 is_suitable_for() . . . 293
9.59.3.5 key_size() . . . 293
9.59.3.6 name() . . . 293
9.59.3.7 type() . . . 294
9.59.4 Member Data Documentation . . . 294

9.59.4.1 kyber_ . . . 294
9.60 filevault::cli::ListCommand Class Reference . . . 294
9.60.1 Constructor & Destructor Documentation . . . 295
9.60.1.1 ListCommand() . . . 295
9.60.2 Member Function Documentation . . . 295
9.60.2.1 description() . . . 295
9.60.2.2 execute() . . . 295
9.60.2.3 name() . . . 296
9.60.2.4 setup() . . . 296
9.60.3 Member Data Documentation . . . 297
9.60.3.1 engine_ . . . 297
9.61 filevault::steganography::LSBSteganography Class Reference . . . 297
9.61.1 Detailed Description . . . 297
9.61.2 Member Function Documentation . . . 297
9.61.2.1 calculate_capacity() . . . 297
9.61.2.2 embed() . . . 298
9.61.2.3 embed_byte() . . . 298
9.61.2.4 extract() . . . 299
9.61.2.5 extract_byte() . . . 299
9.61.3 Member Data Documentation . . . 300
9.61.3.1 LENGTH_HEADER_SIZE . . . 300
9.62 filevault::compression::LzmaCompressor Class Reference . . . 300
9.62.1 Detailed Description . . . 301
9.62.2 Member Function Documentation . . . 301
9.62.2.1 compress() . . . 301
9.62.2.2 decompress() . . . 301
9.62.2.3 name() . . . 302
9.63 filevault::core::ModePreset Struct Reference . . . 302
9.63.1 Detailed Description . . . 302
9.63.2 Member Function Documentation . . . 302
9.63.2.1 apply_to() . . . 302
9.63.2.2 description() . . . 303
9.63.2.3 get_all_presets() . . . 303
9.63.2.4 get_preset() . . . 303
9.63.2.5 name() . . . 303
9.63.2.6 parse_mode() . . . 303
9.63.3 Member Data Documentation . . . 304
9.63.3.1 algorithm . . . 304
9.63.3.2 compression . . . 304
9.63.3.3 compression_level . . . 304
9.63.3.4 kdf . . . 304
9.63.3.5 kdf_iterations . . . 304

9.63.3.6 kdf_memory_kb . . . 304
9.63.3.7 kdf_parallelism . . . 304
9.63.3.8 mode . . . 304
9.63.3.9 security_level . . . 305
9.64 filevault::utils::Password Class Reference . . . 305
9.64.1 Detailed Description . . . 305
9.64.2 Member Function Documentation . . . 305
9.64.2.1 analyze_strength() . . . 305
9.64.2.2 calculate_entropy() . . . 306
9.64.2.3 disable_echo() . . . 307
9.64.2.4 display_strength_meter() . . . 307
9.64.2.5 enable_echo() . . . 307
9.64.2.6 estimate_crack_time() . . . 308
9.64.2.7 get_strength_color() . . . 308
9.64.2.8 get_strength_label() . . . 308
9.64.2.9 is_common_password() . . . 309
9.64.2.10 read_secure() . . . 309
9.64.3 Member Data Documentation . . . 310
9.64.3.1 common_passwords_ . . . 310
9.65 filevault::core::PasswordAnalysis Struct Reference . . . 311
9.65.1 Detailed Description . . . 311
9.65.2 Member Data Documentation . . . 311
9.65.2.1 crack_time_offline . . . 311
9.65.2.2 crack_time_online . . . 311
9.65.2.3 has_digits . . . 311
9.65.2.4 has_lowercase . . . 311
9.65.2.5 has_repeated_chars . . . 311
9.65.2.6 has_special . . . 311
9.65.2.7 has_uppercase . . . 311
9.65.2.8 is_common_password . . . 311
9.65.2.9 length . . . 312
9.65.2.10 score . . . 312
9.65.2.11 strength . . . 312
9.65.2.12 suggestions . . . 312
9.65.2.13 warnings . . . 312
9.66 filevault::core::PBKDF2Params Struct Reference . . . 312
9.66.1 Detailed Description . . . 312
9.66.2 Member Function Documentation . . . 312
9.66.2.1 deserialize() . . . 312
9.66.2.2 serialize() . . . 313
9.66.3 Member Data Documentation . . . 313
9.66.3.1 iterations . . . 313

9.67 filevault::algorithms::classical::Playfair Class Reference . . . 313
9.67.1 Detailed Description . . . 315
9.67.2 Constructor & Destructor Documentation . . . 315
9.67.2.1 Playfair() [1/2] . . . 315
9.67.2.2 Playfair() [2/2] . . . 315
9.67.3 Member Function Documentation . . . 315
9.67.3.1 build_matrix() [1/2] . . . 315
9.67.3.2 build_matrix() [2/2] . . . 316
9.67.3.3 decrypt() [1/2] . . . 316
9.67.3.4 decrypt() [2/2] . . . 317
9.67.3.5 encrypt() [1/2] . . . 317
9.67.3.6 encrypt() [2/2] . . . 318
9.67.3.7 find_position() [1/2] . . . 318
9.67.3.8 find_position() [2/2] . . . 319
9.67.3.9 is_suitable_for() [1/2] . . . 319
9.67.3.10 is_suitable_for() [2/2] . . . 319
9.67.3.11 key_size() [1/2] . . . 319
9.67.3.12 key_size() [2/2] . . . 319
9.67.3.13 name() [1/2] . . . 319
9.67.3.14 name() [2/2] . . . 320
9.67.3.15 type() [1/2] . . . 320
9.67.3.16 type() [2/2] . . . 320
9.67.4 Member Data Documentation . . . 320
9.67.4.1 matrix_ . . . 320
9.68 filevault::cli::PQCBenchmarkResult Struct Reference . . . 320
9.68.1 Member Data Documentation . . . 320
9.68.1.1 algorithm . . . 320
9.68.1.2 decaps_ms . . . 320
9.68.1.3 encaps_ms . . . 320
9.68.1.4 keygen_ms . . . 321
9.68.1.5 success . . . 321
9.69 filevault::algorithms::pqc::PQKeyPair Struct Reference . . . 321
9.69.1 Detailed Description . . . 321
9.69.2 Member Data Documentation . . . 321
9.69.2.1 algorithm . . . 321
9.69.2.2 private_key . . . 321
9.69.2.3 public_key . . . 321
9.70 filevault::utils::ProgressBar Class Reference . . . 321
9.70.1 Detailed Description . . . 322
9.70.2 Constructor & Destructor Documentation . . . 322
9.70.2.1 ProgressBar() . . . 322
9.70.2.2 ~ProgressBar() . . . 322

9.70.3 Member Function Documentation . . . 322
9.70.3.1 hide() . . . 322
9.70.3.2 mark_as_completed() . . . 322
9.70.3.3 set_postfix() . . . 322
9.70.3.4 set_progress() . . . 322
9.70.3.5 show() . . . 323
9.70.3.6 tick() . . . 323
9.70.4 Member Data Documentation . . . 323
9.70.4.1 bar_ . . . 323
9.70.4.2 current_progress_ . . . 323
9.70.4.3 max_progress_ . . . 323
9.71 filevault::core::Result< T > Struct Template Reference . . . 323
9.71.1 Detailed Description . . . 324
9.71.2 Member Function Documentation . . . 324
9.71.2.1 error() . . . 324
9.71.2.2 ok() . . . 324
9.71.2.3 operator bool() . . . 325
9.71.2.4 unwrap() . . . 325
9.71.3 Member Data Documentation . . . 325
9.71.3.1 error_message . . . 325
9.71.3.2 success . . . 325
9.71.3.3 value . . . 325
9.72 filevault::core::Result< void > Struct Reference . . . 325
9.72.1 Detailed Description . . . 326
9.72.2 Member Function Documentation . . . 326
9.72.2.1 error() . . . 326
9.72.2.2 ok() . . . 326
9.72.2.3 operator bool() . . . 326
9.72.2.4 unwrap() . . . 326
9.72.3 Member Data Documentation . . . 326
9.72.3.1 error_message . . . 326
9.72.3.2 success . . . 326
9.72.3.3 value . . . 326
9.73 filevault::algorithms::asymmetric::RSA Class Reference . . . 326
9.73.1 Detailed Description . . . 328
9.73.2 Constructor & Destructor Documentation . . . 328
9.73.2.1 RSA() . . . 328
9.73.2.2 ~RSA() . . . 328
9.73.3 Member Function Documentation . . . 329
9.73.3.1 decrypt() . . . 329
9.73.3.2 encrypt() . . . 329
9.73.3.3 generate_key_pair() . . . 330

9.73.3.4 is_authenticated() . . . 330
9.73.3.5 is_suitable_for() . . . 330
9.73.3.6 key_size() . . . 330
9.73.3.7 max_plaintext_size() . . . 330
9.73.3.8 name() . . . 331
9.73.3.9 requires_padding() . . . 331
9.73.3.10 sign() . . . 331
9.73.3.11 type() . . . 331
9.73.3.12 verify() . . . 331
9.73.4 Member Data Documentation . . . 332
9.73.4.1 key_bits_ . . . 332
9.73.4.2 type_ . . . 332
9.74 filevault::algorithms::asymmetric::RSAKeyPair Struct Reference . . . 332
9.74.1 Detailed Description . . . 332
9.74.2 Member Data Documentation . . . 332
9.74.2.1 bits . . . 332
9.74.2.2 private_key . . . 332
9.74.2.3 public_key . . . 333
9.75 filevault::core::ScryptParams Struct Reference . . . 333
9.75.1 Detailed Description . . . 333
9.75.2 Member Function Documentation . . . 333
9.75.2.1 deserialize() . . . 333
9.75.2.2 serialize() . . . 333
9.75.3 Member Data Documentation . . . 333
9.75.3.1 n . . . 333
9.75.3.2 p . . . 333
9.75.3.3 r . . . 334
9.76 filevault::algorithms::symmetric::Serpent_GCM Class Reference . . . 334
9.76.1 Detailed Description . . . 335
9.76.2 Constructor & Destructor Documentation . . . 335
9.76.2.1 Serpent_GCM() . . . 335
9.76.3 Member Function Documentation . . . 336
9.76.3.1 decrypt() . . . 336
9.76.3.2 encrypt() . . . 336
9.76.3.3 is_suitable_for() . . . 337
9.76.3.4 key_size() . . . 337
9.76.3.5 name() . . . 338
9.76.3.6 process_gcm() . . . 338
9.76.3.7 type() . . . 339
9.77 filevault::cli::SignatureBenchmarkResult Struct Reference . . . 339
9.77.1 Member Data Documentation . . . 339
9.77.1.1 algorithm . . . 339

9.77.1.2 keygen_ms . . . 340
9.77.1.3 sign_ms . . . 340
9.77.1.4 success . . . 340
9.77.1.5 verify_ms . . . 340
9.78 filevault::cli::commands::SignCommand Class Reference . . . 340
9.78.1 Detailed Description . . . 341
9.78.2 Constructor & Destructor Documentation . . . 341
9.78.2.1 SignCommand() . . . 341
9.78.3 Member Function Documentation . . . 341
9.78.3.1 description() . . . 341
9.78.3.2 execute() . . . 341
9.78.3.3 name() . . . 342
9.78.3.4 setup() . . . 342
9.78.4 Member Data Documentation . . . 343
9.78.4.1 algorithm_ . . . 343
9.78.4.2 description_ . . . 343
9.78.4.3 engine_ . . . 343
9.78.4.4 file_path_ . . . 343
9.78.4.5 name_ . . . 343
9.78.4.6 output_path_ . . . 343
9.78.4.7 private_key_path_ . . . 343
9.79 filevault::algorithms::symmetric::SM4_GCM Class Reference . . . 343
9.79.1 Detailed Description . . . 345
9.79.2 Constructor & Destructor Documentation . . . 345
9.79.2.1 SM4_GCM() . . . 345
9.79.2.2 ∼SM4_GCM() . . . 345
9.79.3 Member Function Documentation . . . 345
9.79.3.1 decrypt() . . . 345
9.79.3.2 encrypt() . . . 346
9.79.3.3 is_suitable_for() . . . 347
9.79.3.4 key_size() . . . 347
9.79.3.5 name() . . . 348
9.79.3.6 nonce_size() . . . 348
9.79.3.7 tag_size() . . . 348
9.79.3.8 type() . . . 348
9.80 filevault::cli::commands::StegoCommand Class Reference . . . 349
9.80.1 Detailed Description . . . 350
9.80.2 Member Function Documentation . . . 350
9.80.2.1 description() . . . 350
9.80.2.2 do_capacity() . . . 350
9.80.2.3 do_embed() . . . 351
9.80.2.4 do_extract() . . . 352

9.80.2.5 execute() . . . 352
9.80.2.6 name() . . . 353
9.80.2.7 setup() . . . 353
9.80.3 Member Data Documentation . . . 354
9.80.3.1 bits_per_channel_ . . . 354
9.80.3.2 cover_image_ . . . 354
9.80.3.3 input_file_ . . . 354
9.80.3.4 operation_ . . . 354
9.80.3.5 output_file_ . . . 354
9.80.3.6 verbose_ . . . 354
9.81 filevault::core::StreamingConfig Struct Reference . . . 354
9.81.1 Detailed Description . . . 355
9.81.2 Member Data Documentation . . . 355
9.81.2.1 algorithm . . . 355
9.81.2.2 chunk_size . . . 355
9.81.2.3 compression . . . 355
9.81.2.4 compression_level . . . 355
9.81.2.5 kdf . . . 355
9.81.2.6 level . . . 355
9.81.2.7 progress_callback . . . 355
9.82 filevault::core::StreamingCrypto Class Reference . . . 355
9.82.1 Detailed Description . . . 356
9.82.2 Member Function Documentation . . . 356
9.82.2.1 decrypt_file() . . . 356
9.82.2.2 derive_chunk_nonce() . . . 357
9.82.2.3 encrypt_file() . . . 357
9.82.2.4 get_recommended_chunk_size() . . . 358
9.82.2.5 read_stream_header() . . . 358
9.82.2.6 should_use_streaming() . . . 359
9.82.2.7 write_stream_header() . . . 359
9.83 filevault::core::StreamingResult Struct Reference . . . 360
9.83.1 Detailed Description . . . 360
9.83.2 Member Data Documentation . . . 360
9.83.2.1 bytes_processed . . . 360
9.83.2.2 chunks_processed . . . 360
9.83.2.3 error_message . . . 360
9.83.2.4 processing_time_ms . . . 360
9.83.2.5 success . . . 360
9.83.2.6 throughput_mbps . . . 360
9.84 filevault::algorithms::classical::SubstitutionCipher Class Reference . . . 360
9.84.1 Detailed Description . . . 362
9.84.2 Member Typedef Documentation . . . 362

9.84.2.1 SubstitutionMap . . . 362
9.84.3 Member Function Documentation . . . 362
9.84.3.1 apply_substitution() . . . 362
9.84.3.2 create_reverse_map() . . . 362
9.84.3.3 decrypt() . . . 363
9.84.3.4 encrypt() . . . 364
9.84.3.5 is_suitable_for() . . . 364
9.84.3.6 is_valid_key() . . . 364
9.84.3.7 key_size() . . . 365
9.84.3.8 name() . . . 365
9.84.3.9 parse_key() . . . 365
9.84.3.10 type() . . . 365
9.85 filevault::utils::TableBuilder Class Reference . . . 366
9.85.1 Detailed Description . . . 366
9.85.2 Member Function Documentation . . . 366
9.85.2.1 print_algorithm_list() . . . 366
9.85.2.2 print_benchmark_results() . . . 367
9.85.2.3 print_encryption_config() . . . 367
9.85.2.4 print_file_summary() . . . 367
9.86 filevault::utils::TableFormatter Class Reference . . . 367
9.86.1 Detailed Description . . . 368
9.86.2 Constructor & Destructor Documentation . . . 368
9.86.2.1 TableFormatter() . . . 368
9.86.3 Member Function Documentation . . . 368
9.86.3.1 add_row() . . . 368
9.86.3.2 algorithm_list_table() . . . 368
9.86.3.3 benchmark_table() . . . 369
9.86.3.4 file_info_table() . . . 369
9.86.3.5 print() . . . 370
9.86.3.6 set_border_style() . . . 370
9.86.3.7 set_column_alignment() . . . 370
9.86.3.8 set_column_format() . . . 370
9.86.3.9 to_string() . . . 370
9.86.4 Member Data Documentation . . . 370
9.86.4.1 table_ . . . 370
9.87 filevault::algorithms::symmetric::TripleDES Class Reference . . . 371
9.87.1 Detailed Description . . . 372
9.87.2 Constructor & Destructor Documentation . . . 372
9.87.2.1 TripleDES() . . . 372
9.87.2.2 ~TripleDES() . . . 372
9.87.3 Member Function Documentation . . . 372
9.87.3.1 block_size() . . . 372

9.87.3.2 decrypt() . . . 372
9.87.3.3 encrypt() . . . 373
9.87.3.4 is_suitable_for() . . . 374
9.87.3.5 iv_size() . . . 374
9.87.3.6 key_size() . . . 374
9.87.3.7 name() . . . 375
9.87.3.8 type() . . . 375
9.88 filevault::algorithms::symmetric::Twofish_GCM Class Reference . . . 375
9.88.1 Detailed Description . . . 377
9.88.2 Constructor & Destructor Documentation . . . 377
9.88.2.1 Twofish_GCM() . . . 377
9.88.3 Member Function Documentation . . . 378
9.88.3.1 decrypt() . . . 378
9.88.3.2 encrypt() . . . 378
9.88.3.3 is_suitable_for() . . . 379
9.88.3.4 key_size() . . . 380
9.88.3.5 name() . . . 380
9.88.3.6 nonce_size() . . . 380
9.88.3.7 process_gcm() . . . 380
9.88.3.8 tag_size() . . . 381
9.88.3.9 type() . . . 381
9.88.4 Member Data Documentation . . . 382
9.88.4.1 botan_name_ . . . 382
9.88.4.2 key_bits_ . . . 382
9.88.4.3 type_ . . . 382
9.89 filevault::cli::commands::VerifyCommand Class Reference . . . 382
9.89.1 Detailed Description . . . 383
9.89.2 Constructor & Destructor Documentation . . . 383
9.89.2.1 VerifyCommand() . . . 383
9.89.3 Member Function Documentation . . . 384
9.89.3.1 description() . . . 384
9.89.3.2 execute() . . . 384
9.89.3.3 name() . . . 384
9.89.3.4 setup() . . . 384
9.89.4 Member Data Documentation . . . 385
9.89.4.1 algorithm_ . . . 385
9.89.4.2 description_ . . . 385
9.89.4.3 engine_ . . . 385
9.89.4.4 file_path_ . . . 385
9.89.4.5 name_ . . . 385
9.89.4.6 public_key_path_ . . . 385
9.89.4.7 signature_path_ . . . 385

9.90 filevault::algorithms::classical::Vigenere Class Reference . . . 385
9.90.1 Detailed Description . . . 387
9.90.2 Constructor & Destructor Documentation . . . 387
9.90.2.1 Vigenere() [1/2] . . . 387
9.90.2.2 Vigenere() [2/2] . . . 388
9.90.3 Member Function Documentation . . . 388
9.90.3.1 decrypt() [1/2] . . . 388
9.90.3.2 decrypt() [2/2] . . . 389
9.90.3.3 encrypt() [1/2] . . . 389
9.90.3.4 encrypt() [2/2] . . . 389
9.90.3.5 estimate_key_length() . . . 390
9.90.3.6 is_suitable_for() [1/2] . . . 390
9.90.3.7 is_suitable_for() [2/2] . . . 390
9.90.3.8 kasiski_examination() . . . 390
9.90.3.9 key_size() [1/2] . . . 390
9.90.3.10 key_size() [2/2] . . . 390
9.90.3.11 name() [1/2] . . . 390
9.90.3.12 name() [2/2] . . . 390
9.90.3.13 type() [1/2] . . . 391
9.90.3.14 type() [2/2] . . . 391
9.90.4 Member Data Documentation . . . 391
9.90.4.1 keyword_ . . . 391
9.91 filevault::compression::ZlibCompressor Class Reference . . . 391
9.91.1 Detailed Description . . . 392
9.91.2 Member Function Documentation . . . 392
9.91.2.1 compress() . . . 392
9.91.2.2 decompress() . . . 393
9.91.2.3 name() . . . 393

10 File Documentation **395**

10.1 include/filevault/algorithms/asymmetric/ecc.hpp File Reference . . . 395
10.1.1 Detailed Description . . . 396
10.2 ecc.hpp . . . 396
10.3 include/filevault/algorithms/asymmetric/rsa.hpp File Reference . . . 398
10.3.1 Detailed Description . . . 399
10.4 rsa.hpp . . . 399
10.5 include/filevault/algorithms/classical/caesar.hpp File Reference . . . 400
10.6 caesar.hpp . . . 400
10.7 include/filevault/algorithms/classical/classical_ciphers.hpp File Reference . . . 401
10.8 classical_ciphers.hpp . . . 402
10.9 include/filevault/algorithms/classical/hill.hpp File Reference . . . 403
10.10 hill.hpp . . . 404

10.11 include/filevault/algorithms/classical/playfair.hpp File Reference . . . 405
10.12 playfair.hpp . . . 406
10.13 include/filevault/algorithms/classical/substitution.hpp File Reference . . . 407
10.14 substitution.hpp . . . 408
10.15 include/filevault/algorithms/classical/vigenere.hpp File Reference . . . 409
10.16 vigenere.hpp . . . 411
10.17 include/filevault/algorithms/pqc/post_quantum.hpp File Reference . . . 411
10.18 post_quantum.hpp . . . 413
10.19 include/filevault/algorithms/symmetric/aes_cbc.hpp File Reference . . . 414
10.19.1 Detailed Description . . . 416
10.20 aes_cbc.hpp . . . 416
10.21 include/filevault/algorithms/symmetric/aes_cfb.hpp File Reference . . . 416
10.21.1 Detailed Description . . . 418
10.22 aes_cfb.hpp . . . 418
10.23 include/filevault/algorithms/symmetric/aes_ctr.hpp File Reference . . . 418
10.23.1 Detailed Description . . . 420
10.24 aes_ctr.hpp . . . 420
10.25 include/filevault/algorithms/symmetric/aes_ecb.hpp File Reference . . . 420
10.25.1 Detailed Description . . . 422
10.26 aes_ecb.hpp . . . 422
10.27 include/filevault/algorithms/symmetric/aes_gcm.hpp File Reference . . . 422
10.28 aes_gcm.hpp . . . 423
10.29 include/filevault/algorithms/symmetric/aes_ofb.hpp File Reference . . . 424
10.29.1 Detailed Description . . . 426
10.30 aes_ofb.hpp . . . 426
10.31 include/filevault/algorithms/symmetric/aes_xts.hpp File Reference . . . 426
10.31.1 Detailed Description . . . 428
10.32 aes_xts.hpp . . . 428
10.33 include/filevault/algorithms/symmetric/aria_gcm.hpp File Reference . . . 428
10.33.1 Detailed Description . . . 429
10.34 aria_gcm.hpp . . . 430
10.35 include/filevault/algorithms/symmetric/camellia_gcm.hpp File Reference . . . 430
10.35.1 Detailed Description . . . 431
10.36 camellia_gcm.hpp . . . 432
10.37 include/filevault/algorithms/symmetric/chacha20_poly1305.hpp File Reference . . . 432
10.38 chacha20_poly1305.hpp . . . 434
10.39 include/filevault/algorithms/symmetric/serpent_gcm.hpp File Reference . . . 434
10.40 serpent_gcm.hpp . . . 435
10.41 include/filevault/algorithms/symmetric/sm4_gcm.hpp File Reference . . . 436
10.41.1 Detailed Description . . . 437
10.42 sm4_gcm.hpp . . . 438
10.43 include/filevault/algorithms/symmetric/triple_des.hpp File Reference . . . 438

10.43.1 Detailed Description . . . 440
10.44 triple_des.hpp . . . 440
10.45 include/filevault/algorithms/symmetric/twofish_gcm.hpp File Reference . . . 440
10.45.1 Detailed Description . . . 441
10.46 twofish_gcm.hpp . . . 442
10.47 include/filevault/archive/archive_format.hpp File Reference . . . 442
10.48 archive_format.hpp . . . 443
10.49 include/filevault/cli/app.hpp File Reference . . . 444
10.50 app.hpp . . . 445
10.51 include/filevault/cli/command.hpp File Reference . . . 446
10.52 command.hpp . . . 447
10.53 include/filevault/cli/commands/archive_cmd.hpp File Reference . . . 447
10.54 archive_cmd.hpp . . . 448
10.55 include/filevault/cli/commands/benchmark_cmd.hpp File Reference . . . 449
10.56 benchmark_cmd.hpp . . . 450
10.57 include/filevault/cli/commands/compress_cmd.hpp File Reference . . . 451
10.58 compress_cmd.hpp . . . 452
10.59 include/filevault/cli/commands/config_cmd.hpp File Reference . . . 453
10.60 config_cmd.hpp . . . 453
10.61 include/filevault/cli/commands/decompress_cmd.hpp File Reference . . . 454
10.62 decompress_cmd.hpp . . . 455
10.63 include/filevault/cli/commands/decrypt_cmd.hpp File Reference . . . 456
10.64 decrypt_cmd.hpp . . . 456
10.65 include/filevault/cli/commands/dump_cmd.hpp File Reference . . . 457
10.66 dump_cmd.hpp . . . 458
10.67 include/filevault/cli/commands/encrypt_cmd.hpp File Reference . . . 459
10.68 encrypt_cmd.hpp . . . 459
10.69 include/filevault/cli/commands/hash_cmd.hpp File Reference . . . 460
10.70 hash_cmd.hpp . . . 461
10.71 include/filevault/cli/commands/info_cmd.hpp File Reference . . . 462
10.72 info_cmd.hpp . . . 463
10.73 include/filevault/cli/commands/keygen_cmd.hpp File Reference . . . 464
10.73.1 Detailed Description . . . 465
10.74 keygen_cmd.hpp . . . 465
10.75 include/filevault/cli/commands/keyinfo_cmd.hpp File Reference . . . 466
10.76 keyinfo_cmd.hpp . . . 467
10.77 include/filevault/cli/commands/list_cmd.hpp File Reference . . . 467
10.78 list_cmd.hpp . . . 468
10.79 include/filevault/cli/commands/sign_cmd.hpp File Reference . . . 468
10.80 sign_cmd.hpp . . . 469
10.81 include/filevault/cli/commands/stego_cmd.hpp File Reference . . . 470
10.82 stego_cmd.hpp . . . 471

10.83 include/filevault/cli/commands/verify_cmd.hpp File Reference . . . 472
10.84 verify_cmd.hpp . . . 472
10.85 include/filevault/compression/compressor.hpp File Reference . . . 473
10.86 compressor.hpp . . . 474
10.87 include/filevault/core/crypto_algorithm.hpp File Reference . . . 475
10.88 crypto_algorithm.hpp . . . 476
10.89 include/filevault/core/crypto_engine.hpp File Reference . . . 477
10.90 crypto_engine.hpp . . . 478
10.91 include/filevault/core/file_format.hpp File Reference . . . 478
10.92 file_format.hpp . . . 480
10.93 include/filevault/core/modes.hpp File Reference . . . 482
10.94 modes.hpp . . . 483
10.95 include/filevault/core/result.hpp File Reference . . . 484
10.96 result.hpp . . . 485
10.97 include/filevault/core/streaming.hpp File Reference . . . 486
10.98 streaming.hpp . . . 487
10.99 include/filevault/core/types.hpp File Reference . . . 488
10.100 types.hpp . . . 490
10.101 include/filevault/format/file_header.hpp File Reference . . . 493
10.102 file_header.hpp . . . 494
10.103 include/filevault/steganography/lsb.hpp File Reference . . . 495
10.104 lsb.hpp . . . 496
10.105 include/filevault/utils/config.hpp File Reference . . . 497
10.106 config.hpp . . . 498
10.107 include/filevault/utils/console.hpp File Reference . . . 499
10.108 console.hpp . . . 499
10.109 include/filevault/utils/crypto_utils.hpp File Reference . . . 500
10.110 crypto_utils.hpp . . . 500
10.111 include/filevault/utils/file_io.hpp File Reference . . . 501
10.112 file_io.hpp . . . 501
10.113 include/filevault/utils/password.hpp File Reference . . . 502
10.114 password.hpp . . . 503
10.115 include/filevault/utils/progress.hpp File Reference . . . 503
10.116 progress.hpp . . . 504
10.117 include/filevault/utils/table_formatter.hpp File Reference . . . 505
10.118 table_formatter.hpp . . . 506
10.119 README.md File Reference . . . 506
10.120 src/algorithms/asymmetric/ecc.cpp File Reference . . . 506
10.120.1 Detailed Description . . . 507
10.121 src/algorithms/asymmetric/rsa.cpp File Reference . . . 507
10.121.1 Detailed Description . . . 508
10.122 src/algorithms/classical/caesar.cpp File Reference . . . 508

10.123 src/algorithms/classical/classical_ciphers.cpp File Reference . . . 509
10.124 src/algorithms/classical/hill.cpp File Reference . . . 509
10.125 src/algorithms/classical/playfair.cpp File Reference . . . 510
10.126 src/algorithms/classical/substitution.cpp File Reference . . . 511
10.127 src/algorithms/classical/vigenere.cpp File Reference . . . 512
10.128 src/algorithms/pqc/post_quantum.cpp File Reference . . . 513
10.128.1 Detailed Description . . . 514
10.129 src/algorithms/symmetric/aes_cbc.cpp File Reference . . . 514
10.129.1 Detailed Description . . . 514
10.130 src/algorithms/symmetric/aes_cfb.cpp File Reference . . . 515
10.130.1 Detailed Description . . . 515
10.131 src/algorithms/symmetric/aes_ctr.cpp File Reference . . . 515
10.131.1 Detailed Description . . . 516
10.132 src/algorithms/symmetric/aes_ecb.cpp File Reference . . . 516
10.132.1 Detailed Description . . . 517
10.133 src/algorithms/symmetric/aes_gcm.cpp File Reference . . . 517
10.134 src/algorithms/symmetric/aes_ofb.cpp File Reference . . . 517
10.134.1 Detailed Description . . . 518
10.135 src/algorithms/symmetric/aes_xts.cpp File Reference . . . 518
10.135.1 Detailed Description . . . 519
10.136 src/algorithms/symmetric/aria_gcm.cpp File Reference . . . 519
10.136.1 Detailed Description . . . 520
10.137 src/algorithms/symmetric/camellia_gcm.cpp File Reference . . . 520
10.137.1 Detailed Description . . . 521
10.138 src/algorithms/symmetric/chacha20_poly1305.cpp File Reference . . . 521
10.139 src/algorithms/symmetric/serpent_gcm.cpp File Reference . . . 522
10.140 src/algorithms/symmetric/sm4_gcm.cpp File Reference . . . 522
10.140.1 Detailed Description . . . 523
10.141 src/algorithms/symmetric/triple_des.cpp File Reference . . . 523
10.141.1 Detailed Description . . . 524
10.142 src/algorithms/symmetric/twofish_gcm.cpp File Reference . . . 524
10.142.1 Detailed Description . . . 524
10.143 src/archive/archive_format.cpp File Reference . . . 524
10.144 src/cli/app.cpp File Reference . . . 525
10.145 src/cli/commands/archive_cmd.cpp File Reference . . . 526
10.146 src/cli/commands/benchmark_cmd.cpp File Reference . . . 526
10.146.1 Detailed Description . . . 527
10.147 src/cli/commands/compress_cmd.cpp File Reference . . . 527
10.148 src/cli/commands/config_cmd.cpp File Reference . . . 527
10.149 src/cli/commands/decompress_cmd.cpp File Reference . . . 528
10.150 src/cli/commands/decrypt_cmd.cpp File Reference . . . 528
10.151 src/cli/commands/dump_cmd.cpp File Reference . . . 529

10.152 src/cli/commands/encrypt_cmd.cpp File Reference . . . 529
10.153 src/cli/commands/hash_cmd.cpp File Reference . . . 530
10.154 src/cli/commands/info_cmd.cpp File Reference . . . 531
10.155 src/cli/commands/keygen_cmd.cpp File Reference . . . 531
10.155.1 Detailed Description . . . 532
10.156 src/cli/commands/keyinfo_cmd.cpp File Reference . . . 532
10.157 src/cli/commands/list_cmd.cpp File Reference . . . 532
10.158 src/cli/commands/sign_cmd.cpp File Reference . . . 533
10.159 src/cli/commands/stego_cmd.cpp File Reference . . . 533
10.160 src/cli/commands/verify_cmd.cpp File Reference . . . 534
10.161 src/compression/compressor.cpp File Reference . . . 535
10.162 src/core/crypto_engine.cpp File Reference . . . 535
10.163 src/core/modes.cpp File Reference . . . 536
10.164 src/core/streaming.cpp File Reference . . . 537
10.164.1 Detailed Description . . . 537
10.165 src/core/types.cpp File Reference . . . 538
10.166 src/format/file_format.cpp File Reference . . . 538
10.167 src/format/file_header.cpp File Reference . . . 539
10.168 src/main.cpp File Reference . . . 539
10.168.1 Function Documentation . . . 540
10.168.1.1 main() . . . 540
10.169 src/steganography/lsb.cpp File Reference . . . 540
10.169.1 Macro Definition Documentation . . . 541
10.169.1.1 STB_IMAGE_IMPLEMENTATION . . . 541
10.169.1.2 STB_IMAGE_WRITE_IMPLEMENTATION . . . 541
10.169.1.3 STBI_FAILURE_USERMSG . . . 541
10.170 src/utils/config.cpp File Reference . . . 541
10.171 src/utils/console.cpp File Reference . . . 542
10.172 src/utils/crypto_utils.cpp File Reference . . . 542
10.173 src/utils/file_io.cpp File Reference . . . 543
10.174 src/utils/password.cpp File Reference . . . 544
10.175 src/utils/progress.cpp File Reference . . . 544
10.175.1 Macro Definition Documentation . . . 545
10.175.1.1 FILENO . . . 545
10.175.1.2 ISATTY . . . 545
10.176 src/utils/table_formatter.cpp File Reference . . . 545

Index **547**

Chapter 1

FileVault

Professional cross-platform file encryption CLI tool built with modern C++20 and Botan 3.x cryptographic library.

[](LICENSE)

1.1 Features

1.1.1 Documentation link

- https://vuongdat67.github.io/FileVault_mkdocs/
- https://vuongdat67.github.io/NT140.Q11.ANTT-Group11/doxygen/html/md_README.html
- https://vuongdat67.github.io/NT140.Q11.ANTT-Group11/doxygen/latex/FileVault-API.pdf

1.1.2 Modern Encryption

- **AEAD Ciphers**: AES-GCM (128/192/256), ChaCha20-Poly1305, Serpent-GCM, Twofish-GCM
- **International Standards**: Camellia-GCM, ARIA-GCM (Korea), SM4-GCM (China)
- **Legacy Support**: AES-CBC/CTR/CFB/OFB/XTS, 3DES (for compatibility)
- **Asymmetric**: RSA (2048/3072/4096), ECC (P-256/P-384/P-521)
- **Post-Quantum Cryptography (PQC)**:
  - Kyber-512/768/1024 (ML-KEM) - Key encapsulation
  - Dilithium-2/3/5 (ML-DSA) - Digital signatures
  - KyberHybrid - Quantum-resistant hybrid encryption
- **Classical** (educational): Caesar, Vigenre, Playfair, Hill, Substitution

1.1.3 Key Derivation

- **Argon2id** - Memory-hard, recommended
- **Scrypt** - Memory-hard alternative
- **PBKDF2** (SHA-256/SHA-512) - Legacy compatible

1.1.4 Compression

- **ZLIB** - Fast, good ratio
- **LZMA** - Best ratio, slower
- **BZIP2** - Balanced (coming soon)

1.1.5 Additional Features

- **Steganography** - Hide data in images (LSB)
- **Archive** - Encrypt multiple files
- **Hashing** - SHA-256, SHA-512, SHA-3, BLAKE2b, BLAKE3
- **Benchmarks** - Performance testing

1.2 Quick Start

1.2.1 Windows (MSVC)

```
# Setup
.\scripts\setup-msvc.ps1

# Build
.\scripts\build-msvc.ps1 -Test
```

1.2.2 Windows (MinGW/MSYS2)

```
# In MSYS2 UCRT64 terminal
./scripts/setup-mingw.sh
./scripts/build-mingw.sh -t
```

1.2.3 Linux

```
./scripts/setup-linux.sh    # or --clang for Clang
./scripts/build-linux.sh -t
```

1.2.4 macOS

```
./scripts/setup-macos.sh
./scripts/build-macos.sh -t
```

1.3 Usage

1.3.1 Basic Encryption

```
# Encrypt with AES-256-GCM (default)
filevault encrypt secret.txt

# Decrypt
filevault decrypt secret.txt.fvlt

# With options
filevault encrypt data.zip -a chacha20-poly1305 -s paranoid --compression lzma
```

1.3.2 Mode Presets

```
filevault encrypt file.txt --mode basic      # AES-128-GCM, fast
filevault encrypt file.txt --mode standard   # AES-256-GCM, balanced
filevault encrypt file.txt --mode advanced   # ChaCha20-Poly1305, max security
```

1.3.3 Asymmetric Encryption

```
# Generate RSA key pair
filevault keygen --algorithm rsa-4096 --output mykey

# Encrypt with public key
filevault encrypt secret.txt --pubkey mykey.pub

# Decrypt with private key
filevault decrypt secret.txt.fvlt --privkey mykey.pem
```

1.3.4 Post-Quantum Cryptography (PQC)

```
# Generate Kyber keypair (quantum-resistant)
filevault keygen --algorithm kyber-1024 --output quantum-key

# Encrypt with Kyber-Hybrid (combines classical + PQC)
filevault encrypt secret.txt --algorithm kyber-1024-hybrid

# Generate Dilithium signing keypair
filevault keygen --algorithm dilithium-5 --output dilithium-key

# Hybrid encryption (recommended for quantum threat)
filevault encrypt data.zip -a kyber-1024-hybrid
```

1.3.5 Steganography

```
# Hide data in image
filevault stego embed message.txt cover.png -o hidden.png

# Extract hidden data
filevault stego extract hidden.png -o recovered.txt

# Check capacity
filevault stego capacity photo.jpg
```

1.3.6 Archive

```
# Create encrypted archive
filevault archive create documents/ -o backup.fva

# Extract archive
filevault archive extract backup.fva -o restored/

# List contents
filevault archive list backup.fva
```

1.3.7 Hash

```
filevault hash document.pdf                    # SHA-256 (default)
filevault hash file.iso --algorithm blake2b   # BLAKE2b
filevault hash ./folder/ --recursive          # All files
```

1.3.8 Benchmark

```
filevault benchmark                            # All algorithms
filevault benchmark --algorithm aes-256-gcm   # Specific
filevault benchmark --json -o results.json    # Export
```

1.3.9 List & Info

```
filevault list algorithms    # Supported algorithms
filevault list kdfs          # Key derivation functions
filevault info encrypted.fvlt  # File metadata
```

1.4 Building

1.4.1 Requirements

- **C++20** compatible compiler
- **CMake** >= 3.20
- **Conan** 2.x
- **Ninja** (recommended)

1.4.2 Supported Platforms

Platform	Compiler	Status
Linux	GCC 13+	
Linux	Clang 17+	
Windows	MSVC 2022/2026	
Windows	MinGW GCC 14+	
macOS	Apple Clang 16+	

1.4.3 Manual Build

```
mkdir build && cd build
conan install .. --output-folder=. --build=missing
cmake --preset conan-release -DBUILD_TESTS=ON
cmake --build build/Release --parallel
```

See docs/BUILD.md for detailed instructions.

1.5 Algorithm Comparison

1.5.1 Symmetric (AEAD - Authenticated Encryption)

Algorithm	Key Size	Speed	Security	Use Case
AES-256-GCM	256-bit			General purpose
ChaCha20-Poly1305	256-bit			Mobile, no AES-NI
Serpent-256-GCM	256-bit			High security
Camellia-256-GCM	256-bit			Japan standard
ARIA-256-GCM	256-bit			Korea standard
SM4-GCM	128-bit			China standard

1.5.2 Asymmetric & Post-Quantum

Algorithm	Key Size	Type	Security	Quantum Resistant
RSA-4096	4096-bit	Classical		Vulnerable
ECC-P521	521-bit	Classical		Vulnerable
Kyber-512	N/A	PQC KEM		NIST Level 1
Kyber-768	N/A	PQC KEM		NIST Level 3
Kyber-1024	N/A	PQC KEM		NIST Level 5

Algorithm	Key Size	Type	Security	Quantum Resistant
KyberHybrid	Combined	PQC+AES		Defense in depth
Dilithium-5	N/A	PQC Signature		NIST Level 5

1.5.3 Key Derivation

KDF	Memory	Speed	Resistance
Argon2id	64MB+	Slow	GPU, ASIC
Scrypt	32MB+	Slow	GPU
PBKDF2	Minimal	Fast	Brute force only

1.5.4 Benchmark Results (1 MB data)

Symmetric Encryption:

- AES-256-GCM: ∼700 MB/s (hardware accelerated)
- ChaCha20-Poly1305: ∼600 MB/s (software optimized)
- Kyber-1024-Hybrid: ∼650 MB/s (PQC + AES-GCM)

Asymmetric Operations:

- RSA-4096 Keygen: ∼1.7 seconds
- ECC-P521 Keygen: ∼10 ms
- Kyber-1024 Keygen: ∼0.4 ms

PQC Performance:

- Kyber-1024 KEM Encapsulation: ∼0.6 ms
- Kyber-1024 KEM Decapsulation: ∼0.8 ms
- Dilithium-5 Sign: ∼1.6 ms
- Dilithium-5 Verify: ∼0.8 ms

1.6 Testing

```
# Run all tests
ctest --test-dir build --output-on-failure

# Run specific test
ctest --test-dir build -R "AES_GCM"

# With verbose output
ctest --test-dir build -V
```

1.6.1 Test Categories

- **Unit tests** - Individual components
- **Integration tests** - Full encrypt/decrypt flow
- **Security tests** - Nonce uniqueness, timing attacks
- **NIST vectors** - Standard test vectors

1.7 Project Structure

```
filevault/
 include/filevault/     # Headers
    algorithms/        # Crypto algorithms
    cli/               # CLI commands
    core/              # Core types & engine
    utils/             # Utilities
 src/                   # Implementation
 tests/                 # Test suites
 scripts/               # Build scripts
 docs/                  # Documentation
```

1.8 Configuration

1.8.1 Security Levels

Level	KDF Iterations	Memory	Description
weak	3	64MB	Fast, testing
medium	10	128MB	Balanced
strong	20	256MB	Recommended
paranoid	50	512MB	Maximum

1.8.2 Config File

```
filevault config set default-algorithm aes-256-gcm
filevault config set default-kdf argon2id
filevault config show
```

1.9 Documentation

- BUILD.md - Build instructions
- `.github/copilot/` - Architecture & coding standards

1.10 Contributing

1. Fork the repository
2. Create feature branch: `git checkout -b feature/amazing`
3. Commit changes: `git commit -m 'Add amazing feature'`
4. Push: `git push origin feature/amazing`
5. Open Pull Request

1.11 License

MIT License - see LICENSE for details.

1.12 Acknowledgments

- `Botan` - Crypto library
- `CLI11` - CLI parser
- `spdlog` - Logging
- `indicators` - Progress bars

Chapter 2

Directory Hierarchy

2.1 Directories

algorithms . . . 29
asymmetric . . . 31
ecc.hpp . . . 395
rsa.hpp . . . 398
classical . . . 32
caesar.hpp . . . 400
classical_ciphers.hpp . . . 401
hill.hpp . . . 403
playfair.hpp . . . 405
substitution.hpp . . . 407
vigenere.hpp . . . 409
pqc . . . 40
post_quantum.hpp . . . 411
symmetric . . . 42
aes_cbc.hpp . . . 414
aes_cfb.hpp . . . 416
aes_ctr.hpp . . . 418
aes_ecb.hpp . . . 420
aes_gcm.hpp . . . 422
aes_ofb.hpp . . . 424
aes_xts.hpp . . . 426
aria_gcm.hpp . . . 428
camellia_gcm.hpp . . . 430
chacha20_poly1305.hpp . . . 432
serpent_gcm.hpp . . . 434
sm4_gcm.hpp . . . 436
triple_des.hpp . . . 438
twofish_gcm.hpp . . . 440
algorithms . . . 29
asymmetric . . . 31
ecc.cpp . . . 506
rsa.cpp . . . 507
classical . . . 32
caesar.cpp . . . 508
classical_ciphers.cpp . . . 509
hill.cpp . . . 509
playfair.cpp . . . 510
substitution.cpp . . . 511
vigenere.cpp . . . 512
pqc . . . 40

post_quantum.cpp . . . 513
symmetric . . . 43
aes_cbc.cpp . . . 514
aes_cfb.cpp . . . 515
aes_ctr.cpp . . . 515
aes_ecb.cpp . . . 516
aes_gcm.cpp . . . 517
aes_ofb.cpp . . . 517
aes_xts.cpp . . . 518
aria_gcm.cpp . . . 519
camellia_gcm.cpp . . . 520
chacha20_poly1305.cpp . . . 521
serpent_gcm.cpp . . . 522
sm4_gcm.cpp . . . 522
triple_des.cpp . . . 523
twofish_gcm.cpp . . . 524
archive . . . 30
archive_format.hpp . . . 442
archive . . . 30
archive_format.cpp . . . 524
asymmetric . . . 31
ecc.hpp . . . 395
rsa.hpp . . . 398
asymmetric . . . 31
ecc.cpp . . . 506
rsa.cpp . . . 507
classical . . . 32
caesar.hpp . . . 400
classical_ciphers.hpp . . . 401
hill.hpp . . . 403
playfair.hpp . . . 405
substitution.hpp . . . 407
vigenere.hpp . . . 409
classical . . . 32
caesar.cpp . . . 508
classical_ciphers.cpp . . . 509
hill.cpp . . . 509
playfair.cpp . . . 510
substitution.cpp . . . 511
vigenere.cpp . . . 512
cli . . . 33
commands . . . 34
archive_cmd.hpp . . . 447
benchmark_cmd.hpp . . . 449
compress_cmd.hpp . . . 451
config_cmd.hpp . . . 453
decompress_cmd.hpp . . . 454
decrypt_cmd.hpp . . . 456
dump_cmd.hpp . . . 457
encrypt_cmd.hpp . . . 459
hash_cmd.hpp . . . 460
info_cmd.hpp . . . 462
keygen_cmd.hpp . . . 464
keyinfo_cmd.hpp . . . 466
list_cmd.hpp . . . 467
sign_cmd.hpp . . . 468

stego_cmd.hpp . . . 470
verify_cmd.hpp . . . 472
app.hpp . . . 444
command.hpp . . . 446
cli . . . 34
commands . . . 35
archive_cmd.cpp . . . 526
benchmark_cmd.cpp . . . 526
compress_cmd.cpp . . . 527
config_cmd.cpp . . . 527
decompress_cmd.cpp . . . 528
decrypt_cmd.cpp . . . 528
dump_cmd.cpp . . . 529
encrypt_cmd.cpp . . . 529
hash_cmd.cpp . . . 530
info_cmd.cpp . . . 531
keygen_cmd.cpp . . . 531
keyinfo_cmd.cpp . . . 532
list_cmd.cpp . . . 532
sign_cmd.cpp . . . 533
stego_cmd.cpp . . . 533
verify_cmd.cpp . . . 534
app.cpp . . . 525
commands . . . 34
archive_cmd.hpp . . . 447
benchmark_cmd.hpp . . . 449
compress_cmd.hpp . . . 451
config_cmd.hpp . . . 453
decompress_cmd.hpp . . . 454
decrypt_cmd.hpp . . . 456
dump_cmd.hpp . . . 457
encrypt_cmd.hpp . . . 459
hash_cmd.hpp . . . 460
info_cmd.hpp . . . 462
keygen_cmd.hpp . . . 464
keyinfo_cmd.hpp . . . 466
list_cmd.hpp . . . 467
sign_cmd.hpp . . . 468
stego_cmd.hpp . . . 470
verify_cmd.hpp . . . 472
commands . . . 35
archive_cmd.cpp . . . 526
benchmark_cmd.cpp . . . 526
compress_cmd.cpp . . . 527
config_cmd.cpp . . . 527
decompress_cmd.cpp . . . 528
decrypt_cmd.cpp . . . 528
dump_cmd.cpp . . . 529
encrypt_cmd.cpp . . . 529
hash_cmd.cpp . . . 530
info_cmd.cpp . . . 531
keygen_cmd.cpp . . . 531
keyinfo_cmd.cpp . . . 532
list_cmd.cpp . . . 532
sign_cmd.cpp . . . 533
stego_cmd.cpp . . . 533
verify_cmd.cpp . . . 534

compression . . . 36
compressor.hpp . . . 473
compression . . . 36
compressor.cpp . . . 535
core . . . 37
crypto_algorithm.hpp . . . 475
crypto_engine.hpp . . . 477
file_format.hpp . . . 478
modes.hpp . . . 482
result.hpp . . . 484
streaming.hpp . . . 486
types.hpp . . . 488
core . . . 37
crypto_engine.cpp . . . 535
modes.cpp . . . 536
streaming.cpp . . . 537
types.cpp . . . 538
filevault . . . 38
algorithms . . . 29
asymmetric . . . 31
ecc.hpp . . . 395
rsa.hpp . . . 398
classical . . . 32
caesar.hpp . . . 400
classical_ciphers.hpp . . . 401
hill.hpp . . . 403
playfair.hpp . . . 405
substitution.hpp . . . 407
vigenere.hpp . . . 409
pqc . . . 40
post_quantum.hpp . . . 411
symmetric . . . 42
aes_cbc.hpp . . . 414
aes_cfb.hpp . . . 416
aes_ctr.hpp . . . 418
aes_ecb.hpp . . . 420
aes_gcm.hpp . . . 422
aes_ofb.hpp . . . 424
aes_xts.hpp . . . 426
aria_gcm.hpp . . . 428
camellia_gcm.hpp . . . 430
chacha20_poly1305.hpp . . . 432
serpent_gcm.hpp . . . 434
sm4_gcm.hpp . . . 436
triple_des.hpp . . . 438
twofish_gcm.hpp . . . 440
archive . . . 30
archive_format.hpp . . . 442
cli . . . 33
commands . . . 34
archive_cmd.hpp . . . 447
benchmark_cmd.hpp . . . 449
compress_cmd.hpp . . . 451
config_cmd.hpp . . . 453
decompress_cmd.hpp . . . 454
decrypt_cmd.hpp . . . 456

dump_cmd.hpp . . . 457
encrypt_cmd.hpp . . . 459
hash_cmd.hpp . . . 460
info_cmd.hpp . . . 462
keygen_cmd.hpp . . . 464
keyinfo_cmd.hpp . . . 466
list_cmd.hpp . . . 467
sign_cmd.hpp . . . 468
stego_cmd.hpp . . . 470
verify_cmd.hpp . . . 472
app.hpp . . . 444
command.hpp . . . 446
compression . . . 36
compressor.hpp . . . 473
core . . . 37
crypto_algorithm.hpp . . . 475
crypto_engine.hpp . . . 477
file_format.hpp . . . 478
modes.hpp . . . 482
result.hpp . . . 484
streaming.hpp . . . 486
types.hpp . . . 488
format . . . 38
file_header.hpp . . . 493
steganography . . . 41
lsb.hpp . . . 495
utils . . . 44
config.hpp . . . 497
console.hpp . . . 499
crypto_utils.hpp . . . 500
file_io.hpp . . . 501
password.hpp . . . 502
progress.hpp . . . 503
table_formatter.hpp . . . 505
format . . . 38
file_header.hpp . . . 493
format . . . 39
file_format.cpp . . . 538
file_header.cpp . . . 539
include . . . 39
filevault . . . 38
algorithms . . . 29
asymmetric . . . 31
ecc.hpp . . . 395
rsa.hpp . . . 398
classical . . . 32
caesar.hpp . . . 400
classical_ciphers.hpp . . . 401
hill.hpp . . . 403
playfair.hpp . . . 405
substitution.hpp . . . 407
vigenere.hpp . . . 409
pqc . . . 40
post_quantum.hpp . . . 411
symmetric . . . 42
aes_cbc.hpp . . . 414

aes_cfb.hpp . . . 416
aes_ctr.hpp . . . 418
aes_ecb.hpp . . . 420
aes_gcm.hpp . . . 422
aes_ofb.hpp . . . 424
aes_xts.hpp . . . 426
aria_gcm.hpp . . . 428
camellia_gcm.hpp . . . 430
chacha20_poly1305.hpp . . . 432
serpent_gcm.hpp . . . 434
sm4_gcm.hpp . . . 436
triple_des.hpp . . . 438
twofish_gcm.hpp . . . 440
archive . . . 30
archive_format.hpp . . . 442
cli . . . 33
commands . . . 34
archive_cmd.hpp . . . 447
benchmark_cmd.hpp . . . 449
compress_cmd.hpp . . . 451
config_cmd.hpp . . . 453
decompress_cmd.hpp . . . 454
decrypt_cmd.hpp . . . 456
dump_cmd.hpp . . . 457
encrypt_cmd.hpp . . . 459
hash_cmd.hpp . . . 460
info_cmd.hpp . . . 462
keygen_cmd.hpp . . . 464
keyinfo_cmd.hpp . . . 466
list_cmd.hpp . . . 467
sign_cmd.hpp . . . 468
stego_cmd.hpp . . . 470
verify_cmd.hpp . . . 472
app.hpp . . . 444
command.hpp . . . 446
compression . . . 36
compressor.hpp . . . 473
core . . . 37
crypto_algorithm.hpp . . . 475
crypto_engine.hpp . . . 477
file_format.hpp . . . 478
modes.hpp . . . 482
result.hpp . . . 484
streaming.hpp . . . 486
types.hpp . . . 488
format . . . 38
file_header.hpp . . . 493
steganography . . . 41
lsb.hpp . . . 495
utils . . . 44
config.hpp . . . 497
console.hpp . . . 499
crypto_utils.hpp . . . 500
file_io.hpp . . . 501
password.hpp . . . 502
progress.hpp . . . 503
table_formatter.hpp . . . 505

pqc . . . 40
post_quantum.hpp . . . 411
pqc . . . 40
post_quantum.cpp . . . 513
src . . . 41
algorithms . . . 29
asymmetric . . . 31
ecc.cpp . . . 506
rsa.cpp . . . 507
classical . . . 32
caesar.cpp . . . 508
classical_ciphers.cpp . . . 509
hill.cpp . . . 509
playfair.cpp . . . 510
substitution.cpp . . . 511
vigenere.cpp . . . 512
pqc . . . 40
post_quantum.cpp . . . 513
symmetric . . . 43
aes_cbc.cpp . . . 514
aes_cfb.cpp . . . 515
aes_ctr.cpp . . . 515
aes_ecb.cpp . . . 516
aes_gcm.cpp . . . 517
aes_ofb.cpp . . . 517
aes_xts.cpp . . . 518
aria_gcm.cpp . . . 519
camellia_gcm.cpp . . . 520
chacha20_poly1305.cpp . . . 521
serpent_gcm.cpp . . . 522
sm4_gcm.cpp . . . 522
triple_des.cpp . . . 523
twofish_gcm.cpp . . . 524
archive . . . 30
archive_format.cpp . . . 524
cli . . . 34
commands . . . 35
archive_cmd.cpp . . . 526
benchmark_cmd.cpp . . . 526
compress_cmd.cpp . . . 527
config_cmd.cpp . . . 527
decompress_cmd.cpp . . . 528
decrypt_cmd.cpp . . . 528
dump_cmd.cpp . . . 529
encrypt_cmd.cpp . . . 529
hash_cmd.cpp . . . 530
info_cmd.cpp . . . 531
keygen_cmd.cpp . . . 531
keyinfo_cmd.cpp . . . 532
list_cmd.cpp . . . 532
sign_cmd.cpp . . . 533
stego_cmd.cpp . . . 533
verify_cmd.cpp . . . 534
app.cpp . . . 525
compression . . . 36
compressor.cpp . . . 535
core . . . 37

crypto_engine.cpp . . . 535
modes.cpp . . . 536
streaming.cpp . . . 537
types.cpp . . . 538
format . . . 39
file_format.cpp . . . 538
file_header.cpp . . . 539
steganography . . . 42
lsb.cpp . . . 540
utils . . . 45
config.cpp . . . 541
console.cpp . . . 542
crypto_utils.cpp . . . 542
file_io.cpp . . . 543
password.cpp . . . 544
progress.cpp . . . 544
table_formatter.cpp . . . 545
main.cpp . . . 539
steganography . . . 41
lsb.hpp . . . 495
steganography . . . 42
lsb.cpp . . . 540
symmetric . . . 42
aes_cbc.hpp . . . 414
aes_cfb.hpp . . . 416
aes_ctr.hpp . . . 418
aes_ecb.hpp . . . 420
aes_gcm.hpp . . . 422
aes_ofb.hpp . . . 424
aes_xts.hpp . . . 426
aria_gcm.hpp . . . 428
camellia_gcm.hpp . . . 430
chacha20_poly1305.hpp . . . 432
serpent_gcm.hpp . . . 434
sm4_gcm.hpp . . . 436
triple_des.hpp . . . 438
twofish_gcm.hpp . . . 440
symmetric . . . 43
aes_cbc.cpp . . . 514
aes_cfb.cpp . . . 515
aes_ctr.cpp . . . 515
aes_ecb.cpp . . . 516
aes_gcm.cpp . . . 517
aes_ofb.cpp . . . 517
aes_xts.cpp . . . 518
aria_gcm.cpp . . . 519
camellia_gcm.cpp . . . 520
chacha20_poly1305.cpp . . . 521
serpent_gcm.cpp . . . 522
sm4_gcm.cpp . . . 522
triple_des.cpp . . . 523
twofish_gcm.cpp . . . 524
utils . . . 44
config.hpp . . . 497
console.hpp . . . 499
crypto_utils.hpp . . . 500

file_io.hpp 501
password.hpp 502
progress.hpp 503
table_formatter.hpp 505
utils 45
config.cpp 541
console.cpp 542
crypto_utils.cpp 542
file_io.cpp 543
password.cpp 544
progress.cpp 544
table_formatter.cpp 545

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

filevault . . . 47
filevault::algorithms . . . 47
filevault::algorithms::asymmetric . . . 47
filevault::algorithms::classical . . . 49
filevault::algorithms::pqc . . . 49
filevault::algorithms::symmetric . . . 50
filevault::archive . . . 51
filevault::cli . . . 52
filevault::cli::commands . . . 52
filevault::compression . . . 53
filevault::core . . . 53
filevault::core::presets . . . 61
filevault::format . . . 62
filevault::steganography . . . 62
filevault::utils . . . 62

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

filevault::cli::Application . . . 103
filevault::archive::ArchiveFormat . . . 112
filevault::core::Argon2Params . . . 116
filevault::cli::AsymmetricBenchmarkResult . . . 123
filevault::cli::BenchmarkResult . . . 135
filevault::utils::BlockProgressBar . . . 136
filevault::core::ChunkInfo . . . 155
filevault::compression::CompressionResult . . . 162
filevault::compression::CompressionService . . . 163
filevault::utils::Config . . . 165
filevault::utils::Console . . . 175
filevault::core::CryptoEngine . . . 182
filevault::core::CryptoResult . . . 188
filevault::utils::CryptoUtils . . . 189
filevault::algorithms::pqc::Dilithium . . . 199
filevault::algorithms::asymmetric::ECCKeyPair . . . 211
filevault::algorithms::asymmetric::ECDH . . . 212
filevault::algorithms::asymmetric::ECDHResult . . . 214
filevault::algorithms::asymmetric::ECDSA . . . 215
filevault::algorithms::asymmetric::ECDSASignResult . . . 218
filevault::core::EncryptionConfig . . . 224
filevault::archive::FileEntry . . . 226
filevault::core::FileFormatHandler . . . 228
filevault::core::FileHeader . . . 232
filevault::format::FileHeader . . . 235
filevault::cli::InfoCommand::FileInfo . . . 246
filevault::utils::FileIO . . . 247
filevault::core::HashConfig . . . 256
filevault::cli::ICommand . . . 264
- filevault::cli::BenchmarkCommand . . . 124
- filevault::cli::CompressCommand . . . 156
- filevault::cli::ConfigCommand . . . 170
- filevault::cli::DecompressCommand . . . 190
- filevault::cli::DecryptCommand . . . 195
- filevault::cli::EncryptCommand . . . 218
- filevault::cli::HashCommand . . . 250
- filevault::cli::InfoCommand . . . 272
- filevault::cli::KeygenCommand . . . 277
- filevault::cli::ListCommand . . . 294
- filevault::cli::commands::ArchiveCommand . . . 106

filevault::cli::commands::DumpCommand . . . 203
filevault::cli::commands::KeyInfoCommand . . . 281
filevault::cli::commands::SignCommand . . . 340
filevault::cli::commands::StegoCommand . . . 349
filevault::cli::commands::VerifyCommand . . . 382
filevault::compression::ICompressor . . . 267
filevault::compression::Bzip2Compressor . . . 137
filevault::compression::LzmaCompressor . . . 300
filevault::compression::ZlibCompressor . . . 391
filevault::core::ICryptoAlgorithm . . . 268
filevault::algorithms::asymmetric::ECCHybrid . . . 206
filevault::algorithms::asymmetric::RSA . . . 326
filevault::algorithms::classical::Caesar . . . 138
filevault::algorithms::classical::Caesar . . . 138
filevault::algorithms::classical::HillCipher . . . 257
filevault::algorithms::classical::Playfair . . . 313
filevault::algorithms::classical::Playfair . . . 313
filevault::algorithms::classical::SubstitutionCipher . . . 360
filevault::algorithms::classical::Vigenere . . . 385
filevault::algorithms::classical::Vigenere . . . 385
filevault::algorithms::pqc::Kyber . . . 284
filevault::algorithms::pqc::KyberHybrid . . . 289
filevault::algorithms::symmetric::AES_CBC . . . 65
filevault::algorithms::symmetric::AES_CFB . . . 70
filevault::algorithms::symmetric::AES_CTR . . . 76
filevault::algorithms::symmetric::AES_ECB . . . 80
filevault::algorithms::symmetric::AES_GCM . . . 86
filevault::algorithms::symmetric::AES_OFB . . . 92
filevault::algorithms::symmetric::AES_XTS . . . 97
filevault::algorithms::symmetric::ARIA_GCM . . . 117
filevault::algorithms::symmetric::Camellia_GCM . . . 145
filevault::algorithms::symmetric::ChaCha20Poly1305 . . . 150
filevault::algorithms::symmetric::SM4_GCM . . . 343
filevault::algorithms::symmetric::Serpent_GCM . . . 334
filevault::algorithms::symmetric::TripleDES . . . 371
filevault::algorithms::symmetric::Twofish_GCM . . . 375
filevault::steganography::LSBSteganography . . . 297
filevault::core::ModePreset . . . 302
filevault::utils::Password . . . 305
filevault::core::PasswordAnalysis . . . 311
filevault::core::PBKDF2Params . . . 312
filevault::cli::PQCBenchmarkResult . . . 320
filevault::algorithms::pqc::PQKeyPair . . . 321
filevault::utils::ProgressBar . . . 321
filevault::core::Result< T > . . . 323
filevault::core::Result< void > . . . 325
filevault::algorithms::asymmetric::RSAKeyPair . . . 332
filevault::core::ScryptParams . . . 333
filevault::cli::SignatureBenchmarkResult . . . 339
filevault::core::StreamingConfig . . . 354
filevault::core::StreamingCrypto . . . 355
filevault::core::StreamingResult . . . 360
filevault::utils::TableBuilder . . . 366
filevault::utils::TableFormatter . . . 367

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

filevault::algorithms::symmetric::AES_CBC
AES-CBC encryption (legacy mode, not authenticated) . . . 65
filevault::algorithms::symmetric::AES_CFB
AES-CFB encryption algorithm . . . 70
filevault::algorithms::symmetric::AES_CTR
AES-CTR encryption (stream cipher mode, not authenticated) . . . 76
filevault::algorithms::symmetric::AES_ECB
AES-ECB encryption algorithm . . . 80
filevault::algorithms::symmetric::AES_GCM
AES-GCM AEAD encryption . . . 86
filevault::algorithms::symmetric::AES_OFB
AES-OFB encryption algorithm . . . 92
filevault::algorithms::symmetric::AES_XTS
AES-XTS encryption algorithm . . . 97
filevault::cli::Application
Main CLI application . . . 103
filevault::cli::commands::ArchiveCommand
Archive command - compress and encrypt multiple files . . . 106
filevault::archive::ArchiveFormat
Simple archive format handler . . . 112
filevault::core::Argon2Params
KDF parameters for Argon2 . . . 116
filevault::algorithms::symmetric::ARIA_GCM
ARIA-GCM AEAD encryption . . . 117
filevault::cli::AsymmetricBenchmarkResult . . . 123
filevault::cli::BenchmarkCommand . . . 124
filevault::cli::BenchmarkResult . . . 135
filevault::utils::BlockProgressBar
Block-style progress bar . . . 136
filevault::compression::Bzip2Compressor
BZIP2 compressor (better ratio, slower) . . . 137
filevault::algorithms::classical::Caesar
Caesar Cipher - Educational only . . . 138
filevault::algorithms::symmetric::Camellia_GCM
Camellia-GCM AEAD encryption . . . 145
filevault::algorithms::symmetric::ChaCha20Poly1305
ChaCha20-Poly1305 AEAD encryption . . . 150
filevault::core::ChunkInfo
Chunk information for streaming encryption . . . 155

filevault::cli::CompressCommand
Compress command - standalone compression/decompression . . . 156
filevault::compression::CompressionResult
Compression result . . . 162
filevault::compression::CompressionService
Compression service - factory for compressors . . . 163
filevault::utils::Config
Configuration manager for FileVault . . . 165
filevault::cli::ConfigCommand
Config command - manage FileVault configuration . . . 170
filevault::utils::Console
Console utilities for colored output . . . 175
filevault::core::CryptoEngine
Main cryptographic engine Manages algorithms and provides key derivation . . . 182
filevault::core::CryptoResult
Result of cryptographic operations . . . 188
filevault::utils::CryptoUtils
Crypto utility functions . . . 189
filevault::cli::DecompressCommand
Decompress command - alias for compress -d . . . 190
filevault::cli::DecryptCommand . . . 195
filevault::algorithms::pqc::Dilithium
Dilithium Digital Signature Algorithm . . . 199
filevault::cli::commands::DumpCommand
Command to dump file content in various formats (hex, binary, base64) . . . 203
filevault::algorithms::asymmetric::ECCHybrid
Hybrid encryption using ECDH + AES-GCM . . . 206
filevault::algorithms::asymmetric::ECCKeyPair
ECC key pair (ECDH or ECDSA) . . . 211
filevault::algorithms::asymmetric::ECDH
Elliptic Curve Diffie-Hellman (ECDH) key exchange . . . 212
filevault::algorithms::asymmetric::ECDHResult
ECDH Key Exchange result . . . 214
filevault::algorithms::asymmetric::ECDSA
Elliptic Curve Digital Signature Algorithm (ECDSA) . . . 215
filevault::algorithms::asymmetric::ECDSASignResult
ECDSA signature result . . . 218
filevault::cli::EncryptCommand
Encrypt command . . . 218
filevault::core::EncryptionConfig
Configuration for encryption operations . . . 224
filevault::archive::FileEntry
Archive file entry metadata . . . 226
filevault::core::FileFormatHandler
File format handler . . . 228
filevault::core::FileHeader
File format header . . . 232
filevault::format::FileHeader
FileVault encrypted file format . . . 235
filevault::cli::InfoCommand::FileInfo
Parse encrypted file header . . . 246
filevault::utils::FileIO
File I/O utilities . . . 247
filevault::cli::HashCommand
Hash command - Calculate cryptographic hashes . . . 250
filevault::core::HashConfig
Configuration for hashing operations . . . 256

filevault::algorithms::classical::HillCipher
Hill cipher (2x2 matrix) . . . 257
filevault::cli::ICommand
Base interface for CLI commands . . . 264
filevault::compression::ICompressor
Interface for compression algorithms . . . 267
filevault::core::ICryptoAlgorithm
Interface for cryptographic algorithms . . . 268
filevault::cli::InfoCommand
Info command - display encrypted file metadata . . . 272
filevault::cli::KeygenCommand
Command to generate key pairs for asymmetric encryption . . . 277
filevault::cli::commands::KeyInfoCommand
Command to display information about cryptographic keys . . . 281
filevault::algorithms::pqc::Kyber
Kyber Key Encapsulation Mechanism (KEM) . . . 284
filevault::algorithms::pqc::KyberHybrid
Hybrid encryption combining Kyber with AES-GCM . . . 289
filevault::cli::ListCommand . . . 294
filevault::steganography::LSBSteganography
LSB (Least Significant Bit) Steganography . . . 297
filevault::compression::LzmaCompressor
LZMA compressor (maximum compression, slowest) . . . 300
filevault::core::ModePreset
Mode presets for different user levels . . . 302
filevault::utils::Password
Password utilities for secure input and strength analysis . . . 305
filevault::core::PasswordAnalysis
Password strength analysis result . . . 311
filevault::core::PBKDF2Params
KDF parameters for PBKDF2 . . . 312
filevault::algorithms::classical::Playfair
Playfair Cipher - Digraph substitution . . . 313
filevault::cli::PQCBenchmarkResult . . . 320
filevault::algorithms::pqc::PQKeyPair
Key pair for post-quantum algorithms . . . 321
filevault::utils::ProgressBar
Modern progress bar wrapper . . . 321
filevault::core::Result< T >
Generic result type . . . 323
filevault::core::Result< void >
Specialization for void . . . 325
filevault::algorithms::asymmetric::RSA
RSA asymmetric encryption algorithm . . . 326
filevault::algorithms::asymmetric::RSAKeyPair
RSA key pair . . . 332
filevault::core::ScryptParams
KDF parameters for Scrypt . . . 333
filevault::algorithms::symmetric::Serpent_GCM
Serpent-256-GCM implementation . . . 334
filevault::cli::SignatureBenchmarkResult . . . 339
filevault::cli::commands::SignCommand
Command to create digital signature for a file . . . 340
filevault::algorithms::symmetric::SM4_GCM
SM4-GCM AEAD encryption . . . 343
filevault::cli::commands::StegoCommand
Steganography command for embedding/extracting data in images . . . 349

filevault::core::StreamingConfig
Configuration for streaming encryption 354
filevault::core::StreamingCrypto
Streaming encryption/decryption for large files 355
filevault::core::StreamingResult
Result of streaming operation 360
filevault::algorithms::classical::SubstitutionCipher
Simple substitution cipher 360
filevault::utils::TableBuilder
Quick table builders 366
filevault::utils::TableFormatter
Beautiful table formatter 367
filevault::algorithms::symmetric::TripleDES
Triple-DES (3DES/TDEA) encryption 371
filevault::algorithms::symmetric::Twofish_GCM
Twofish-256-GCM implementation 375
filevault::cli::commands::VerifyCommand
Command to verify digital signature of a file 382
filevault::algorithms::classical::Vigenere
Vigenre Cipher - Polyalphabetic substitution 385
filevault::compression::ZlibCompressor
ZLIB compressor (fast, good compression) 391

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

include/filevault/algorithms/asymmetric/ecc.hpp
Elliptic Curve Cryptography (ECC) implementation . . . 395
include/filevault/algorithms/asymmetric/rsa.hpp
RSA asymmetric encryption . . . 398
include/filevault/algorithms/classical/caesar.hpp . . . 400
include/filevault/algorithms/classical/classical_ciphers.hpp . . . 401
include/filevault/algorithms/classical/hill.hpp . . . 403
include/filevault/algorithms/classical/playfair.hpp . . . 405
include/filevault/algorithms/classical/substitution.hpp . . . 407
include/filevault/algorithms/classical/vigenere.hpp . . . 409
include/filevault/algorithms/pqc/post_quantum.hpp . . . 411
include/filevault/algorithms/symmetric/aes_cbc.hpp
AES-CBC encryption implementation . . . 414
include/filevault/algorithms/symmetric/aes_cfb.hpp
AES-CFB (Cipher Feedback) mode encryption . . . 416
include/filevault/algorithms/symmetric/aes_ctr.hpp
AES-CTR (Counter mode) encryption implementation . . . 418
include/filevault/algorithms/symmetric/aes_ecb.hpp
AES-ECB (Electronic Codebook) mode encryption . . . 420
include/filevault/algorithms/symmetric/aes_gcm.hpp . . . 422
include/filevault/algorithms/symmetric/aes_ofb.hpp
AES-OFB (Output Feedback) mode encryption . . . 424
include/filevault/algorithms/symmetric/aes_xts.hpp
AES-XTS (XEX-based Tweaked-codebook mode with ciphertext Stealing) . . . 426
include/filevault/algorithms/symmetric/aria_gcm.hpp
ARIA-GCM AEAD encryption algorithm . . . 428
include/filevault/algorithms/symmetric/camellia_gcm.hpp
Camellia-GCM AEAD encryption algorithm . . . 430
include/filevault/algorithms/symmetric/chacha20_poly1305.hpp . . . 432
include/filevault/algorithms/symmetric/serpent_gcm.hpp . . . 434
include/filevault/algorithms/symmetric/sm4_gcm.hpp
SM4-GCM AEAD encryption algorithm . . . 436
include/filevault/algorithms/symmetric/triple_des.hpp
Triple-DES (3DES) encryption implementation . . . 438
include/filevault/algorithms/symmetric/twofish_gcm.hpp
Twofish-256-GCM encryption algorithm . . . 440
include/filevault/archive/archive_format.hpp . . . 442
include/filevault/cli/app.hpp . . . 444
include/filevault/cli/command.hpp . . . 446
include/filevault/cli/commands/archive_cmd.hpp . . . 447

include/filevault/cli/commands/benchmark_cmd.hpp . . . 449
include/filevault/cli/commands/compress_cmd.hpp . . . 451
include/filevault/cli/commands/config_cmd.hpp . . . 453
include/filevault/cli/commands/decompress_cmd.hpp . . . 454
include/filevault/cli/commands/decrypt_cmd.hpp . . . 456
include/filevault/cli/commands/dump_cmd.hpp . . . 457
include/filevault/cli/commands/encrypt_cmd.hpp . . . 459
include/filevault/cli/commands/hash_cmd.hpp . . . 460
include/filevault/cli/commands/info_cmd.hpp . . . 462
include/filevault/cli/commands/keygen_cmd.hpp
Key generation command for asymmetric encryption . . . 464
include/filevault/cli/commands/keyinfo_cmd.hpp . . . 466
include/filevault/cli/commands/list_cmd.hpp . . . 467
include/filevault/cli/commands/sign_cmd.hpp . . . 468
include/filevault/cli/commands/stego_cmd.hpp . . . 470
include/filevault/cli/commands/verify_cmd.hpp . . . 472
include/filevault/compression/compressor.hpp . . . 473
include/filevault/core/crypto_algorithm.hpp . . . 475
include/filevault/core/crypto_engine.hpp . . . 477
include/filevault/core/file_format.hpp . . . 478
include/filevault/core/modes.hpp . . . 482
include/filevault/core/result.hpp . . . 484
include/filevault/core/streaming.hpp . . . 486
include/filevault/core/types.hpp . . . 488
include/filevault/format/file_header.hpp . . . 493
include/filevault/steganography/lsb.hpp . . . 495
include/filevault/utils/config.hpp . . . 497
include/filevault/utils/console.hpp . . . 499
include/filevault/utils/crypto_utils.hpp . . . 500
include/filevault/utils/file_io.hpp . . . 501
include/filevault/utils/password.hpp . . . 502
include/filevault/utils/progress.hpp . . . 503
include/filevault/utils/table_formatter.hpp . . . 505
src/main.cpp . . . 539
src/algorithms/asymmetric/ecc.cpp
Elliptic Curve Cryptography implementation . . . 506
src/algorithms/asymmetric/rsa.cpp
RSA asymmetric encryption implementation . . . 507
src/algorithms/classical/caesar.cpp . . . 508
src/algorithms/classical/classical_ciphers.cpp . . . 509
src/algorithms/classical/hill.cpp . . . 509
src/algorithms/classical/playfair.cpp . . . 510
src/algorithms/classical/substitution.cpp . . . 511
src/algorithms/classical/vigenere.cpp . . . 512
src/algorithms/pqc/post_quantum.cpp
Post-Quantum Cryptography implementations using Botan 3.x . . . 513
src/algorithms/symmetric/aes_cbc.cpp
AES-CBC encryption implementation . . . 514
src/algorithms/symmetric/aes_cfb.cpp
AES-CFB encryption implementation . . . 515
src/algorithms/symmetric/aes_ctr.cpp
AES-CTR encryption implementation . . . 515
src/algorithms/symmetric/aes_ecb.cpp
AES-ECB encryption implementation . . . 516
src/algorithms/symmetric/aes_gcm.cpp . . . 517
src/algorithms/symmetric/aes_ofb.cpp
AES-OFB encryption implementation . . . 517

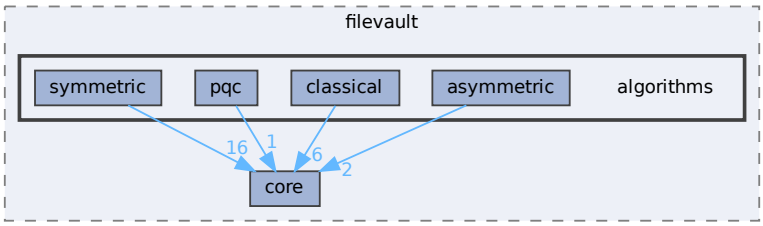
src/algorithms/symmetric/aes_xts.cpp
AES-XTS encryption implementation for disk encryption . . . 518
src/algorithms/symmetric/aria_gcm.cpp
Implementation of ARIA-GCM AEAD encryption . . . 519
src/algorithms/symmetric/camellia_gcm.cpp
Implementation of Camellia-GCM AEAD encryption . . . 520
src/algorithms/symmetric/chacha20_poly1305.cpp . . . 521
src/algorithms/symmetric/serpent_gcm.cpp . . . 522
src/algorithms/symmetric/sm4_gcm.cpp
Implementation of SM4-GCM AEAD encryption . . . 522
src/algorithms/symmetric/triple_des.cpp
Triple-DES (3DES) encryption implementation . . . 523
src/algorithms/symmetric/twofish_gcm.cpp
Twofish-GCM encryption implementation . . . 524
src/archive/archive_format.cpp . . . 524
src/cli/app.cpp . . . 525
src/cli/commands/archive_cmd.cpp . . . 526
src/cli/commands/benchmark_cmd.cpp
Enhanced benchmark command with all algorithms and tabulate tables . . . 526
src/cli/commands/compress_cmd.cpp . . . 527
src/cli/commands/config_cmd.cpp . . . 527
src/cli/commands/decompress_cmd.cpp . . . 528
src/cli/commands/decrypt_cmd.cpp . . . 528
src/cli/commands/dump_cmd.cpp . . . 529
src/cli/commands/encrypt_cmd.cpp . . . 529
src/cli/commands/hash_cmd.cpp . . . 530
src/cli/commands/info_cmd.cpp . . . 531
src/cli/commands/keygen_cmd.cpp
Key generation command implementation . . . 531
src/cli/commands/keyinfo_cmd.cpp . . . 532
src/cli/commands/list_cmd.cpp . . . 532
src/cli/commands/sign_cmd.cpp . . . 533
src/cli/commands/stego_cmd.cpp . . . 533
src/cli/commands/verify_cmd.cpp . . . 534
src/compression/compressor.cpp . . . 535
src/core/crypto_engine.cpp . . . 535
src/core/modes.cpp . . . 536
src/core/streaming.cpp
Streaming encryption implementation for large files . . . 537
src/core/types.cpp . . . 538
src/format/file_format.cpp . . . 538
src/format/file_header.cpp . . . 539
src/steganography/lsb.cpp . . . 540
src/utils/config.cpp . . . 541
src/utils/console.cpp . . . 542
src/utils/crypto_utils.cpp . . . 542
src/utils/file_io.cpp . . . 543
src/utils/password.cpp . . . 544
src/utils/progress.cpp . . . 544
src/utils/table_formatter.cpp . . . 545

Chapter 7

Directory Documentation

7.1 include/filevault/algorithms Directory Reference

Directory dependency graph for algorithms:

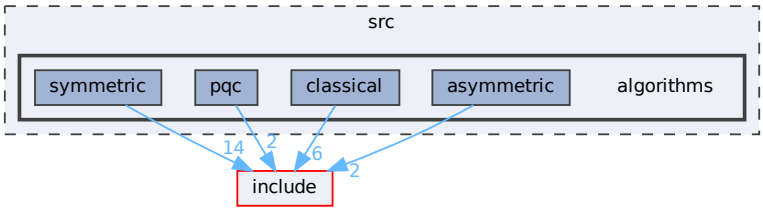


Directories

- directory asymmetric
- directory classical
- directory pqc
- directory symmetric

7.2 src/algorithms Directory Reference

Directory dependency graph for algorithms:

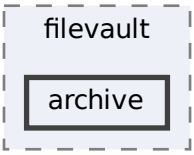


Directories

- directory asymmetric
- directory classical
- directory pqc
- directory symmetric

7.3 include/filevault/archive Directory Reference

Directory dependency graph for archive:

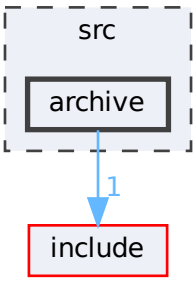


Files

- file archive_format.hpp

7.4 src/archive Directory Reference

Directory dependency graph for archive:

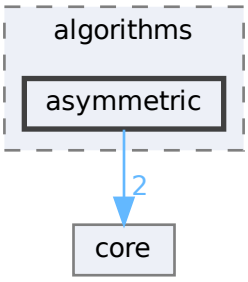


Files

- file archive_format.cpp

7.5 include/filevault/algorithms/asymmetric Directory Reference

Directory dependency graph for asymmetric:



Files

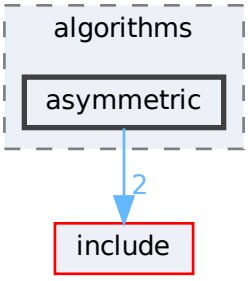
- file ecc.hpp

  Elliptic Curve Cryptography (ECC) implementation.
- file rsa.hpp

  RSA asymmetric encryption.

7.6 src/algorithms/asymmetric Directory Reference

Directory dependency graph for asymmetric:



Files

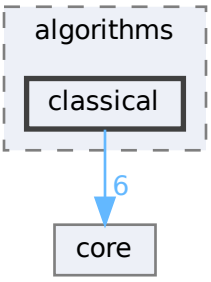
- file ecc.cpp

  Elliptic Curve Cryptography implementation.
- file rsa.cpp

  RSA asymmetric encryption implementation.

7.7 include/filevault/algorithms/classical Directory Reference

Directory dependency graph for classical:

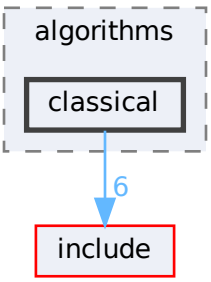


Files

- file caesar.hpp
- file classical_ciphers.hpp
- file hill.hpp
- file playfair.hpp
- file substitution.hpp
- file vigenere.hpp

7.8 src/algorithms/classical Directory Reference

Directory dependency graph for classical:



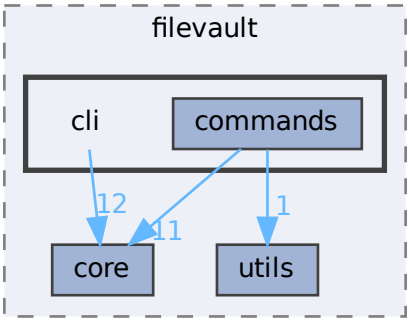
Files

- file caesar.cpp
- file classical_ciphers.cpp
- file hill.cpp

- file playfair.cpp
- file substitution.cpp
- file vigenere.cpp

7.9 include/filevault/cli Directory Reference

Directory dependency graph for cli:



Directories

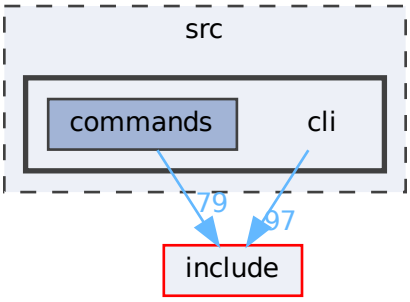
- directory commands

Files

- file app.hpp
- file command.hpp

7.10 src/cli Directory Reference

Directory dependency graph for cli:



Directories

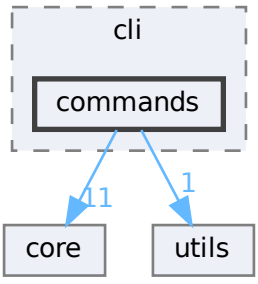
- directory commands

Files

- file app.cpp

7.11 include/filevault/cli/commands Directory Reference

Directory dependency graph for commands:



Files

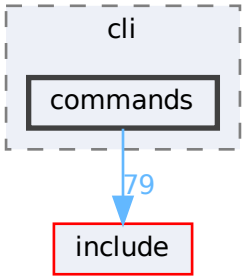
- file archive_cmd.hpp
- file benchmark_cmd.hpp
- file compress_cmd.hpp
- file config_cmd.hpp

- file decompress_cmd.hpp
- file decrypt_cmd.hpp
- file dump_cmd.hpp
- file encrypt_cmd.hpp
- file hash_cmd.hpp
- file info_cmd.hpp
- file keygen_cmd.hpp

  Key generation command for asymmetric encryption.
- file keyinfo_cmd.hpp
- file list_cmd.hpp
- file sign_cmd.hpp
- file stego_cmd.hpp
- file verify_cmd.hpp

7.12 src/cli/commands Directory Reference

Directory dependency graph for commands:



Files

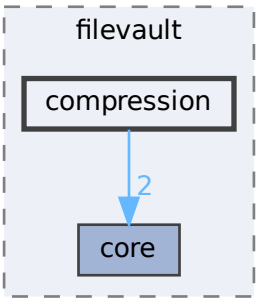
- file archive_cmd.cpp
- file benchmark_cmd.cpp

  Enhanced benchmark command with all algorithms and tabulate tables.
- file compress_cmd.cpp
- file config_cmd.cpp
- file decompress_cmd.cpp
- file decrypt_cmd.cpp
- file dump_cmd.cpp
- file encrypt_cmd.cpp
- file hash_cmd.cpp
- file info_cmd.cpp
- file keygen_cmd.cpp

  Key generation command implementation.
- file keyinfo_cmd.cpp
- file list_cmd.cpp
- file sign_cmd.cpp
- file stego_cmd.cpp
- file verify_cmd.cpp

7.13 include/filevault/compression Directory Reference

Directory dependency graph for compression:

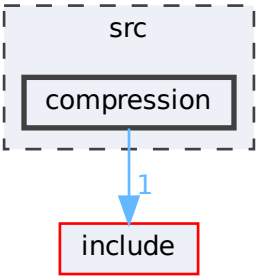


Files

- file compressor.hpp

7.14 src/compression Directory Reference

Directory dependency graph for compression:

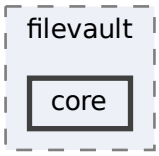


Files

- file compressor.cpp

7.15 include/filevault/core Directory Reference

Directory dependency graph for core:

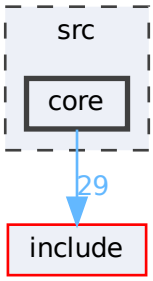


Files

- file crypto_algorithm.hpp
- file crypto_engine.hpp
- file file_format.hpp
- file modes.hpp
- file result.hpp
- file streaming.hpp
- file types.hpp

7.16 src/core Directory Reference

Directory dependency graph for core:



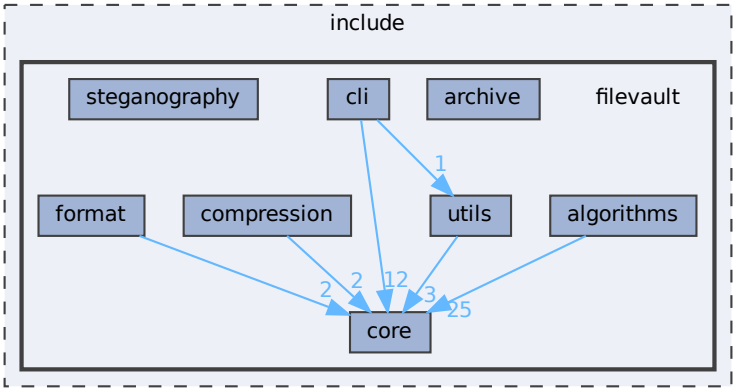
Files

- file crypto_engine.cpp
- file modes.cpp
- file streaming.cpp

  Streaming encryption implementation for large files.
- file types.cpp

7.17 include/filevault Directory Reference

Directory dependency graph for filevault:

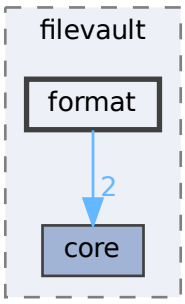


Directories

- directory algorithms
- directory archive
- directory cli
- directory compression
- directory core
- directory format
- directory steganography
- directory utils

7.18 include/filevault/format Directory Reference

Directory dependency graph for format:

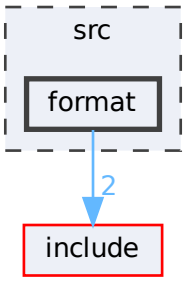


Files

- file file_header.hpp

7.19 src/format Directory Reference

Directory dependency graph for format:

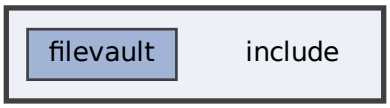


Files

- file file_format.cpp
- file file_header.cpp

7.20 include Directory Reference

Directory dependency graph for include:

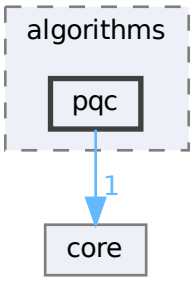


Directories

- directory filevault

7.21 include/filevault/algorithms/pqc Directory Reference

Directory dependency graph for pqc:

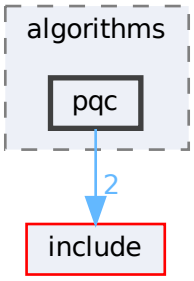


Files

- file post_quantum.hpp

7.22 src/algorithms/pqc Directory Reference

Directory dependency graph for pqc:



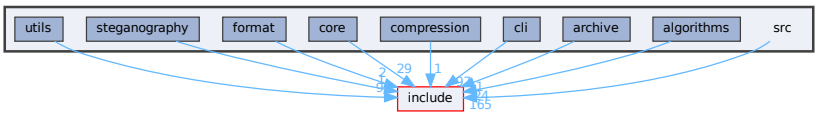
Files

- file post_quantum.cpp

  Post-Quantum Cryptography implementations using Botan 3.x.

7.23 src Directory Reference

Directory dependency graph for src:



Directories

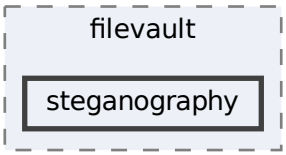
- directory algorithms
- directory archive
- directory cli
- directory compression
- directory core
- directory format
- directory steganography
- directory utils

Files

- file main.cpp

7.24 include/filevault/steganography Directory Reference

Directory dependency graph for steganography:

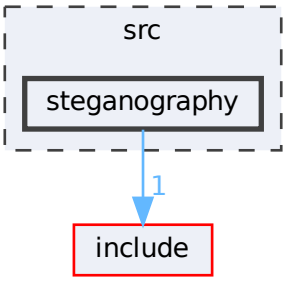


Files

- file lsb.hpp

7.25 src/steganography Directory Reference

Directory dependency graph for steganography:

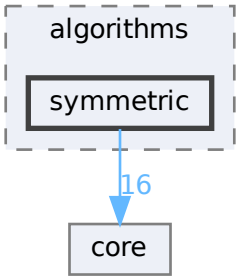


Files

- file lsb.cpp

7.26 include/filevault/algorithms/symmetric Directory Reference

Directory dependency graph for symmetric:



Files

- file aes_cbc.hpp

  AES-CBC encryption implementation.
- file aes_cfb.hpp

  AES-CFB (Cipher Feedback) mode encryption.
- file aes_ctr.hpp

  AES-CTR (Counter mode) encryption implementation.
- file aes_ecb.hpp

AES-ECB (Electronic Codebook) mode encryption.

- file aes_gcm.hpp
- file aes_ofb.hpp

  AES-OFB (Output Feedback) mode encryption.
- file aes_xts.hpp

  AES-XTS (XEX-based Tweaked-codebook mode with ciphertext Stealing).
- file aria_gcm.hpp

  ARIA-GCM AEAD encryption algorithm.
- file camellia_gcm.hpp

  Camellia-GCM AEAD encryption algorithm.
- file chacha20_poly1305.hpp
- file serpent_gcm.hpp
- file sm4_gcm.hpp

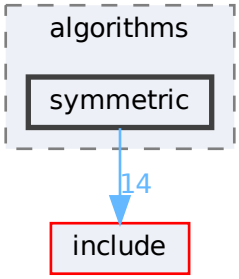
  SM4-GCM AEAD encryption algorithm.
- file triple_des.hpp

  Triple-DES (3DES) encryption implementation.
- file twofish_gcm.hpp

  Twofish-256-GCM encryption algorithm.

7.27 src/algorithms/symmetric Directory Reference

Directory dependency graph for symmetric:



Files

- file aes_cbc.cpp

  AES-CBC encryption implementation.
- file aes_cfb.cpp

  AES-CFB encryption implementation.
- file aes_ctr.cpp

  AES-CTR encryption implementation.
- file aes_ecb.cpp

  AES-ECB encryption implementation.
- file aes_gcm.cpp
- file aes_ofb.cpp

  AES-OFB encryption implementation.

- file aes_xts.cpp

  AES-XTS encryption implementation for disk encryption.
- file aria_gcm.cpp

  Implementation of ARIA-GCM AEAD encryption.
- file camellia_gcm.cpp

  Implementation of Camellia-GCM AEAD encryption.
- file chacha20_poly1305.cpp
- file serpent_gcm.cpp
- file sm4_gcm.cpp

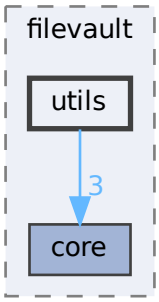
  Implementation of SM4-GCM AEAD encryption.
- file triple_des.cpp

  Triple-DES (3DES) encryption implementation.
- file twofish_gcm.cpp

  Twofish-GCM encryption implementation.

7.28 include/filevault/utils Directory Reference

Directory dependency graph for utils:

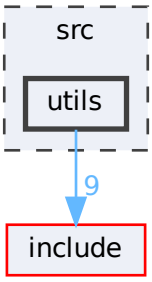


Files

- file config.hpp
- file console.hpp
- file crypto_utils.hpp
- file file_io.hpp
- file password.hpp
- file progress.hpp
- file table_formatter.hpp

7.29 src/utils Directory Reference

Directory dependency graph for utils:



Files

- file config.cpp
- file console.cpp
- file crypto_utils.cpp
- file file_io.cpp
- file password.cpp
- file progress.cpp
- file table_formatter.cpp

Chapter 8

Namespace Documentation

8.1 filevault Namespace Reference

Namespaces

- namespace algorithms
- namespace archive
- namespace cli
- namespace compression
- namespace core
- namespace format
- namespace steganography
- namespace utils

8.2 filevault::algorithms Namespace Reference

Namespaces

- namespace asymmetric
- namespace classical
- namespace pqc
- namespace symmetric

8.3 filevault::algorithms::asymmetric Namespace Reference

Classes

- struct ECCKeyPair

  ECC key pair (ECDH or ECDSA).
- struct ECDHResult

  ECDH Key Exchange result.
- struct ECDSASignResult

  ECDSA signature result.
- class ECDH

  Elliptic Curve Diffie-Hellman (ECDH) key exchange.
- class ECDSA

  Elliptic Curve Digital Signature Algorithm (ECDSA).
- class ECCHybrid

  Hybrid encryption using ECDH + AES-GCM.
- struct RSAKeyPair

  RSA key pair.

- class RSA

  RSA asymmetric encryption algorithm.

Enumerations

- enum class ECCurve { SECP256R1 , SECP384R1 , SECP521R1 , X25519 }

  Supported elliptic curves.

Functions

- static std::string get_botan_curve_name (ECCurve curve)
- static size_t get_curve_key_size (ECCurve curve)

8.3.1 Enumeration Type Documentation

8.3.1.1 ECCurve

```
enum class filevault::algorithms::asymmetric::ECCurve  [strong]
```

Supported elliptic curves.

Enumerator

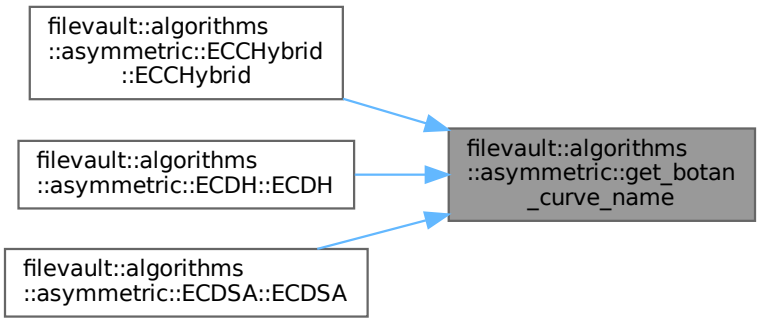
SECP256R1	
SECP384R1	
SECP521R1	
X25519	

8.3.2 Function Documentation

8.3.2.1 get_botan_curve_name()

```
std::string filevault::algorithms::asymmetric::get_botan_curve_name (
            ECCurve curve)  [static]
```

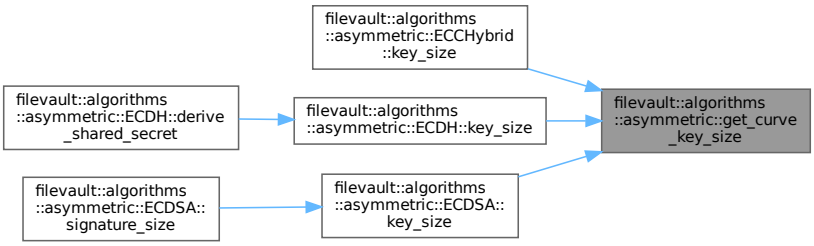
Here is the caller graph for this function:



8.3.2.2 get_curve_key_size()

```
size_t filevault::algorithms::asymmetric::get_curve_key_size (
            ECCurve curve)  [static]
```

Here is the caller graph for this function:



8.4 filevault::algorithms::classical Namespace Reference

Classes

- class Caesar

  Caesar Cipher - Educational only.
- class Vigenere

  Vigenre Cipher - Polyalphabetic substitution.
- class Playfair

  Playfair Cipher - Digraph substitution.
- class HillCipher

  Hill cipher (2x2 matrix).
- class SubstitutionCipher

  Simple substitution cipher.

8.5 filevault::algorithms::pqc Namespace Reference

Classes

- struct PQKeyPair

  Key pair for post-quantum algorithms.
- class Kyber

  Kyber Key Encapsulation Mechanism (KEM).
- class Dilithium

  Dilithium Digital Signature Algorithm.
- class KyberHybrid

  Hybrid encryption combining Kyber with AES-GCM.

Functions

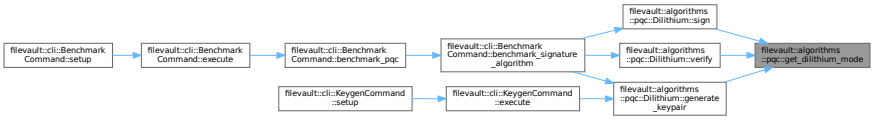
- static Botan::KyberMode get_kyber_mode (Kyber::Variant variant)
- static Botan::DilithiumMode get_dilithium_mode (Dilithium::Variant variant)

8.5.1 Function Documentation

8.5.1.1 get_dilithium_mode()

```
Botan::DilithiumMode filevault::algorithms::pqc::get_dilithium_mode (
            Dilithium::Variant variant)  [static]
```

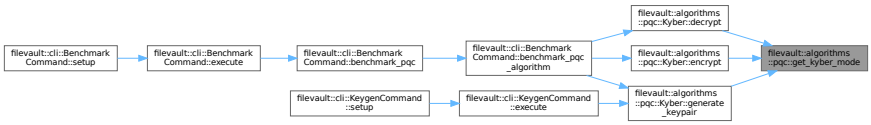
Here is the caller graph for this function:



8.5.1.2 get_kyber_mode()

```
Botan::KyberMode filevault::algorithms::pqc::get_kyber_mode (
            Kyber::Variant variant)  [static]
```

Here is the caller graph for this function:



8.6 filevault::algorithms::symmetric Namespace Reference

Classes

- class AES_CBC

  AES-CBC encryption (legacy mode, not authenticated).
- class AES_CFB

  AES-CFB encryption algorithm.
- class AES_CTR

  AES-CTR encryption (stream cipher mode, not authenticated).
- class AES_ECB

  AES-ECB encryption algorithm.
- class AES_GCM

  AES-GCM AEAD encryption.
- class AES_OFB

  AES-OFB encryption algorithm.
- class AES_XTS

  AES-XTS encryption algorithm.
- class ARIA_GCM

  ARIA-GCM AEAD encryption.
- class Camellia_GCM

  Camellia-GCM AEAD encryption.
- class ChaCha20Poly1305

  ChaCha20-Poly1305 AEAD encryption.
- class Serpent_GCM

Serpent-256-GCM implementation.

- class SM4_GCM

  SM4-GCM AEAD encryption.

- class TripleDES

  Triple-DES (3DES/TDEA) encryption.

- class Twofish_GCM

  Twofish-256-GCM implementation.

Functions

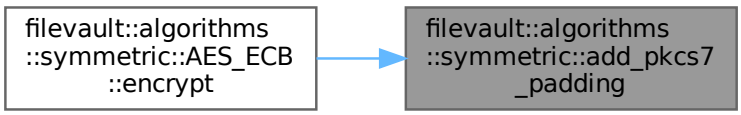
- static std::vector< uint8_t > add_pkcs7_padding (std::span< const uint8_t > data, size_t block_size)
- static std::vector< uint8_t > remove_pkcs7_padding (std::span< const uint8_t > data)

8.6.1 Function Documentation

8.6.1.1 add_pkcs7_padding()

```
std::vector< uint8_t > filevault::algorithms::symmetric::add_pkcs7_padding (
            std::span< const uint8_t > data,
            size_t block_size)  [static]
```

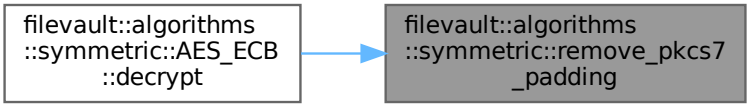
Here is the caller graph for this function:



8.6.1.2 remove_pkcs7_padding()

```
std::vector< uint8_t > filevault::algorithms::symmetric::remove_pkcs7_padding (
            std::span< const uint8_t > data)  [static]
```

Here is the caller graph for this function:



8.7 filevault::archive Namespace Reference

Classes

- struct FileEntry

Archive file entry metadata.

- class ArchiveFormat

  Simple archive format handler.

8.8 filevault::cli Namespace Reference

Namespaces

- namespace commands

Classes

- class Application

  Main CLI application.
- class ICommand

  Base interface for CLI commands.
- struct BenchmarkResult
- struct AsymmetricBenchmarkResult
- struct PQCBenchmarkResult
- struct SignatureBenchmarkResult
- class BenchmarkCommand
- class CompressCommand

  Compress command - standalone compression/decompression.
- class ConfigCommand

  Config command - manage FileVault configuration.
- class DecompressCommand

  Decompress command - alias for compress -d.
- class DecryptCommand
- class EncryptCommand

  Encrypt command.
- class HashCommand

  Hash command - Calculate cryptographic hashes.
- class InfoCommand

  Info command - display encrypted file metadata.
- class KeygenCommand

  Command to generate key pairs for asymmetric encryption.
- class ListCommand

8.9 filevault::cli::commands Namespace Reference

Classes

- class ArchiveCommand

  Archive command - compress and encrypt multiple files.
- class DumpCommand

  Command to dump file content in various formats (hex, binary, base64).
- class KeyInfoCommand

  Command to display information about cryptographic keys.
- class SignCommand

  Command to create digital signature for a file.
- class StegoCommand

  Steganography command for embedding/extracting data in images.
- class VerifyCommand

  Command to verify digital signature of a file.

8.10 filevault::compression Namespace Reference

Classes

- struct CompressionResult

  Compression result.
- class ICompressor

  Interface for compression algorithms.
- class CompressionService

  Compression service - factory for compressors.
- class ZlibCompressor

  ZLIB compressor (fast, good compression).
- class Bzip2Compressor

  BZIP2 compressor (better ratio, slower).
- class LzmaCompressor

  LZMA compressor (maximum compression, slowest).

8.11 filevault::core Namespace Reference

Namespaces

- namespace presets

Classes

- class ICryptoAlgorithm

  Interface for cryptographic algorithms.
- class CryptoEngine

  Main cryptographic engine Manages algorithms and provides key derivation.
- struct Argon2Params

  KDF parameters for Argon2.
- struct PBKDF2Params

  KDF parameters for PBKDF2.
- struct ScryptParams

  KDF parameters for Scrypt.
- struct FileHeader

  File format header.
- class FileFormatHandler

  File format handler.
- struct ModePreset

  Mode presets for different user levels.
- struct CryptoResult

  Result of cryptographic operations.
- struct Result

  Generic result type.
- struct Result< void >

  Specialization for void.
- struct ChunkInfo

  Chunk information for streaming encryption.
- struct StreamingConfig

  Configuration for streaming encryption.
- struct StreamingResult

  Result of streaming operation.

- class StreamingCrypto

  Streaming encryption/decryption for large files.
- struct EncryptionConfig

  Configuration for encryption operations.
- struct HashConfig

  Configuration for hashing operations.
- struct PasswordAnalysis

  Password strength analysis result.

Typedefs

- using StreamProgressCallback = std::function<bool(const ChunkInfo& info)>

  Progress callback for streaming operations.

Enumerations

- enum class AlgorithmID : uint8_t {
  UNKNOWN = 0x00 , AES_128_GCM = 0x01 , AES_192_GCM = 0x02 , AES_256_GCM = 0x03 ,
  CHACHA20_POLY1305 = 0x04 , SERPENT_256_GCM = 0x05 , TWOFISH_128_GCM = 0x06 ,
  TWOFISH_192_GCM = 0x07 ,
  TWOFISH_256_GCM = 0x08 , CAMELLIA_128_GCM = 0x09 , CAMELLIA_192_GCM = 0x0A ,
  CAMELLIA_256_GCM = 0x0B ,
  ARIA_128_GCM = 0x0C , ARIA_192_GCM = 0x0D , ARIA_256_GCM = 0x0E , SM4_GCM = 0x0F ,
  CAESAR = 0x10 , VIGENERE = 0x11 , PLAYFAIR = 0x12 , SUBSTITUTION = 0x13 ,
  HILL = 0x14 , AES_128_CBC = 0x20 , AES_192_CBC = 0x21 , AES_256_CBC = 0x22 ,
  AES_128_CTR = 0x23 , AES_192_CTR = 0x24 , AES_256_CTR = 0x25 , AES_128_CFB = 0x26 ,
  AES_192_CFB = 0x27 , AES_256_CFB = 0x28 , AES_128_OFB = 0x29 , AES_192_OFB = 0x2A ,
  AES_256_OFB = 0x2B , AES_128_ECB = 0x2C , AES_192_ECB = 0x2D , AES_256_ECB = 0x2E ,
  AES_128_XTS = 0x2F , AES_256_XTS = 0x30 , TRIPLE_DES_CBC = 0x40 , RSA_2048 = 0x50 ,
  RSA_3072 = 0x51 , RSA_4096 = 0x52 , ECC_P256 = 0x60 , ECC_P384 = 0x61 ,
  ECC_P521 = 0x62 }

  Algorithm identifiers.
- enum class KDFID : uint8_t {
  NONE = 0x00 , ARGON2ID = 0x01 , ARGON2I = 0x02 , PBKDF2_SHA256 = 0x03 ,
  PBKDF2_SHA512 = 0x04 , SCRYPT = 0x05 }

  KDF identifiers.
- enum class CompressionID : uint8_t { NONE = 0x00 , ZLIB = 0x01 , BZIP2 = 0x02 , LZMA = 0x03 }

  Compression identifiers.
- enum class AlgorithmType {
  AES_128_GCM , AES_192_GCM , AES_256_GCM , CHACHA20_POLY1305 ,
  SERPENT_256_GCM , TWOFISH_128_GCM , TWOFISH_192_GCM , TWOFISH_256_GCM ,
  CAMELLIA_128_GCM , CAMELLIA_192_GCM , CAMELLIA_256_GCM , ARIA_128_GCM ,
  ARIA_192_GCM , ARIA_256_GCM , SM4_GCM , AES_128_CBC ,
  AES_192_CBC , AES_256_CBC , AES_128_CTR , AES_192_CTR ,
  AES_256_CTR , AES_128_CFB , AES_192_CFB , AES_256_CFB ,
  AES_128_OFB , AES_192_OFB , AES_256_OFB , AES_128_ECB ,
  AES_192_ECB , AES_256_ECB , AES_128_XTS , AES_256_XTS ,
  TRIPLE_DES_CBC , RSA_2048 , RSA_3072 , RSA_4096 ,
  ECC_P256 , ECC_P384 , ECC_P521 , CAESAR ,
  VIGENERE , PLAYFAIR , SUBSTITUTION , HILL ,
  KYBER_512 , KYBER_768 , KYBER_1024 , DILITHIUM_2 ,
  DILITHIUM_3 , DILITHIUM_5 , KYBER_512_HYBRID , KYBER_768_HYBRID ,
  KYBER_1024_HYBRID }

  Enumeration of encryption algorithm types.

- enum class HashType { MD5 , SHA1 , SHA224 , SHA256 , SHA384 , SHA512 , SHA512_256 , SHA3_224 , SHA3_256 , SHA3_384 , SHA3_512 , BLAKE2B_256 , BLAKE2B_384 , BLAKE2B_512 , BLAKE2S_256 }

  Enumeration of hash algorithm types.
- enum class CompressionType { NONE , ZLIB , BZIP2 , LZMA }

  Compression algorithm types.
- enum class UserMode { STUDENT , PROFESSIONAL , ADVANCED }

  User mode/profile for algorithm selection.
- enum class KDFType { ARGON2ID , ARGON2I , PBKDF2_SHA256 , PBKDF2_SHA512 , SCRYPT }

  Enumeration of Key Derivation Function types.
- enum class SecurityLevel { WEAK , MEDIUM , STRONG , PARANOID }

  Security level determining algorithm parameters.
- enum class PasswordStrength { VERY_WEAK , WEAK , FAIR , STRONG , VERY_STRONG }

  Password strength levels.

Variables

- static constexpr uint8_t STREAM_MAGIC [4] = {'F', 'V', 'S', 'T'}
- static constexpr uint8_t STREAM_VERSION = 1
- constexpr uint8_t FILE_FORMAT_MAGIC [8] = {'F', 'V', 'A', 'U', 'L', 'T', '0', '1'}

  FileVault file format version 1.0.
- constexpr uint8_t FILE_FORMAT_VERSION_MAJOR = 1
- constexpr uint8_t FILE_FORMAT_VERSION_MINOR = 0

8.11.1 Typedef Documentation

8.11.1.1 StreamProgressCallback

```
using filevault::core::StreamProgressCallback = std::function<bool(const ChunkInfo& info)>
```

Progress callback for streaming operations.

Parameters

info	Current chunk information

Returns

false to cancel operation, true to continue

8.11.2 Enumeration Type Documentation

8.11.2.1 AlgorithmID

```
enum class filevault::core::AlgorithmID : uint8_t [strong]
```

Algorithm identifiers.

Enumerator

UNKNOWN	
AES_128_GCM	
AES_192_GCM	
AES_256_GCM	
CHACHA20_POLY1305	
SERPENT_256_GCM	
TWOFISH_128_GCM	
TWOFISH_192_GCM	
TWOFISH_256_GCM	
CAMELLIA_128_GCM	
CAMELLIA_192_GCM	
CAMELLIA_256_GCM	
ARIA_128_GCM	
ARIA_192_GCM	
ARIA_256_GCM	
SM4_GCM	
CAESAR	
VIGENERE	
PLAYFAIR	
SUBSTITUTION	
HILL	
AES_128_CBC	
AES_192_CBC	
AES_256_CBC	
AES_128_CTR	
AES_192_CTR	
AES_256_CTR	
AES_128_CFB	
AES_192_CFB	
AES_256_CFB	
AES_128_OFB	
AES_192_OFB	
AES_256_OFB	
AES_128_ECB	
AES_192_ECB	
AES_256_ECB	
AES_128_XTS	
AES_256_XTS	
TRIPLE_DES_CBC	
RSA_2048	
RSA_3072	

RSA_4096	
ECC_P256	
ECC_P384	
ECC_P521	

8.11.2.2 AlgorithmType

`enum class filevault::core::AlgorithmType [strong]`

Enumeration of encryption algorithm types.

Enumerator

AES_128_GCM	
AES_192_GCM	
AES_256_GCM	
CHACHA20_POLY1305	
SERPENT_256_GCM	
TWOFISH_128_GCM	
TWOFISH_192_GCM	
TWOFISH_256_GCM	
CAMELLIA_128_GCM	
CAMELLIA_192_GCM	
CAMELLIA_256_GCM	
ARIA_128_GCM	
ARIA_192_GCM	
ARIA_256_GCM	
SM4_GCM	
AES_128_CBC	
AES_192_CBC	
AES_256_CBC	
AES_128_CTR	
AES_192_CTR	
AES_256_CTR	
AES_128_CFB	
AES_192_CFB	
AES_256_CFB	
AES_128_OFB	
AES_192_OFB	
AES_256_OFB	
AES_128_ECB	
AES_192_ECB	
AES_256_ECB	
AES_128_XTS	

AES_256_XTS	
TRIPLE_DES_CBC	
RSA_2048	
RSA_3072	
RSA_4096	
ECC_P256	
ECC_P384	
ECC_P521	
CAESAR	
VIGENERE	
PLAYFAIR	
SUBSTITUTION	
HILL	
KYBER_512	
KYBER_768	
KYBER_1024	
DILITHIUM_2	
DILITHIUM_3	
DILITHIUM_5	
KYBER_512_HYBRID	
KYBER_768_HYBRID	
KYBER_1024_HYBRID	

8.11.2.3 CompressionID

`enum class filevault::core::CompressionID : uint8_t [strong]`
Compression identifiers.

Enumerator

NONE	
ZLIB	
BZIP2	
LZMA	

8.11.2.4 CompressionType

`enum class filevault::core::CompressionType [strong]`
Compression algorithm types.

Enumerator

NONE	
ZLIB	
BZIP2	
LZMA	

8.11.2.5 HashType

`enum class filevault::core::HashType [strong]`
Enumeration of hash algorithm types.

Enumerator

MD5	
SHA1	
SHA224	
SHA256	
SHA384	
SHA512	
SHA512_256	
SHA3_224	
SHA3_256	
SHA3_384	
SHA3_512	
BLAKE2B_256	
BLAKE2B_384	
BLAKE2B_512	
BLAKE2S_256	

8.11.2.6 KDFID

`enum class filevault::core::KDFID : uint8_t [strong]`
KDF identifiers.

Enumerator

NONE	
ARGON2ID	
ARGON2I	
PBKDF2_SHA256	
PBKDF2_SHA512	
SCRYPT	

8.11.2.7 KDFType

`enum class filevault::core::KDFType [strong]`
Enumeration of Key Derivation Function types.

Enumerator

ARGON2ID	
ARGON2I	
PBKDF2_SHA256	
PBKDF2_SHA512	
SCRYPT	

8.11.2.8 PasswordStrength

`enum class filevault::core::PasswordStrength [strong]`
Password strength levels.

Enumerator

VERY_WEAK	
WEAK	
FAIR	
STRONG	
VERY_STRONG	

8.11.2.9 SecurityLevel

`enum class filevault::core::SecurityLevel [strong]`
Security level determining algorithm parameters.

Enumerator

WEAK	
MEDIUM	
STRONG	
PARANOID	

8.11.2.10 UserMode

`enum class filevault::core::UserMode [strong]`
User mode/profile for algorithm selection.

Enumerator

STUDENT	
PROFESSIONAL	
ADVANCED	

8.11.3 Variable Documentation

8.11.3.1 FILE_FORMAT_MAGIC

```
uint8_t filevault::core::FILE_FORMAT_MAGIC[8] = {'F', 'V', 'A', 'U', 'L', 'T', '0', '1'}  [constexpr]
```

FileVault file format version 1.0.
Format structure: [Magic:8][Version:2][AlgoID:1][KDFID:1][CompID:1][Reserved:3] [Salt:32][KDF_Params↩
:Variable][NonceSize:1][Nonce:Variable][Compressed_Flag:1] [Ciphertext][Auth_Tag:16 (AEAD only)]
Magic: "FVAULT01" (8 bytes) Version: Major.Minor (1 byte each) AlgoID: Algorithm identifier (1 byte) KDFID↩
: KDF identifier (1 byte) CompID: Compression identifier (1 byte) Reserved: Future use (3 bytes) Salt: Random salt for KDF (32 bytes) KDF_Params: Variable-length KDF parameters NonceSize: Size of nonce/IV in bytes (1 byte) Nonce: Random nonce/IV (variable, e.g. 12 for GCM, 16 for CBC/CTR) Compressed_Flag: 0x00=No, 0x01=Yes (1 byte) Ciphertext: Encrypted data Auth_Tag: GCM authentication tag (16 bytes, only for AEAD)

8.11.3.2 FILE_FORMAT_VERSION_MAJOR

```
uint8_t filevault::core::FILE_FORMAT_VERSION_MAJOR = 1  [constexpr]
```

8.11.3.3 FILE_FORMAT_VERSION_MINOR

```
uint8_t filevault::core::FILE_FORMAT_VERSION_MINOR = 0  [constexpr]
```

8.11.3.4 STREAM_MAGIC

```
uint8_t filevault::core::STREAM_MAGIC[4] = {'F', 'V', 'S', 'T'}  [static], [constexpr]
```

8.11.3.5 STREAM_VERSION

```
uint8_t filevault::core::STREAM_VERSION = 1  [static], [constexpr]
```

8.12 filevault::core::presets Namespace Reference

Variables

- const ModePreset BASIC
- const ModePreset STANDARD
- const ModePreset ADVANCED

8.12.1 Variable Documentation

8.12.1.1 ADVANCED

```
const ModePreset filevault::core::presets::ADVANCED
```

Initial value:

```
= {
        .mode = UserMode::ADVANCED,
        .algorithm = AlgorithmType::AES_256_GCM,
        .kdf = KDFType::ARGON2ID,
        .security_level = SecurityLevel::PARANOID,
        .compression = CompressionType::LZMA,
        .compression_level = 9,
        .kdf_iterations = 200000,
        .kdf_memory_kb = 131072,
        .kdf_parallelism = 8
    }
```

8.12.1.2 BASIC

```
const ModePreset filevault::core::presets::BASIC
```

Initial value:

```
= {
        .mode = UserMode::STUDENT,
        .algorithm = AlgorithmType::AES_256_GCM,
```

```
        .kdf = KDFType::PBKDF2_SHA256,
        .security_level = SecurityLevel::MEDIUM,
        .compression = CompressionType::ZLIB,
        .compression_level = 6,
        .kdf_iterations = 100000,
        .kdf_memory_kb = 32768,
        .kdf_parallelism = 2
    }
```

8.12.1.3 STANDARD

```
const ModePreset filevault::core::presets::STANDARD
```

Initial value:

```
= {
        .mode = UserMode::PROFESSIONAL,
        .algorithm = AlgorithmType::AES_256_GCM,
        .kdf = KDFType::ARGON2ID,
        .security_level = SecurityLevel::STRONG,
        .compression = CompressionType::ZLIB,
        .compression_level = 9,
        .kdf_iterations = 100000,
        .kdf_memory_kb = 65536,
        .kdf_parallelism = 4
    }
```

8.13 filevault::format Namespace Reference

Classes

- class FileHeader

  FileVault encrypted file format.

8.14 filevault::steganography Namespace Reference

Classes

- class LSBSteganography

  LSB (Least Significant Bit) Steganography.

8.15 filevault::utils Namespace Reference

Classes

- class Config

  Configuration manager for FileVault.
- class Console

  Console utilities for colored output.
- class CryptoUtils

  Crypto utility functions.
- class FileIO

  File I/O utilities.
- class Password

  Password utilities for secure input and strength analysis.
- class ProgressBar

  Modern progress bar wrapper.
- class BlockProgressBar

  Block-style progress bar.
- class TableFormatter

  Beautiful table formatter.
- class TableBuilder

  Quick table builders.

Functions

- static bool is_terminal ()

Variables

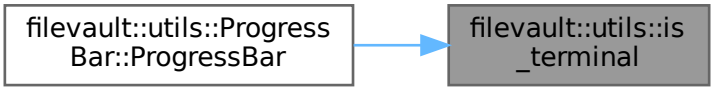
- static std::vector< std::vector< std::string > > mingw_data

8.15.1 Function Documentation

8.15.1.1 is_terminal()

```
bool filevault::utils::is_terminal ()  [static]
```

Here is the caller graph for this function:



8.15.2 Variable Documentation

8.15.2.1 mingw_data

```
std::vector<std::vector<std::string> > filevault::utils::mingw_data  [static]
```

Chapter 9

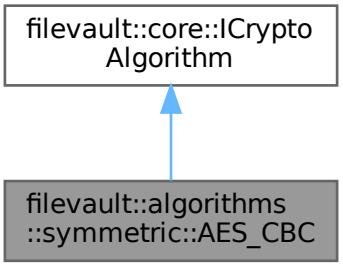
Class Documentation

9.1 filevault::algorithms::symmetric::AES_CBC Class Reference

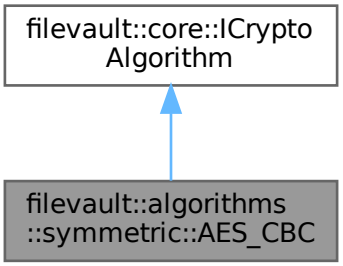
AES-CBC encryption (legacy mode, not authenticated).

`#include <aes_cbc.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::AES_CBC:



Collaboration diagram for filevault::algorithms::symmetric::AES_CBC:



Public Member Functions

- AES_CBC (size_t key_bits=256)
- virtual ~AES_CBC ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t iv_size () const
- size_t block_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.1.1 Detailed Description

AES-CBC encryption (legacy mode, not authenticated).
Supports 128, 192, and 256 bit keys. Uses PKCS7 padding. Requires unique IV for each encryption.

Warning

> This mode does NOT provide authentication! Ciphertext can be modified without detection. Consider using AES-GCM for authenticated encryption.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 AES_CBC()

```
filevault::algorithms::symmetric::AES_CBC::AES_CBC (
            size_t key_bits = 256)  [explicit]
```

9.1.2.2 ~AES_CBC()

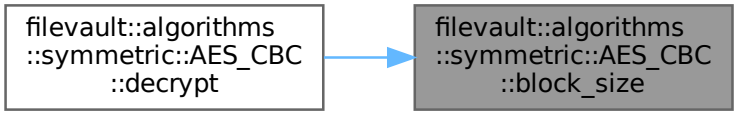
```
virtual filevault::algorithms::symmetric::AES_CBC::~AES_CBC ()  [virtual], [default]
```

9.1.3 Member Function Documentation

9.1.3.1 block_size()

```
size_t filevault::algorithms::symmetric::AES_CBC::block_size () const  [inline]
```

Here is the caller graph for this function:



9.1.3.2 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_CBC::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

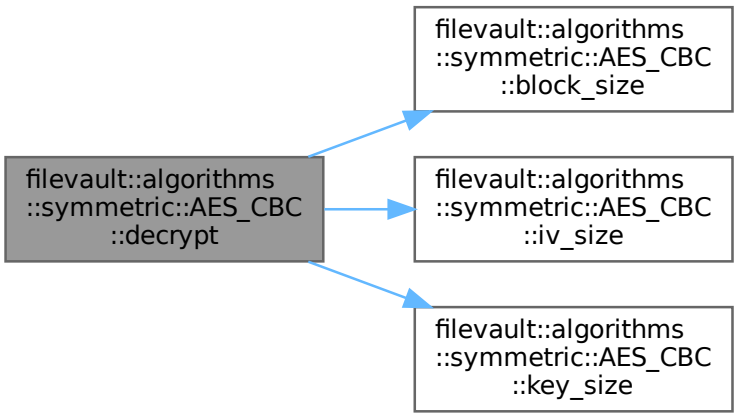
ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

Here is the call graph for this function:



9.1.3.3 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_CBC::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

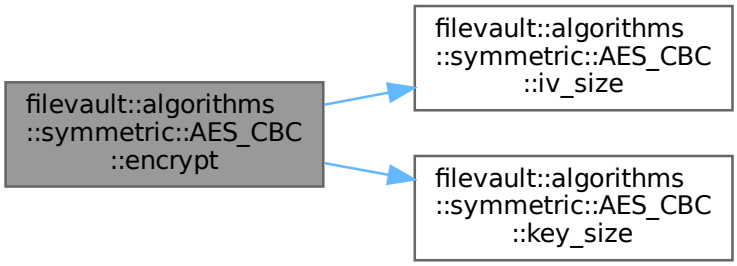
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.1.3.4 is_suitable_for()

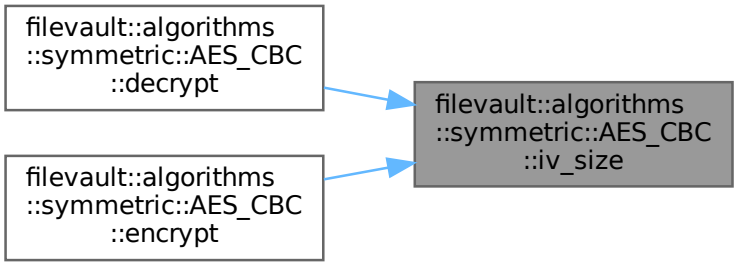
```
bool filevault::algorithms::symmetric::AES_CBC::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.1.3.5 iv_size()

```
size_t filevault::algorithms::symmetric::AES_CBC::iv_size () const  [inline]
```

Here is the caller graph for this function:

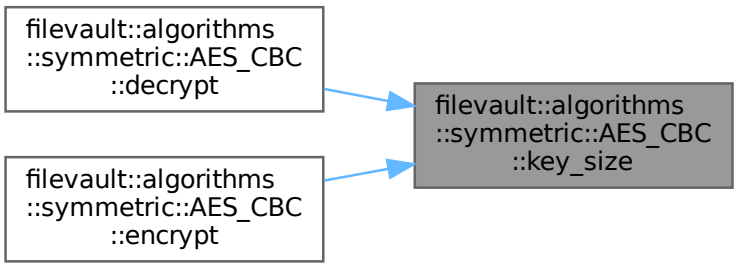


9.1.3.6 key_size()

```
size_t filevault::algorithms::symmetric::AES_CBC::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



9.1.3.7 name()

`std::string filevault::algorithms::symmetric::AES_CBC::name () const [override], [virtual]`
Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.1.3.8 type()

`core::AlgorithmType filevault::algorithms::symmetric::AES_CBC::type () const [override], [virtual]`
Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.1.4 Member Data Documentation

9.1.4.1 botan_name_

`std::string filevault::algorithms::symmetric::AES_CBC::botan_name_ [private]`

9.1.4.2 key_bits_

`size_t filevault::algorithms::symmetric::AES_CBC::key_bits_ [private]`

9.1.4.3 type_

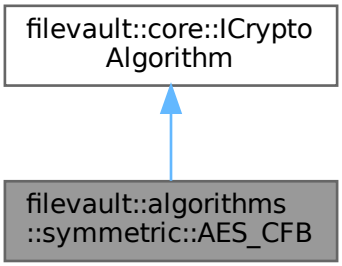
`core::AlgorithmType filevault::algorithms::symmetric::AES_CBC::type_ [private]`
The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/aes_cbc.hpp
- src/algorithms/symmetric/aes_cbc.cpp

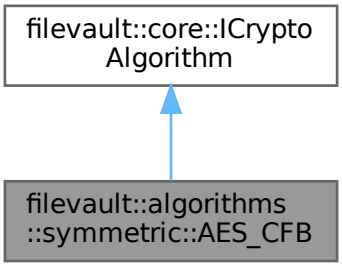
9.2 filevault::algorithms::symmetric::AES_CFB Class Reference

AES-CFB encryption algorithm.
`#include <aes_cfb.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::AES_CFB:



Collaboration diagram for filevault::algorithms::symmetric::AES_CFB:



Public Member Functions

- AES_CFB (size_t key_bits=256)

  Construct AES-CFB with specified key size.
- ~AES_CFB () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t iv_size () const
- size_t block_size () const

- bool requires_padding () const
- bool is_authenticated () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.2.1 Detailed Description

AES-CFB encryption algorithm.
Features:

- Self-synchronizing stream cipher
- No padding required
- Can recover from bit errors (self-healing)
- Encryption and decryption use same operation

9.2.2 Constructor & Destructor Documentation

9.2.2.1 AES_CFB()

```
filevault::algorithms::symmetric::AES_CFB::AES_CFB (
            size_t key_bits = 256)  [explicit]
```

Construct AES-CFB with specified key size.

Parameters

key_bits	Key size in bits (128, 192, or 256)

9.2.2.2 ~AES_CFB()

```
filevault::algorithms::symmetric::AES_CFB::~AES_CFB ()  [override], [default]
```

9.2.3 Member Function Documentation

9.2.3.1 block_size()

```
size_t filevault::algorithms::symmetric::AES_CFB::block_size () const  [inline]
```

9.2.3.2 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_CFB::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

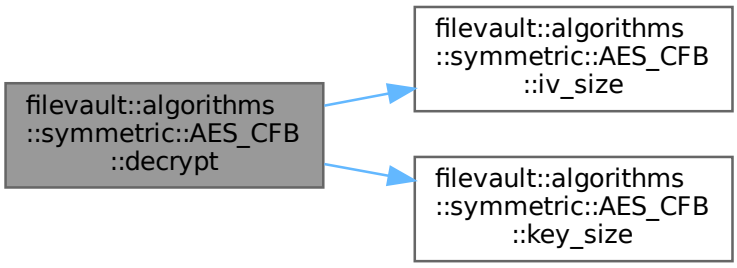
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.2.3.3 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_CFB::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

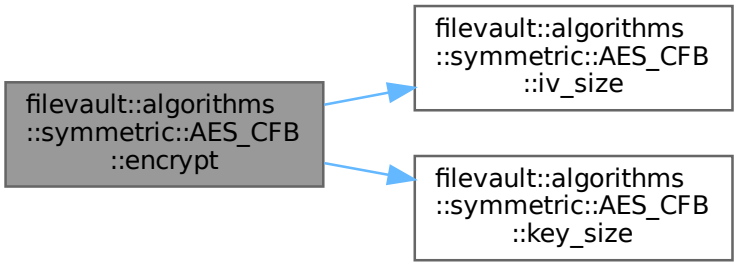
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.2.3.4 is_authenticated()

```
bool filevault::algorithms::symmetric::AES_CFB::is_authenticated () const  [inline]
```

9.2.3.5 is_suitable_for()

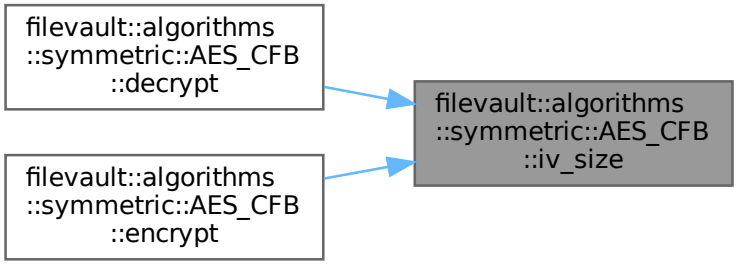
```
bool filevault::algorithms::symmetric::AES_CFB::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.2.3.6 iv_size()

```
size_t filevault::algorithms::symmetric::AES_CFB::iv_size () const  [inline]
```

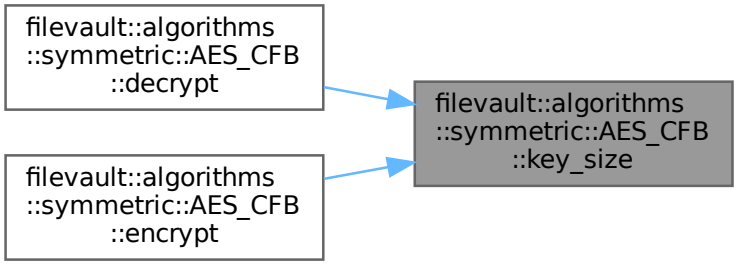
Here is the caller graph for this function:



9.2.3.7 key_size()

```
size_t filevault::algorithms::symmetric::AES_CFB::key_size () const  [inline], [override], [virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.2.3.8 name()

```
std::string filevault::algorithms::symmetric::AES_CFB::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.2.3.9 requires_padding()

```
bool filevault::algorithms::symmetric::AES_CFB::requires_padding () const  [inline]
```

9.2.3.10 type()

```
core::AlgorithmType filevault::algorithms::symmetric::AES_CFB::type () const  [override], [virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.2.4 Member Data Documentation

9.2.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::AES_CFB::botan_name_  [private]
```

9.2.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::AES_CFB::key_bits_  [private]
```

9.2.4.3 type_

```
core::AlgorithmType filevault::algorithms::symmetric::AES_CFB::type_  [private]
```

The documentation for this class was generated from the following files:

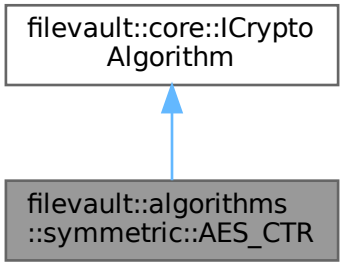
- include/filevault/algorithms/symmetric/aes_cfb.hpp
- src/algorithms/symmetric/aes_cfb.cpp

9.3 filevault::algorithms::symmetric::AES_CTR Class Reference

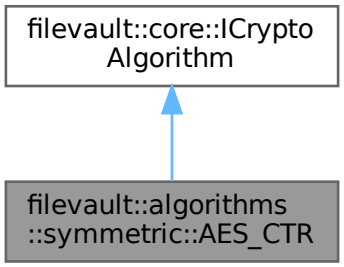
AES-CTR encryption (stream cipher mode, not authenticated).

`#include <aes_ctr.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::AES_CTR:



Collaboration diagram for filevault::algorithms::symmetric::AES_CTR:



Public Member Functions

- AES_CTR (size_t key_bits=256)
- virtual ~AES_CTR ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.

- size_t nonce_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.3.1 Detailed Description

AES-CTR encryption (stream cipher mode, not authenticated).
Supports 128, 192, and 256 bit keys. No padding needed (stream cipher). Requires unique nonce/counter for each encryption.

Warning

This mode does NOT provide authentication!

9.3.2 Constructor & Destructor Documentation

9.3.2.1 AES_CTR()

```
filevault::algorithms::symmetric::AES_CTR::AES_CTR (
            size_t key_bits = 256)  [explicit]
```

9.3.2.2 ~AES_CTR()

```
virtual filevault::algorithms::symmetric::AES_CTR::~AES_CTR ()  [virtual], [default]
```

9.3.3 Member Function Documentation

9.3.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_CTR::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

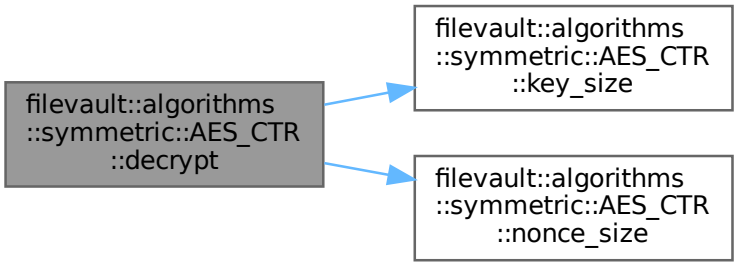
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.3.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_CTR::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

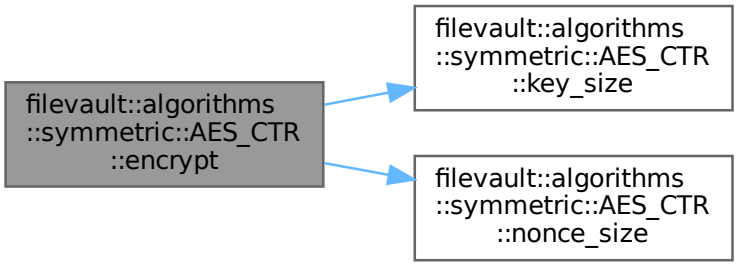
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.3.3.3 is_suitable_for()

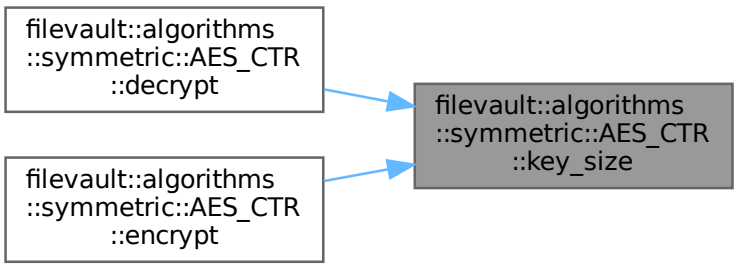
```
bool filevault::algorithms::symmetric::AES_CTR::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.3.3.4 key_size()

```
size_t filevault::algorithms::symmetric::AES_CTR::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.3.3.5 name()

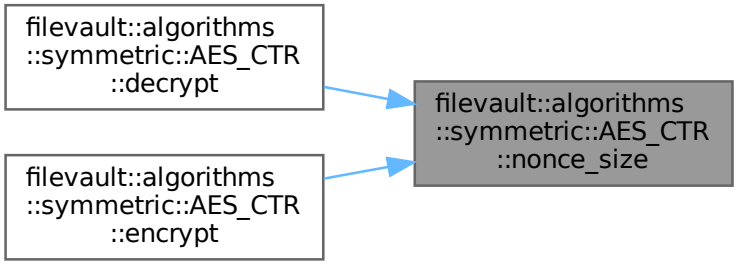
```
std::string filevault::algorithms::symmetric::AES_CTR::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.3.3.6 nonce_size()

```
size_t filevault::algorithms::symmetric::AES_CTR::nonce_size () const  [inline]
```

Here is the caller graph for this function:



9.3.3.7 type()

```
core::AlgorithmType filevault::algorithms::symmetric::AES_CTR::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.3.4 Member Data Documentation

9.3.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::AES_CTR::botan_name_  [private]
```

9.3.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::AES_CTR::key_bits_  [private]
```

9.3.4.3 type_

```
core::AlgorithmType filevault::algorithms::symmetric::AES_CTR::type_  [private]
```

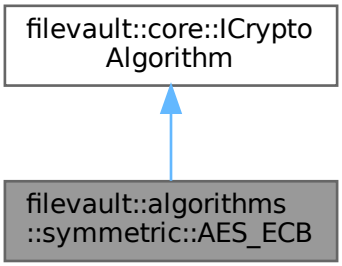
The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/aes_ctr.hpp
- src/algorithms/symmetric/aes_ctr.cpp

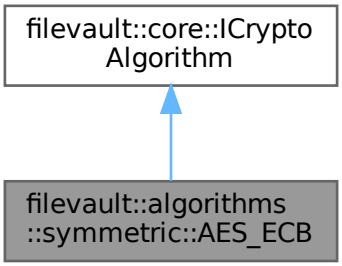
9.4 filevault::algorithms::symmetric::AES_ECB Class Reference

AES-ECB encryption algorithm.
`#include <aes_ecb.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::AES_ECB:



Collaboration diagram for filevault::algorithms::symmetric::AES_ECB:



Public Member Functions

- AES_ECB (size_t key_bits=256)

  Construct AES-ECB with specified key size.
- ~AES_ECB () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t block_size () const
- bool requires_padding () const

- bool is_authenticated () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.4.1 Detailed Description

AES-ECB encryption algorithm.
SECURITY WARNING: ECB mode is insecure!

- Identical plaintext blocks produce identical ciphertext blocks
- Patterns in plaintext are visible in ciphertext
- Use CBC, CTR, or GCM instead for real security

Only use for:

- Educational purposes
- Single block encryption
- Legacy system compatibility

9.4.2 Constructor & Destructor Documentation

9.4.2.1 AES_ECB()

```
filevault::algorithms::symmetric::AES_ECB::AES_ECB (
            size_t key_bits = 256)  [explicit]
```

Construct AES-ECB with specified key size.

Parameters

key_bits	Key size in bits (128, 192, or 256)

9.4.2.2 ~AES_ECB()

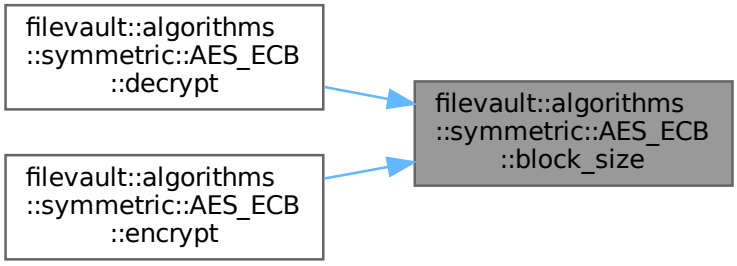
```
filevault::algorithms::symmetric::AES_ECB::~AES_ECB ()  [override], [default]
```

9.4.3 Member Function Documentation

9.4.3.1 block_size()

```
size_t filevault::algorithms::symmetric::AES_ECB::block_size () const  [inline]
```

Here is the caller graph for this function:



9.4.3.2 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_ECB::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

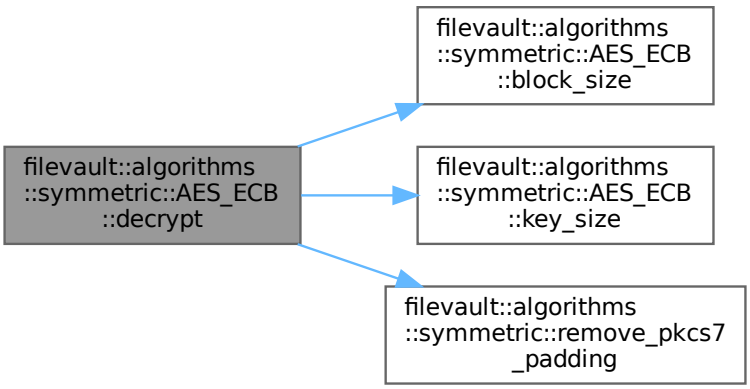
ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

Here is the call graph for this function:



9.4.3.3 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_ECB::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

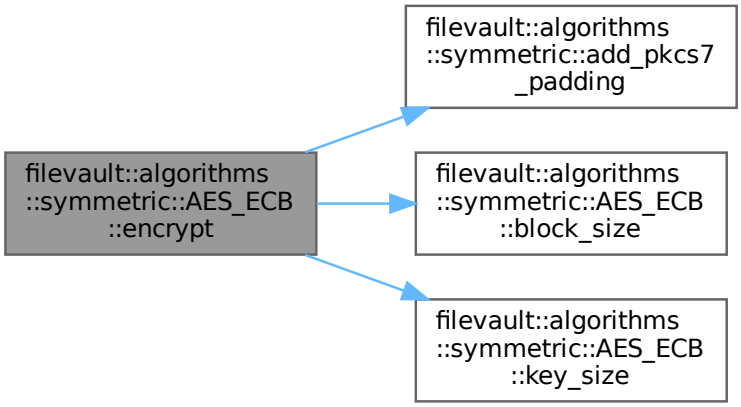
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.4.3.4 is_authenticated()

```
bool filevault::algorithms::symmetric::AES_ECB::is_authenticated () const  [inline]
```

9.4.3.5 is_suitable_for()

```
bool filevault::algorithms::symmetric::AES_ECB::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

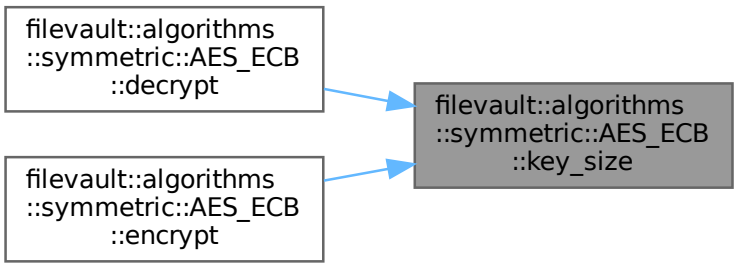
Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.4.3.6 key_size()

```
size_t filevault::algorithms::symmetric::AES_ECB::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



9.4.3.7 name()

```
std::string filevault::algorithms::symmetric::AES_ECB::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.4.3.8 requires_padding()

```
bool filevault::algorithms::symmetric::AES_ECB::requires_padding () const  [inline]
```

9.4.3.9 type()

```
core::AlgorithmType filevault::algorithms::symmetric::AES_ECB::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.4.4 Member Data Documentation

9.4.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::AES_ECB::botan_name_  [private]
```

9.4.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::AES_ECB::key_bits_  [private]
```

9.4.4.3 type_

```
core::AlgorithmType filevault::algorithms::symmetric::AES_ECB::type_  [private]
```

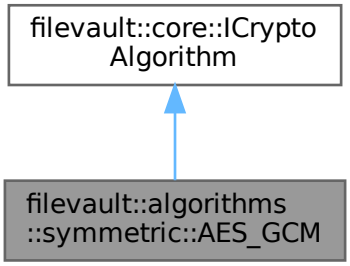
The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/aes_ecb.hpp
- src/algorithms/symmetric/aes_ecb.cpp

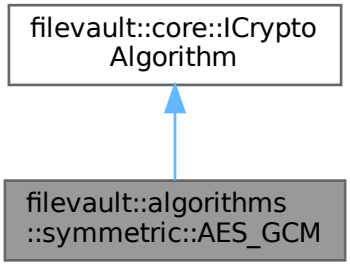
9.5 filevault::algorithms::symmetric::AES_GCM Class Reference

AES-GCM AEAD encryption.
`#include <aes_gcm.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::AES_GCM:



Collaboration diagram for filevault::algorithms::symmetric::AES_GCM:



Public Member Functions

- AES_GCM (size_t key_bits=256)
- virtual ~AES_GCM ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t nonce_size () const
- size_t tag_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.5.1 Detailed Description

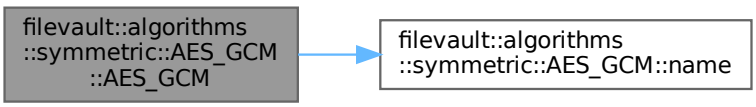
AES-GCM AEAD encryption.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 AES_GCM()

```
filevault::algorithms::symmetric::AES_GCM::AES_GCM (
            size_t key_bits = 256)  [explicit]
```

Here is the call graph for this function:



9.5.2.2 ~AES_GCM()

```
virtual filevault::algorithms::symmetric::AES_GCM::~AES_GCM ()  [virtual], [default]
```

9.5.3 Member Function Documentation

9.5.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_GCM::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

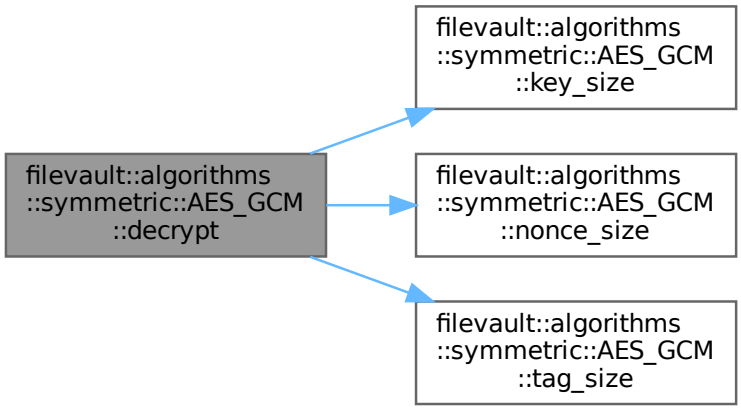
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:

9.5.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_GCM::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

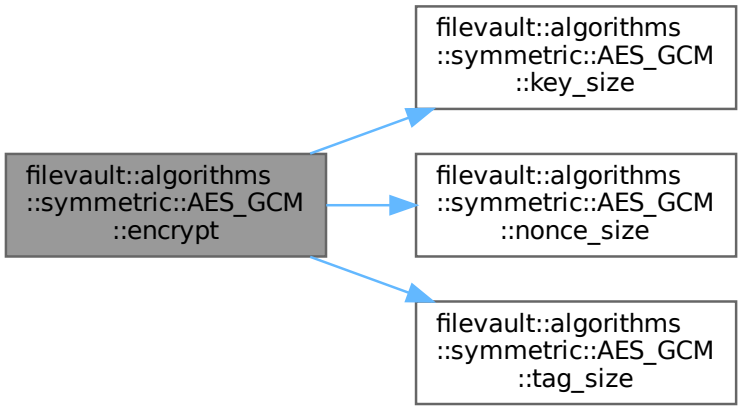
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:

9.5.3.3 is_suitable_for()

```
bool filevault::algorithms::symmetric::AES_GCM::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.5.3.4 key_size()

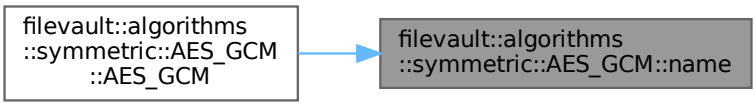
```
size_t filevault::algorithms::symmetric::AES_GCM::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:

9.5.3.5 name()

```
std::string filevault::algorithms::symmetric::AES_GCM::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.5.3.6 nonce_size()

```
size_t filevault::algorithms::symmetric::AES_GCM::nonce_size () const  [inline]
```

Here is the caller graph for this function:

9.5.3.7 tag_size()

```
size_t filevault::algorithms::symmetric::AES_GCM::tag_size () const  [inline]
```

Here is the caller graph for this function:

9.5.3.8 type()

```
core::AlgorithmType filevault::algorithms::symmetric::AES_GCM::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.5.4 Member Data Documentation

9.5.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::AES_GCM::botan_name_  [private]
```

9.5.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::AES_GCM::key_bits_  [private]
```

9.5.4.3 type_

`core::AlgorithmType filevault::algorithms::symmetric::AES_GCM::type_ [private]`

The documentation for this class was generated from the following files:

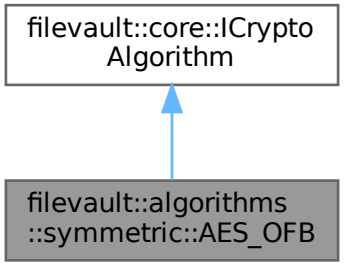
- include/filevault/algorithms/symmetric/aes_gcm.hpp
- src/algorithms/symmetric/aes_gcm.cpp

9.6 filevault::algorithms::symmetric::AES_OFB Class Reference

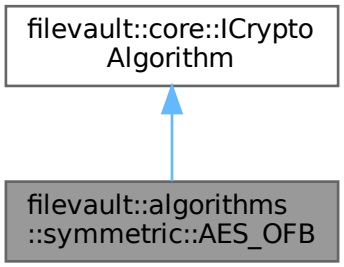
AES-OFB encryption algorithm.

`#include <aes_ofb.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::AES_OFB:



Collaboration diagram for filevault::algorithms::symmetric::AES_OFB:



Public Member Functions

- AES_OFB (size_t key_bits=256)

  Construct AES-OFB with specified key size.
- ~AES_OFB () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

Get algorithm type.

- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.

- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.

- size_t key_size () const override

  Get recommended key size in bytes.

- size_t iv_size () const
- size_t block_size () const
- bool requires_padding () const
- bool is_authenticated () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.6.1 Detailed Description

AES-OFB encryption algorithm.
Features:

- True stream cipher mode
- No error propagation (bit flip affects only one bit)
- Keystream can be pre-computed
- No padding required

9.6.2 Constructor & Destructor Documentation

9.6.2.1 AES_OFB()

```
filevault::algorithms::symmetric::AES_OFB::AES_OFB (
            size_t key_bits = 256)  [explicit]
```

Construct AES-OFB with specified key size.

Parameters

key_bits	Key size in bits (128, 192, or 256)

9.6.2.2 ~AES_OFB()

```
filevault::algorithms::symmetric::AES_OFB::~AES_OFB ()  [override], [default]
```

9.6.3 Member Function Documentation

9.6.3.1 block_size()

```
size_t filevault::algorithms::symmetric::AES_OFB::block_size () const  [inline]
```

9.6.3.2 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_OFB::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

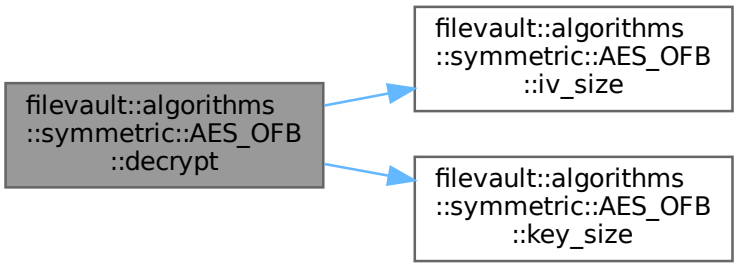
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.6.3.3 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_OFB::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

Parameters

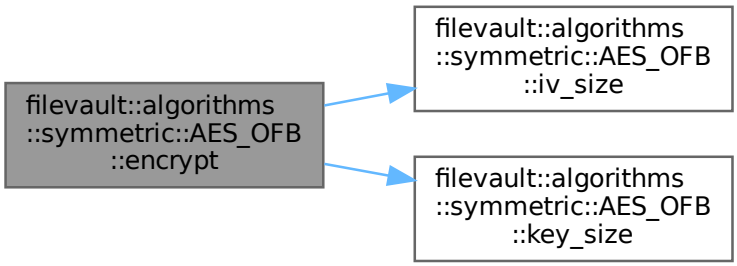
plaintext	Input data to encrypt
key	Encryption key (derived from password)

config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.6.3.4 is_authenticated()

```
bool filevault::algorithms::symmetric::AES_OFB::is_authenticated () const  [inline]
```

9.6.3.5 is_suitable_for()

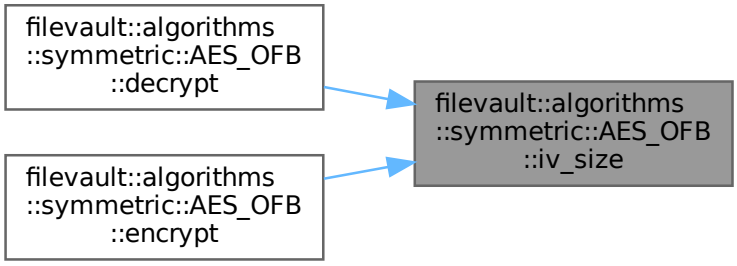
```
bool filevault::algorithms::symmetric::AES_OFB::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.6.3.6 iv_size()

```
size_t filevault::algorithms::symmetric::AES_OFB::iv_size () const  [inline]
```

Here is the caller graph for this function:



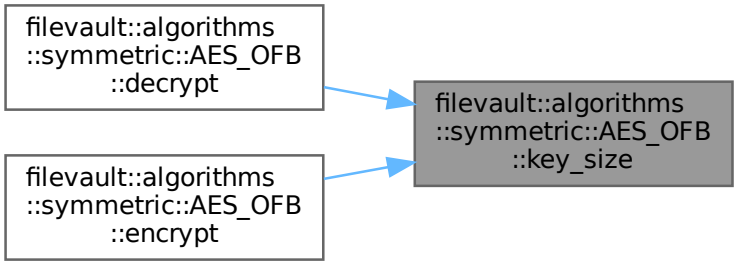
9.6.3.7 key_size()

`size_t filevault::algorithms::symmetric::AES_OFB::key_size () const [inline], [override], [virtual]`

Get recommended key size in bytes.

Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



9.6.3.8 name()

`std::string filevault::algorithms::symmetric::AES_OFB::name () const [override], [virtual]`

Get algorithm name.

Implements filevault::core::ICryptoAlgorithm.

9.6.3.9 requires_padding()

`bool filevault::algorithms::symmetric::AES_OFB::requires_padding () const [inline]`

9.6.3.10 type()

`core::AlgorithmType filevault::algorithms::symmetric::AES_OFB::type () const [override], [virtual]`

Get algorithm type.

Implements filevault::core::ICryptoAlgorithm.

9.6.4 Member Data Documentation

9.6.4.1 botan_name_

`std::string filevault::algorithms::symmetric::AES_OFB::botan_name_ [private]`

9.6.4.2 key_bits_

`size_t filevault::algorithms::symmetric::AES_OFB::key_bits_ [private]`

9.6.4.3 type_

`core::AlgorithmType filevault::algorithms::symmetric::AES_OFB::type_ [private]`

The documentation for this class was generated from the following files:

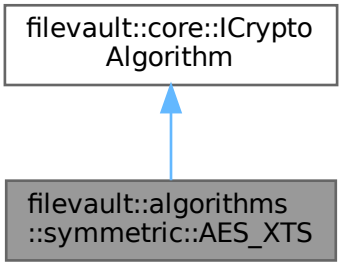
- include/filevault/algorithms/symmetric/aes_ofb.hpp
- src/algorithms/symmetric/aes_ofb.cpp

9.7 filevault::algorithms::symmetric::AES_XTS Class Reference

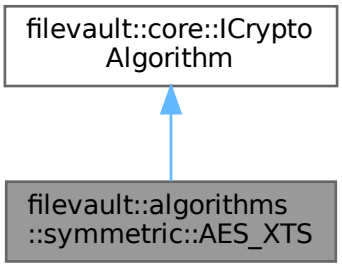
AES-XTS encryption algorithm.

`#include <aes_xts.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::AES_XTS:



Collaboration diagram for filevault::algorithms::symmetric::AES_XTS:



Public Member Functions

- AES_XTS (size_t key_bits=256)

  Construct AES-XTS with specified key size.
- ~AES_XTS () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

Get recommended key size in bytes.

- size_t tweak_size () const
- size_t block_size () const
- bool requires_padding () const
- bool is_authenticated () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.7.1 Detailed Description

AES-XTS encryption algorithm.
Features:

- Designed for disk/sector encryption
- Uses two keys (effectively double key size)
- Tweak value provides sector-level randomization
- No ciphertext expansion (same size as plaintext)
- Not suitable for streaming data

9.7.2 Constructor & Destructor Documentation

9.7.2.1 AES_XTS()

```
filevault::algorithms::symmetric::AES_XTS::AES_XTS (
            size_t key_bits = 256)  [explicit]
```

Construct AES-XTS with specified key size.

Parameters

key_bits	Key size per cipher in bits (128 or 256) Total key is 2x this (256 or 512 bits)

9.7.2.2 ~AES_XTS()

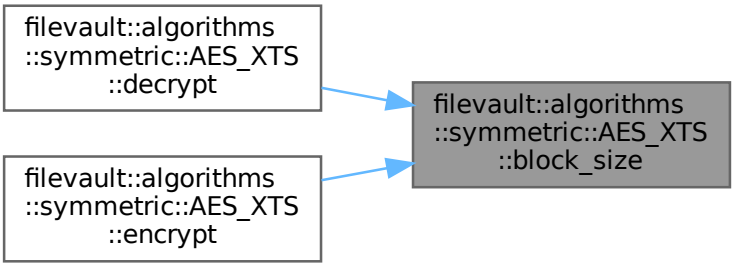
```
filevault::algorithms::symmetric::AES_XTS::~AES_XTS ()  [override], [default]
```

9.7.3 Member Function Documentation

9.7.3.1 block_size()

```
size_t filevault::algorithms::symmetric::AES_XTS::block_size () const  [inline]
```

Here is the caller graph for this function:



9.7.3.2 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_XTS::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

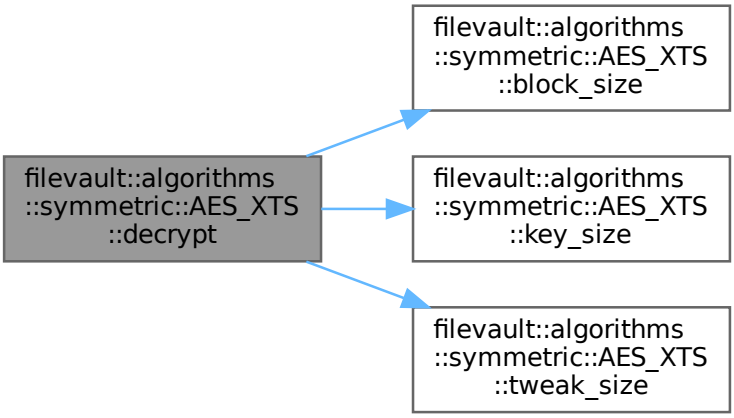
ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

Here is the call graph for this function:



9.7.3.3 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::AES_XTS::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

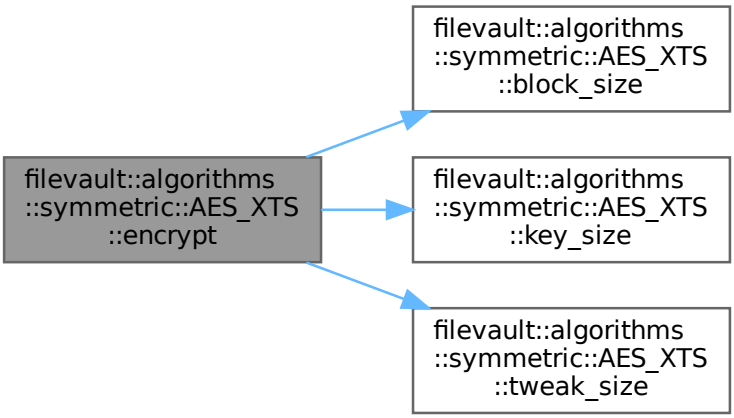
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.7.3.4 is_authenticated()

```
bool filevault::algorithms::symmetric::AES_XTS::is_authenticated () const  [inline]
```

9.7.3.5 is_suitable_for()

```
bool filevault::algorithms::symmetric::AES_XTS::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

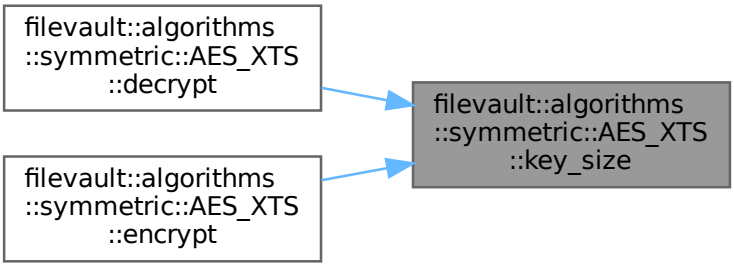
Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.7.3.6 key_size()

```
size_t filevault::algorithms::symmetric::AES_XTS::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



9.7.3.7 name()

```
std::string filevault::algorithms::symmetric::AES_XTS::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

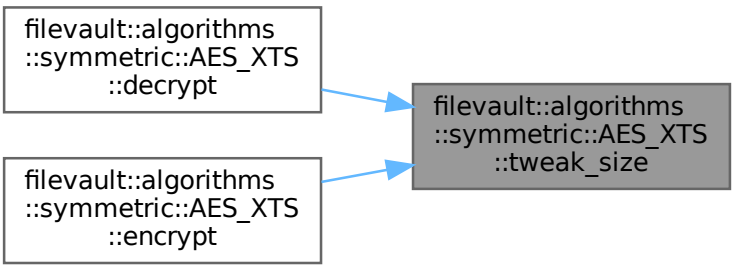
9.7.3.8 requires_padding()

```
bool filevault::algorithms::symmetric::AES_XTS::requires_padding () const  [inline]
```

9.7.3.9 tweak_size()

```
size_t filevault::algorithms::symmetric::AES_XTS::tweak_size () const  [inline]
```

Here is the caller graph for this function:



9.7.3.10 type()

```
core::AlgorithmType filevault::algorithms::symmetric::AES_XTS::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.7.4 Member Data Documentation

9.7.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::AES_XTS::botan_name_  [private]
```

9.7.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::AES_XTS::key_bits_  [private]
```

9.7.4.3 type_

```
core::AlgorithmType filevault::algorithms::symmetric::AES_XTS::type_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/aes_xts.hpp
- src/algorithms/symmetric/aes_xts.cpp

9.8 filevault::cli::Application Class Reference

Main CLI application.

`#include <app.hpp>`

Public Member Functions

- Application ()
- ~Application ()
- void initialize ()

  Initialize application and register commands.
- int run (int argc, char **argv)

  Run application with command line arguments.

Private Member Functions

- void register_commands ()
- void setup_logging ()

Private Attributes

- CLI::App app_
- std::unique_ptr< core::CryptoEngine > engine_
- std::vector< std::unique_ptr< ICommand > > commands_
- bool verbose_ = false
- std::string log_level_ = "info"

9.8.1 Detailed Description

Main CLI application.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 Application()

```
filevault::cli::Application::Application ()
```

9.8.2.2 ~Application()

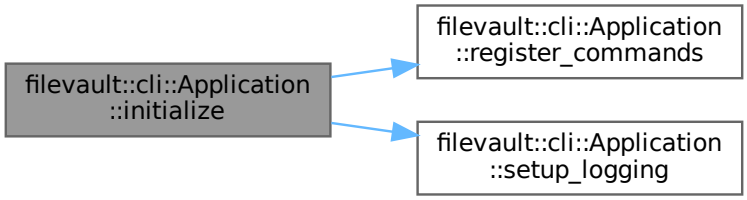
```
filevault::cli::Application::~Application ()  [default]
```

9.8.3 Member Function Documentation

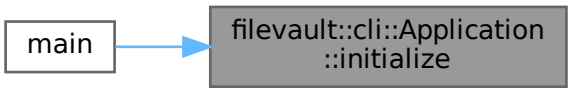
9.8.3.1 initialize()

```
void filevault::cli::Application::initialize ()
```

Initialize application and register commands.
Here is the call graph for this function:



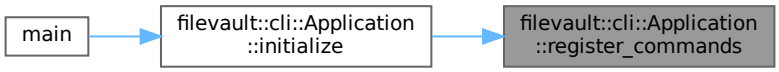
Here is the caller graph for this function:



9.8.3.2 register_commands()

```
void filevault::cli::Application::register_commands ()  [private]
```

Here is the caller graph for this function:

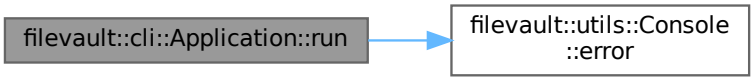


9.8.3.3 run()

```
int filevault::cli::Application::run (
            int argc,
            char ** argv)
```

Run application with command line arguments.

Here is the call graph for this function:

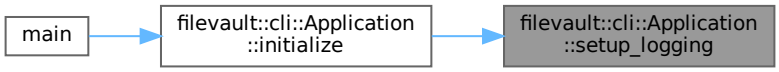


Here is the caller graph for this function:

9.8.3.4 setup_logging()

```
void filevault::cli::Application::setup_logging ()  [private]
```

Here is the caller graph for this function:



9.8.4 Member Data Documentation

9.8.4.1 app_

```
CLI::App filevault::cli::Application::app_  [private]
```

9.8.4.2 commands_

```
std::vector<std::unique_ptr<ICommand> > filevault::cli::Application::commands_  [private]
```

9.8.4.3 engine_

```
std::unique_ptr<core::CryptoEngine> filevault::cli::Application::engine_  [private]
```

9.8.4.4 log_level_

```
std::string filevault::cli::Application::log_level_ = "info"  [private]
```

9.8.4.5 verbose_

```
bool filevault::cli::Application::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

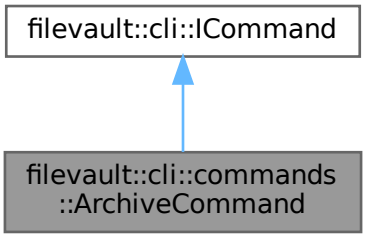
- include/filevault/cli/app.hpp
- src/cli/app.cpp

9.9 filevault::cli::commands::ArchiveCommand Class Reference

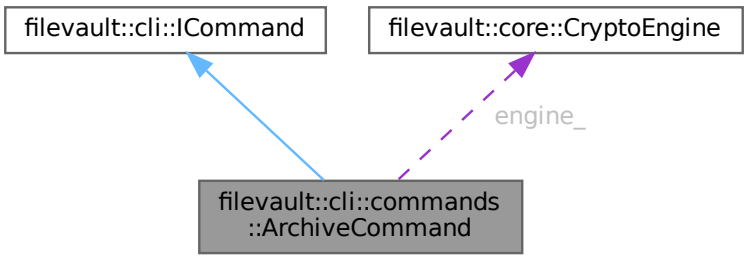
Archive command - compress and encrypt multiple files.

```
#include <archive_cmd.hpp>
```

Inheritance diagram for filevault::cli::commands::ArchiveCommand:



Collaboration diagram for filevault::cli::commands::ArchiveCommand:



Public Member Functions

- ArchiveCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- int do_create ()
- int do_extract ()

Private Attributes

- core::CryptoEngine & engine_
- std::vector< std::string > input_files_
- std::string output_file_
- std::string password_
- std::string algorithm_ = "aes-256-gcm"
- std::string compression_ = "zlib"
- std::string kdf_ = "argon2id"
- std::string security_level_ = "medium"
- bool extract_ = false
- bool verbose_ = false
- std::string extract_dir_ = "."

9.9.1 Detailed Description

Archive command - compress and encrypt multiple files.
Creates an archive from multiple files, compresses it, then encrypts. Format: [metadata][file1_data][file2_data]...

9.9.2 Constructor & Destructor Documentation

9.9.2.1 ArchiveCommand()

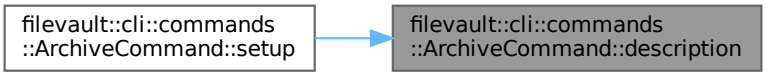
```
filevault::cli::commands::ArchiveCommand::ArchiveCommand (
            core::CryptoEngine & engine)  [inline], [explicit]
```

9.9.3 Member Function Documentation

9.9.3.1 description()

```
std::string filevault::cli::commands::ArchiveCommand::description () const  [inline], [override],
[virtual]
```

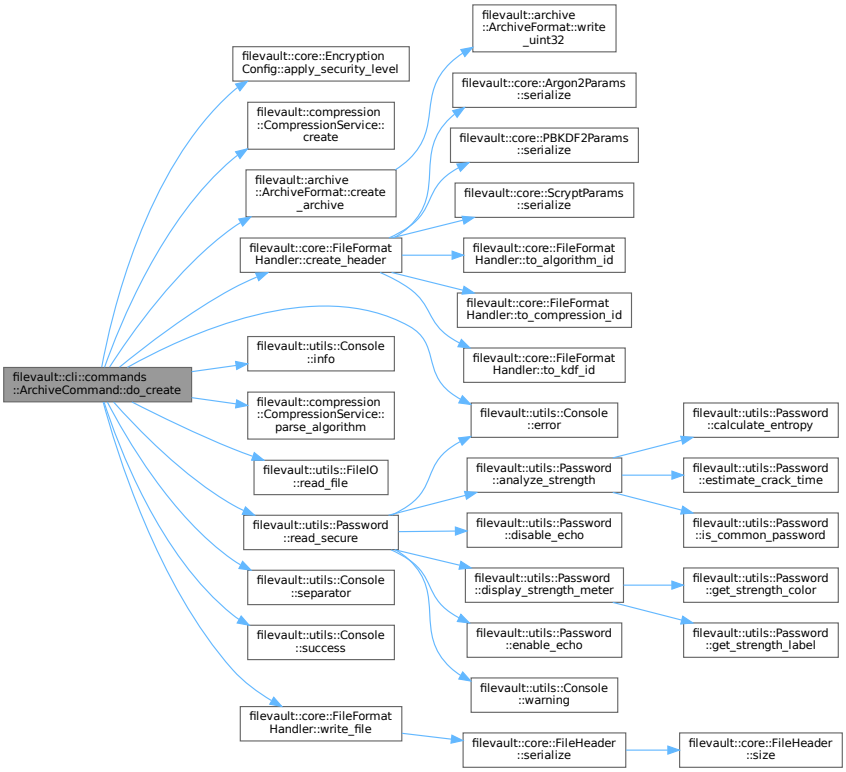
Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:



9.9.3.2 do_create()

```
int filevault::cli::commands::ArchiveCommand::do_create ()  [private]
```

Here is the call graph for this function:

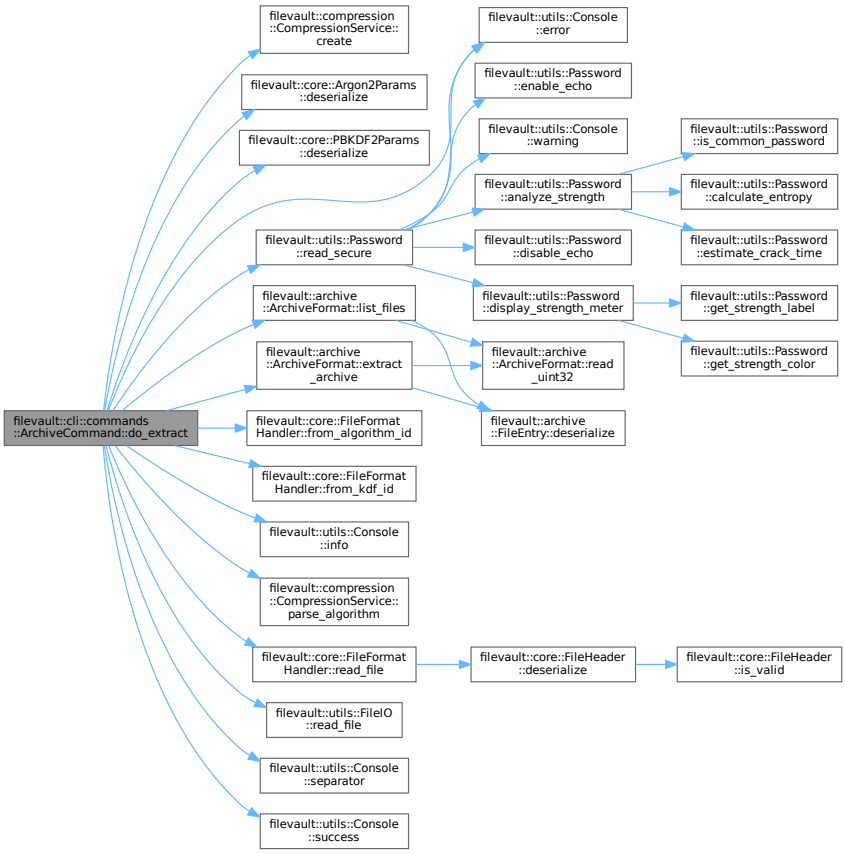


Here is the caller graph for this function:

9.9.3.3 do_extract()

```
int filevault::cli::commands::ArchiveCommand::do_extract ()  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:

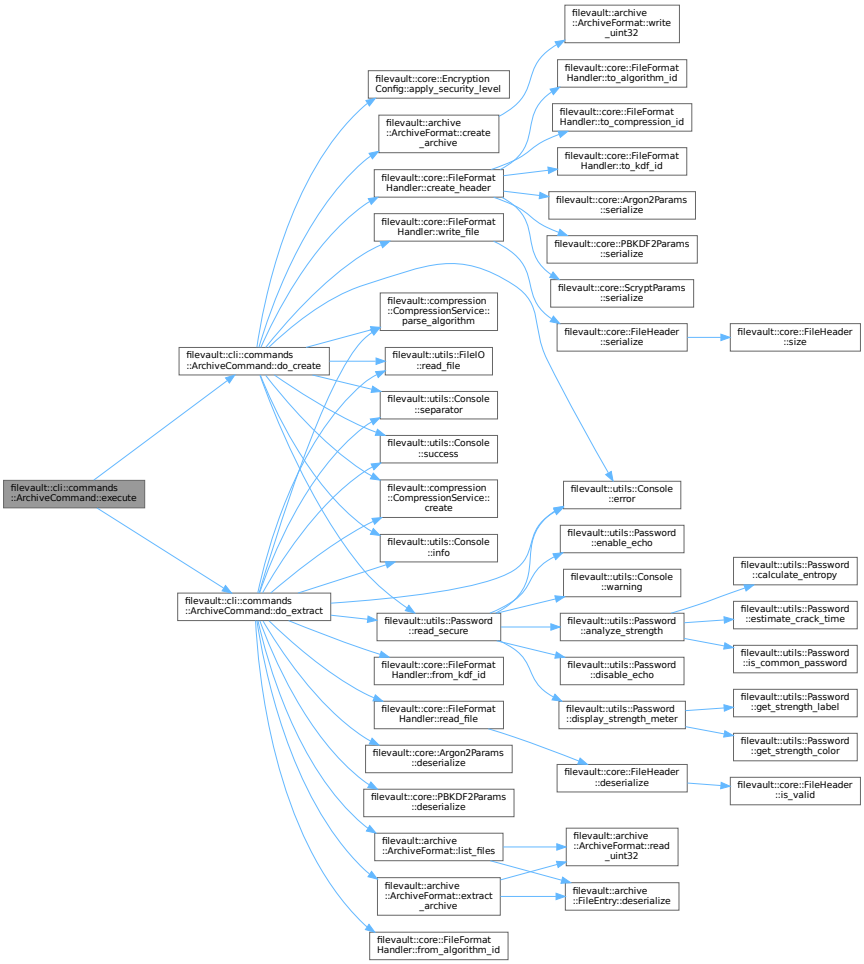
9.9.3.4 execute()

`int filevault::cli::commands::ArchiveCommand::execute () [override], [virtual]`

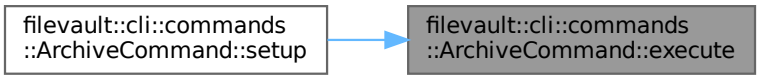
Execute the command.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



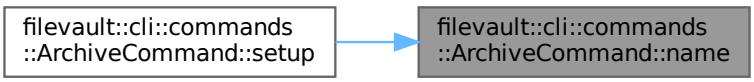
9.9.3.5 name()

`std::string filevault::cli::commands::ArchiveCommand::name () const [inline], [override], [virtual]`

Get command name.

Implements filevault::cli::ICommand.

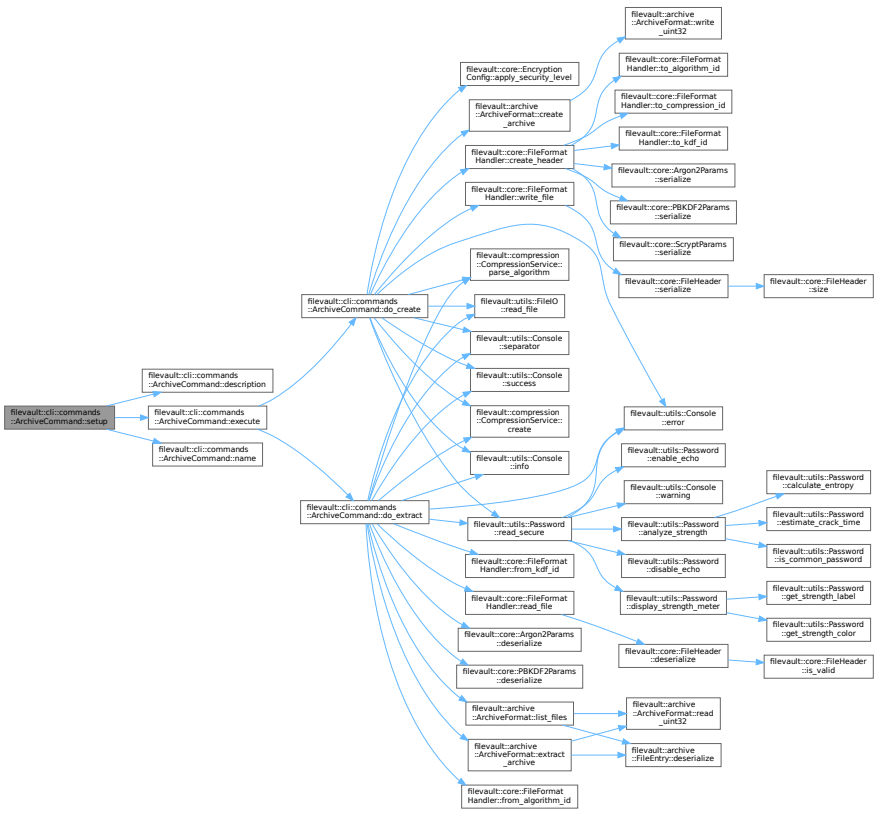
Here is the caller graph for this function:



9.9.3.6 setup()

```
void filevault::cli::commands::ArchiveCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



9.9.4 Member Data Documentation

9.9.4.1 algorithm_

```
std::string filevault::cli::commands::ArchiveCommand::algorithm_ = "aes-256-gcm"  [private]
```

9.9.4.2 compression_

```
std::string filevault::cli::commands::ArchiveCommand::compression_ = "zlib"  [private]
```

9.9.4.3 engine_

```
core::CryptoEngine& filevault::cli::commands::ArchiveCommand::engine_  [private]
```

9.9.4.4 extract_

```
bool filevault::cli::commands::ArchiveCommand::extract_ = false  [private]
```

9.9.4.5 extract_dir_

```
std::string filevault::cli::commands::ArchiveCommand::extract_dir_ = "."  [private]
```

9.9.4.6 input_files_

```
std::vector<std::string> filevault::cli::commands::ArchiveCommand::input_files_  [private]
```

9.9.4.7 kdf_

```
std::string filevault::cli::commands::ArchiveCommand::kdf_ = "argon2id"  [private]
```

9.9.4.8 output_file_

```
std::string filevault::cli::commands::ArchiveCommand::output_file_  [private]
```

9.9.4.9 password_

```
std::string filevault::cli::commands::ArchiveCommand::password_  [private]
```

9.9.4.10 security_level_

```
std::string filevault::cli::commands::ArchiveCommand::security_level_ = "medium"  [private]
```

9.9.4.11 verbose_

```
bool filevault::cli::commands::ArchiveCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/archive_cmd.hpp
- src/cli/commands/archive_cmd.cpp

9.10 filevault::archive::ArchiveFormat Class Reference

Simple archive format handler.

`#include <archive_format.hpp>`

Static Public Member Functions

- static std::vector< uint8_t > create_archive (const std::vector< std::filesystem::path > &files)

  Create archive from multiple files.
- static bool extract_archive (std::span< const uint8_t > archive_data, const std::filesystem::path &output_dir)

  Extract files from archive.
- static std::vector< FileEntry > list_files (std::span< const uint8_t > archive_data)

  List files in archive.

Static Public Attributes

- static constexpr char MAGIC [7] = "FVARCH"
- static constexpr uint8_t VERSION = 1

Static Private Member Functions

- static void write_uint32 (std::vector< uint8_t > &buffer, uint32_t value)
- static void write_uint64 (std::vector< uint8_t > &buffer, uint64_t value)
- static uint32_t read_uint32 (std::span< const uint8_t > data, size_t &offset)
- static uint64_t read_uint64 (std::span< const uint8_t > data, size_t &offset)
- static std::string read_string (std::span< const uint8_t > data, size_t &offset)

9.10.1 Detailed Description

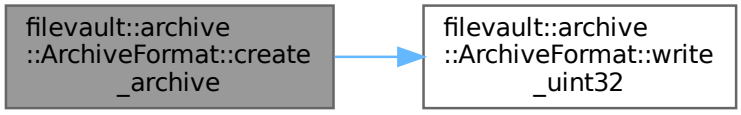
Simple archive format handler.
Format: [Magic: "FVARCH"] [Version: 1 byte] [Entry count: 4 bytes] [Entry1 metadata] [Entry2 metadata] ... [File1 data] [File2 data] ...

9.10.2 Member Function Documentation

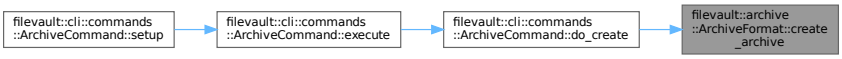
9.10.2.1 create_archive()

```
std::vector< uint8_t > filevault::archive::ArchiveFormat::create_archive (
            const std::vector< std::filesystem::path > & files)  [static]
```

Create archive from multiple files.
Here is the call graph for this function:



Here is the caller graph for this function:

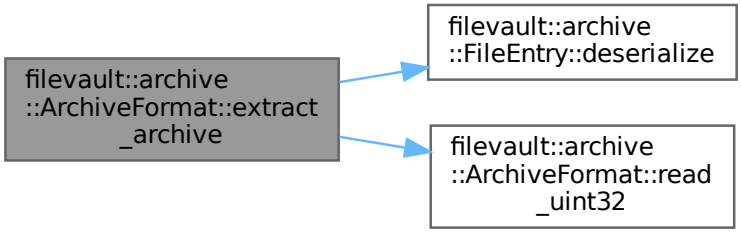


9.10.2.2 extract_archive()

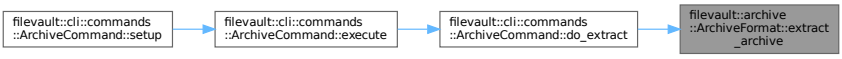
```
bool filevault::archive::ArchiveFormat::extract_archive (
            std::span< const uint8_t > archive_data,
            const std::filesystem::path & output_dir)  [static]
```

Extract files from archive.

Here is the call graph for this function:



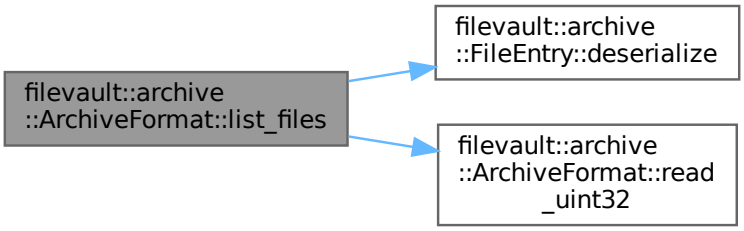
Here is the caller graph for this function:



9.10.2.3 list_files()

```
std::vector< FileEntry > filevault::archive::ArchiveFormat::list_files (
            std::span< const uint8_t > archive_data)  [static]
```

List files in archive.
Here is the call graph for this function:

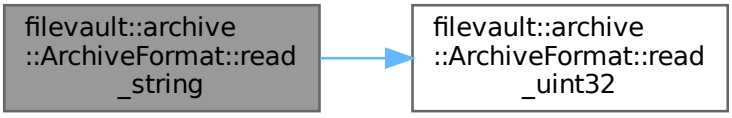


Here is the caller graph for this function:

9.10.2.4 read_string()

```
std::string filevault::archive::ArchiveFormat::read_string (
            std::span< const uint8_t > data,
            size_t & offset)  [static], [private]
```

Here is the call graph for this function:



9.10.2.5 read_uint32()

```
uint32_t filevault::archive::ArchiveFormat::read_uint32 (
            std::span< const uint8_t > data,
            size_t & offset)  [static], [private]
```

Here is the caller graph for this function:

9.10.2.6 read_uint64()

```
uint64_t filevault::archive::ArchiveFormat::read_uint64 (
            std::span< const uint8_t > data,
            size_t & offset)  [static], [private]
```

9.10.2.7 write_uint32()

```
void filevault::archive::ArchiveFormat::write_uint32 (
            std::vector< uint8_t > & buffer,
            uint32_t value)  [static], [private]
```

Here is the caller graph for this function:

9.10.2.8 write_uint64()

```
void filevault::archive::ArchiveFormat::write_uint64 (
            std::vector< uint8_t > & buffer,
            uint64_t value)  [static], [private]
```

9.10.3 Member Data Documentation

9.10.3.1 MAGIC

```
char filevault::archive::ArchiveFormat::MAGIC[7] = "FVARCH"  [static], [constexpr]
```

9.10.3.2 VERSION

```
uint8_t filevault::archive::ArchiveFormat::VERSION = 1  [static], [constexpr]
```

The documentation for this class was generated from the following files:

- include/filevault/archive/archive_format.hpp
- src/archive/archive_format.cpp

9.11 filevault::core::Argon2Params Struct Reference

KDF parameters for Argon2.

`#include <file_format.hpp>`

Public Member Functions

- std::vector< uint8_t > serialize () const

Static Public Member Functions

- static Argon2Params deserialize (std::span< const uint8_t > data)

Public Attributes

- uint32_t memory_kb = 65536
- uint32_t iterations = 3
- uint32_t parallelism = 4

9.11.1 Detailed Description

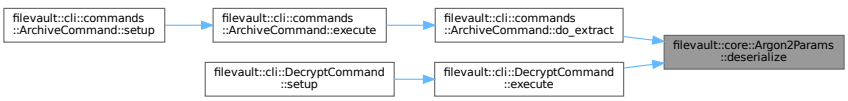
KDF parameters for Argon2.

9.11.2 Member Function Documentation

9.11.2.1 deserialize()

```
Argon2Params filevault::core::Argon2Params::deserialize (
            std::span< const uint8_t > data)  [static]
```

Here is the caller graph for this function:



9.11.2.2 serialize()

```
std::vector< uint8_t > filevault::core::Argon2Params::serialize () const
```

Here is the caller graph for this function:

9.11.3 Member Data Documentation

9.11.3.1 iterations

```
uint32_t filevault::core::Argon2Params::iterations = 3
```

9.11.3.2 memory_kb

```
uint32_t filevault::core::Argon2Params::memory_kb = 65536
```

9.11.3.3 parallelism

```
uint32_t filevault::core::Argon2Params::parallelism = 4
```

The documentation for this struct was generated from the following files:

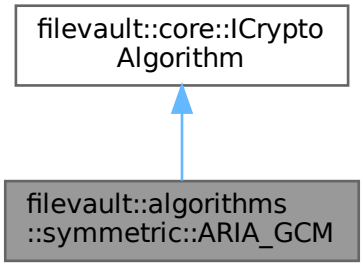
- include/filevault/core/file_format.hpp
- src/format/file_format.cpp

9.12 filevault::algorithms::symmetric::ARIA_GCM Class Reference

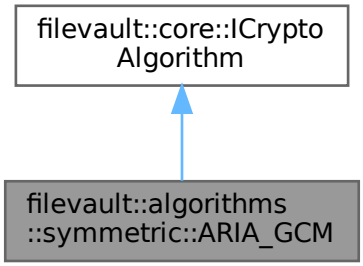
ARIA-GCM AEAD encryption.

```
#include <aria_gcm.hpp>
```

Inheritance diagram for filevault::algorithms::symmetric::ARIA_GCM:



Collaboration diagram for filevault::algorithms::symmetric::ARIA_GCM:



Public Member Functions

- ARIA_GCM (size_t key_bits=256)

  Construct ARIA-GCM with specified key size.
- virtual ~ARIA_GCM ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t nonce_size () const
- size_t tag_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.12.1 Detailed Description

ARIA-GCM AEAD encryption.
Supports 128, 192, and 256-bit keys. Uses GCM mode for authenticated encryption.

9.12.2 Constructor & Destructor Documentation

9.12.2.1 ARIA_GCM()

```
filevault::algorithms::symmetric::ARIA_GCM::ARIA_GCM (
            size_t key_bits = 256)  [explicit]
```

Construct ARIA-GCM with specified key size.

Parameters

key_bits	Key size in bits (128, 192, or 256)

9.12.2.2 ~ARIA_GCM()

```
virtual filevault::algorithms::symmetric::ARIA_GCM::~ARIA_GCM ()  [virtual], [default]
```

9.12.3 Member Function Documentation

9.12.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::ARIA_GCM::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

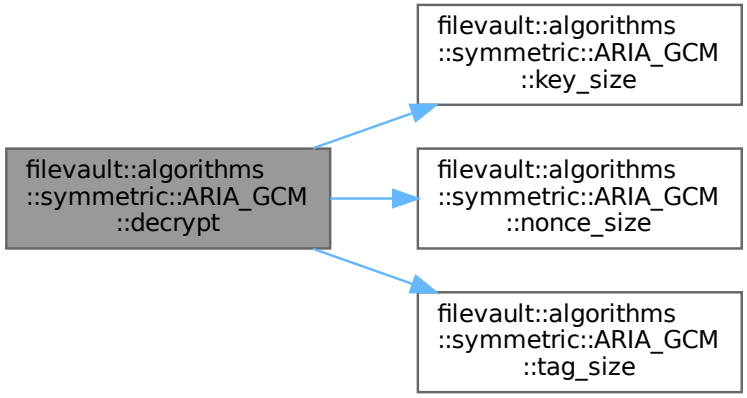
ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

Here is the call graph for this function:



9.12.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::ARIA_GCM::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

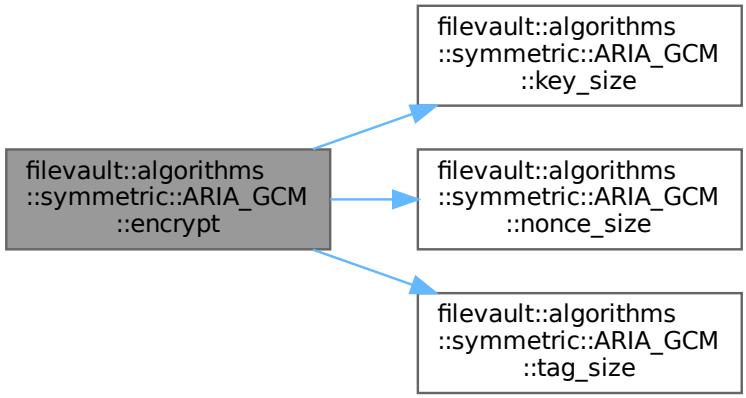
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.12.3.3 is_suitable_for()

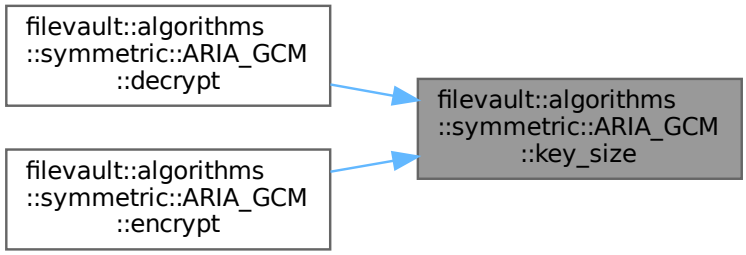
```
bool filevault::algorithms::symmetric::ARIA_GCM::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.12.3.4 key_size()

```
size_t filevault::algorithms::symmetric::ARIA_GCM::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.12.3.5 name()

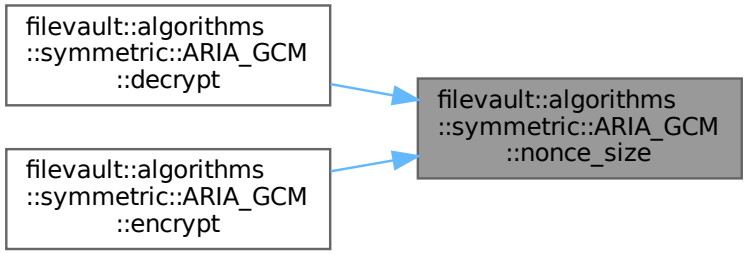
```
std::string filevault::algorithms::symmetric::ARIA_GCM::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.12.3.6 nonce_size()

```
size_t filevault::algorithms::symmetric::ARIA_GCM::nonce_size () const  [inline]
```

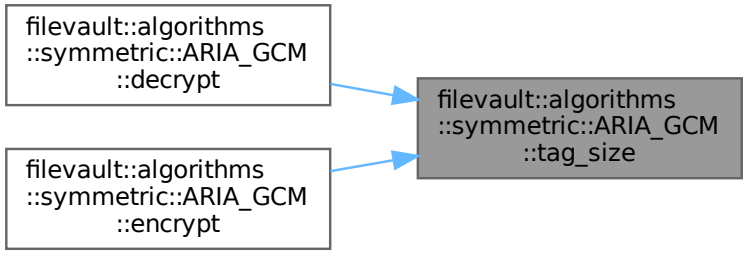
Here is the caller graph for this function:



9.12.3.7 tag_size()

```
size_t filevault::algorithms::symmetric::ARIA_GCM::tag_size () const  [inline]
```

Here is the caller graph for this function:



9.12.3.8 type()

```
core::AlgorithmType filevault::algorithms::symmetric::ARIA_GCM::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.12.4 Member Data Documentation

9.12.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::ARIA_GCM::botan_name_  [private]
```

9.12.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::ARIA_GCM::key_bits_  [private]
```

9.12.4.3 type_

```
core::AlgorithmType filevault::algorithms::symmetric::ARIA_GCM::type_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/aria_gcm.hpp
- src/algorithms/symmetric/aria_gcm.cpp

9.13 filevault::cli::AsymmetricBenchmarkResult Struct Reference

```
#include <benchmark_cmd.hpp>
```

Public Attributes

- std::string algorithm
- double keygen_ms = 0
- double encrypt_ms = 0
- double decrypt_ms = 0
- bool success = false

9.13.1 Member Data Documentation

9.13.1.1 algorithm

```
std::string filevault::cli::AsymmetricBenchmarkResult::algorithm
```

9.13.1.2 decrypt_ms

```
double filevault::cli::AsymmetricBenchmarkResult::decrypt_ms = 0
```

9.13.1.3 encrypt_ms

```
double filevault::cli::AsymmetricBenchmarkResult::encrypt_ms = 0
```

9.13.1.4 keygen_ms

```
double filevault::cli::AsymmetricBenchmarkResult::keygen_ms = 0
```

9.13.1.5 success

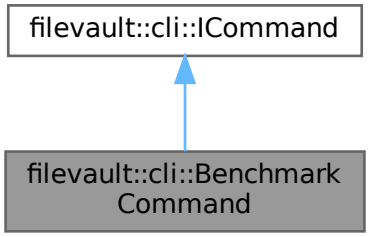
```
bool filevault::cli::AsymmetricBenchmarkResult::success = false
```

The documentation for this struct was generated from the following file:

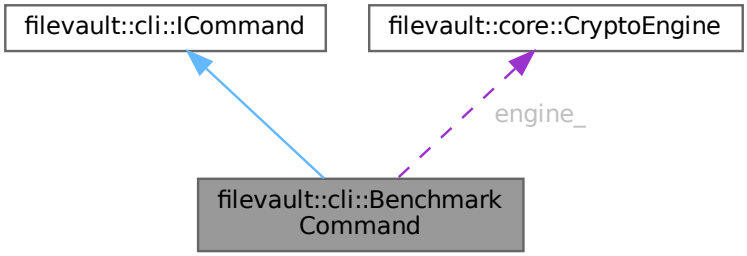
- include/filevault/cli/commands/benchmark_cmd.hpp

9.14 filevault::cli::BenchmarkCommand Class Reference

`#include <benchmark_cmd.hpp>`
Inheritance diagram for filevault::cli::BenchmarkCommand:



Collaboration diagram for filevault::cli::BenchmarkCommand:



Public Member Functions

- BenchmarkCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- void benchmark_symmetric (nlohmann::json &json_results)

- void benchmark_asymmetric (nlohmann::json &json_results)
- void benchmark_pqc (nlohmann::json &json_results)
- void benchmark_kdf (nlohmann::json &json_results)
- void benchmark_compression (nlohmann::json &json_results)
- void benchmark_hash (nlohmann::json &json_results)
- BenchmarkResult benchmark_algorithm (core::AlgorithmType algo_type)
- AsymmetricBenchmarkResult benchmark_asymmetric_algorithm (core::AlgorithmType algo_type)
- PQCBenchmarkResult benchmark_pqc_algorithm (core::AlgorithmType algo_type)
- BenchmarkResult benchmark_hybrid_algorithm (core::AlgorithmType algo_type)
- SignatureBenchmarkResult benchmark_signature_algorithm (core::AlgorithmType algo_type)
- std::string get_platform_info ()
- void save_json_output (const nlohmann::json &results)
- void save_log_output (const std::string &log_content)

Private Attributes

- core::CryptoEngine & engine_
- std::string algorithm_
- std::string output_file_
- size_t data_size_ = 1048576
- int iterations_ = 5
- bool all_ = false
- bool json_output_ = false
- bool pqc_only_ = false
- bool symmetric_only_ = false
- bool asymmetric_only_ = false
- bool hash_only_ = false
- bool kdf_only_ = false
- bool compression_only_ = false

9.14.1 Constructor & Destructor Documentation

9.14.1.1 BenchmarkCommand()

```
filevault::cli::BenchmarkCommand::BenchmarkCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.14.2 Member Function Documentation

9.14.2.1 benchmark_algorithm()

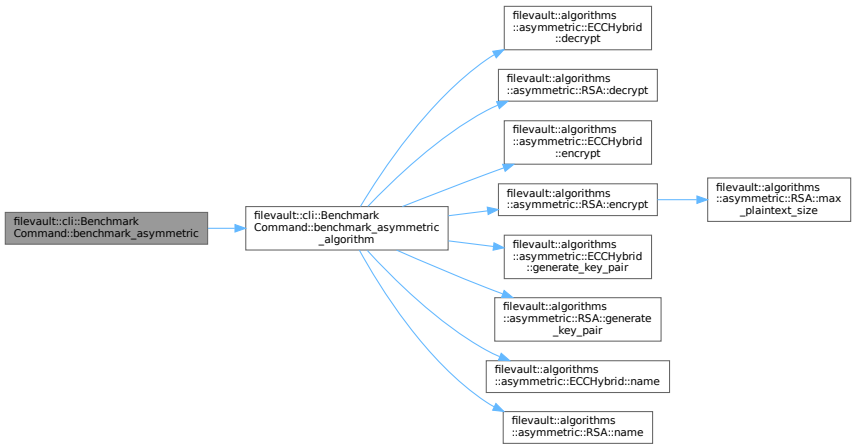
```
BenchmarkResult filevault::cli::BenchmarkCommand::benchmark_algorithm (
            core::AlgorithmType algo_type)  [private]
```

Here is the caller graph for this function:

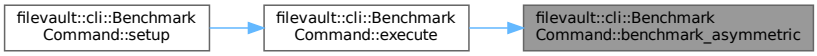
9.14.2.2 benchmark_asymmetric()

```
void filevault::cli::BenchmarkCommand::benchmark_asymmetric (
            nlohmann::json & json_results)  [private]
```

Here is the call graph for this function:



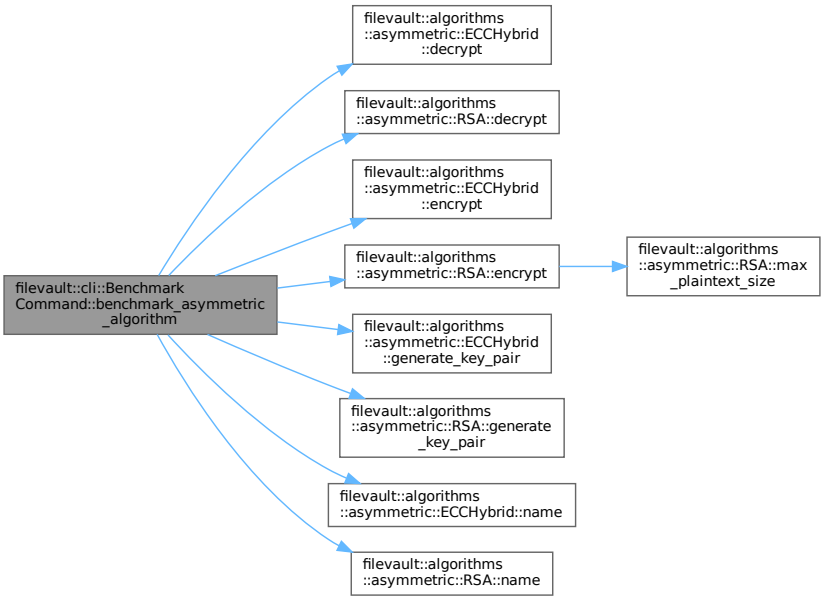
Here is the caller graph for this function:



9.14.2.3 benchmark_asymmetric_algorithm()

```
AsymmetricBenchmarkResult filevault::cli::BenchmarkCommand::benchmark_asymmetric_algorithm (
            core::AlgorithmType algo_type)  [private]
```

Here is the call graph for this function:

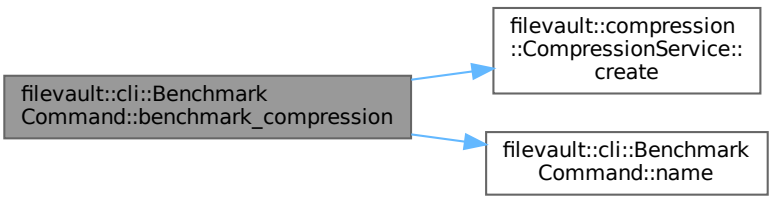


Here is the caller graph for this function:

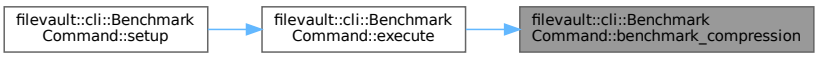
9.14.2.4 benchmark_compression()

```
void filevault::cli::BenchmarkCommand::benchmark_compression (
            nlohmann::json & json_results)  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:

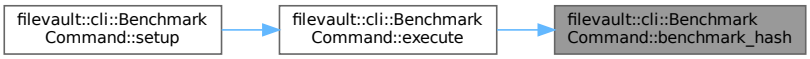


9.14.2.5 benchmark_hash()

```
void filevault::cli::BenchmarkCommand::benchmark_hash (
            nlohmann::json & json_results)  [private]
```

Here is the call graph for this function:

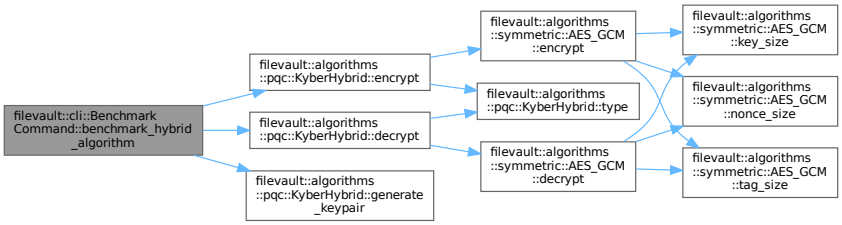
Here is the caller graph for this function:



9.14.2.6 benchmark_hybrid_algorithm()

```
BenchmarkResult filevault::cli::BenchmarkCommand::benchmark_hybrid_algorithm (
            core::AlgorithmType algo_type)  [private]
```

Here is the call graph for this function:

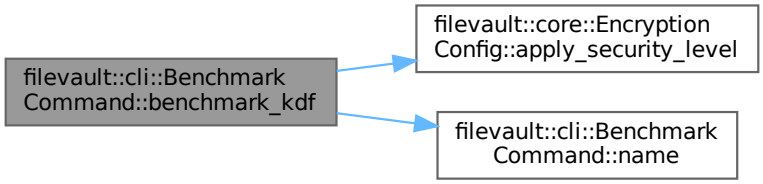


Here is the caller graph for this function:

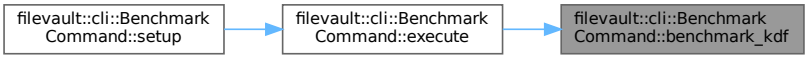
9.14.2.7 benchmark_kdf()

```
void filevault::cli::BenchmarkCommand::benchmark_kdf (
            nlohmann::json & json_results)  [private]
```

Here is the call graph for this function:



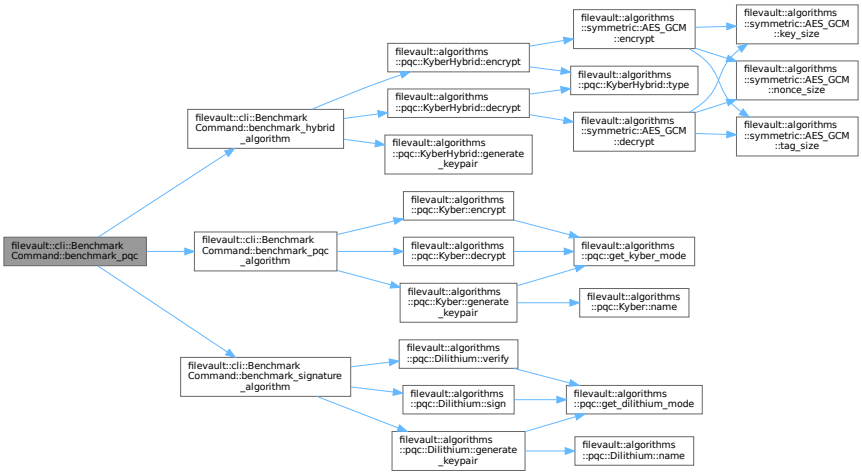
Here is the caller graph for this function:



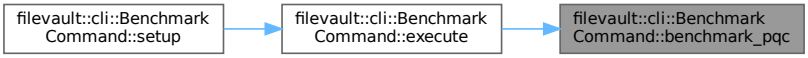
9.14.2.8 benchmark_pqc()

```
void filevault::cli::BenchmarkCommand::benchmark_pqc (
            nlohmann::json & json_results)  [private]
```

Here is the call graph for this function:



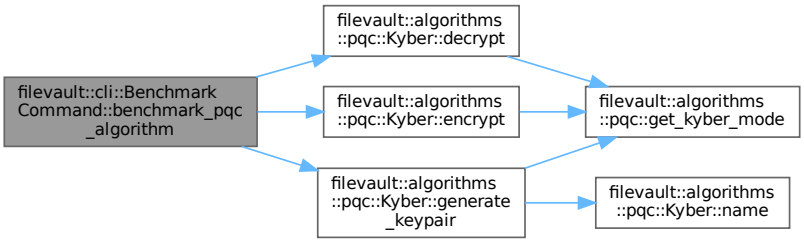
Here is the caller graph for this function:



9.14.2.9 benchmark_pqc_algorithm()

```
PQCBenchmarkResult filevault::cli::BenchmarkCommand::benchmark_pqc_algorithm (
            core::AlgorithmType algo_type)  [private]
```

Here is the call graph for this function:

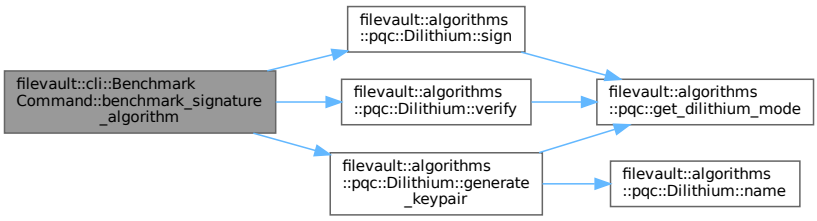


Here is the caller graph for this function:

9.14.2.10 benchmark_signature_algorithm()

```
SignatureBenchmarkResult filevault::cli::BenchmarkCommand::benchmark_signature_algorithm (
            core::AlgorithmType algo_type)  [private]
```

Here is the call graph for this function:

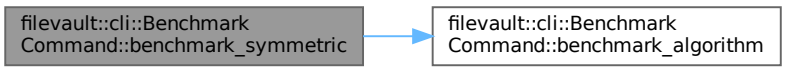


Here is the caller graph for this function:

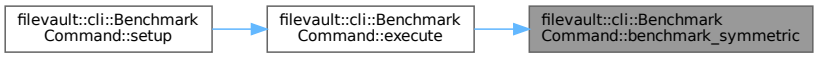
9.14.2.11 benchmark_symmetric()

```
void filevault::cli::BenchmarkCommand::benchmark_symmetric (
            nlohmann::json & json_results)  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



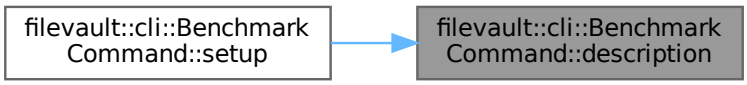
9.14.2.12 description()

```
std::string filevault::cli::BenchmarkCommand::description () const  [inline], [override],
[virtual]
```

Get command description.

Implements filevault::cli::ICommand.

Here is the caller graph for this function:



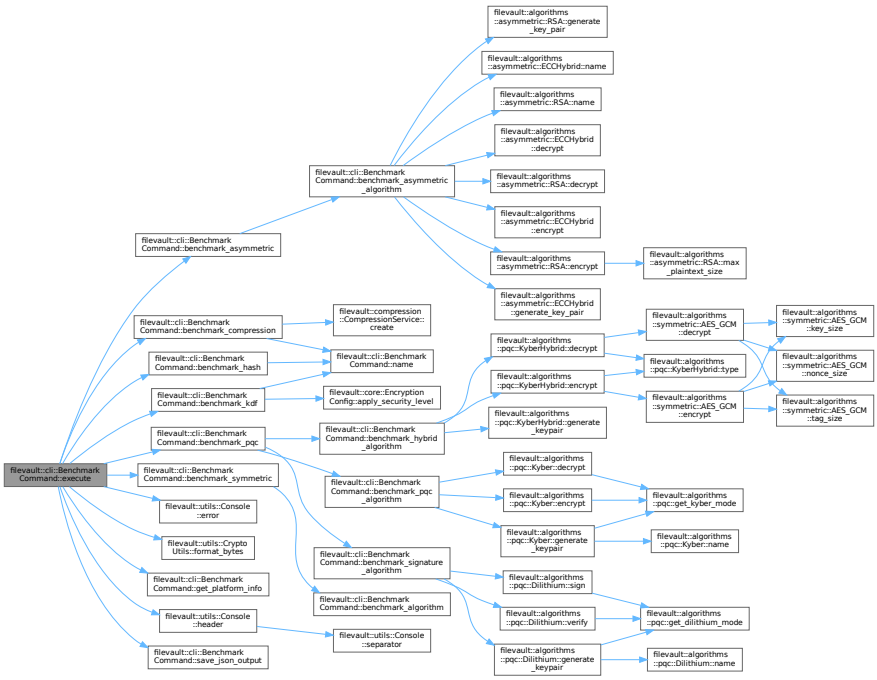
9.14.2.13 execute()

`int filevault::cli::BenchmarkCommand::execute () [override], [virtual]`

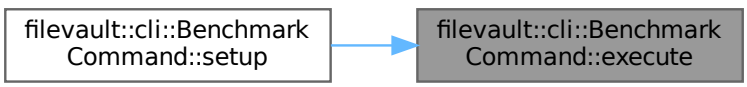
Execute the command.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



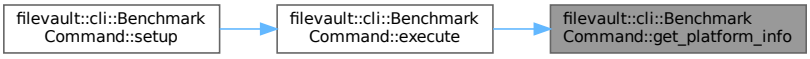
Here is the caller graph for this function:



9.14.2.14 get_platform_info()

```
std::string filevault::cli::BenchmarkCommand::get_platform_info ()  [private]
```

Here is the caller graph for this function:



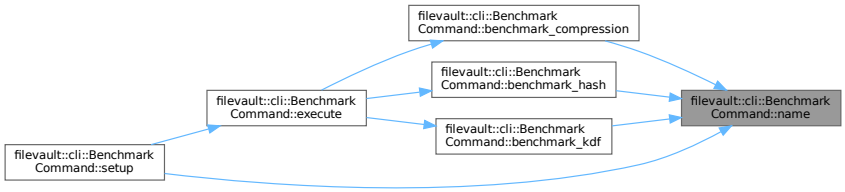
9.14.2.15 name()

```
std::string filevault::cli::BenchmarkCommand::name () const  [inline], [override], [virtual]
```

Get command name.

Implements filevault::cli::ICommand.

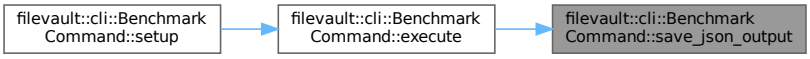
Here is the caller graph for this function:



9.14.2.16 save_json_output()

```
void filevault::cli::BenchmarkCommand::save_json_output (
            const nlohmann::json & results)  [private]
```

Here is the caller graph for this function:



9.14.2.17 save_log_output()

```
void filevault::cli::BenchmarkCommand::save_log_output (
            const std::string & log_content)  [private]
```

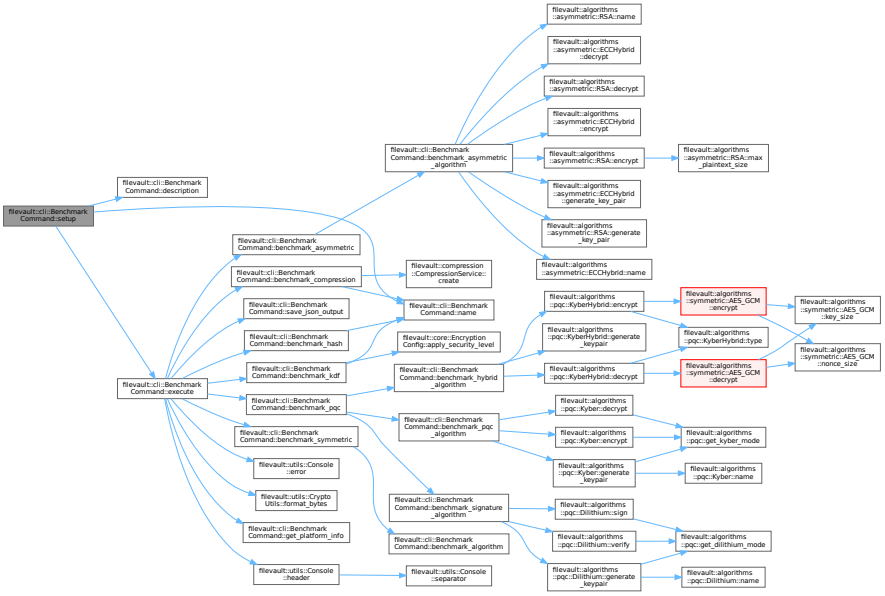
9.14.2.18 setup()

```
void filevault::cli::BenchmarkCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



9.14.3 Member Data Documentation

9.14.3.1 algorithm_

```
std::string filevault::cli::BenchmarkCommand::algorithm_  [private]
```

9.14.3.2 all_

```
bool filevault::cli::BenchmarkCommand::all_ = false  [private]
```

9.14.3.3 asymmetric_only_

```
bool filevault::cli::BenchmarkCommand::asymmetric_only_ = false  [private]
```

9.14.3.4 compression_only_

```
bool filevault::cli::BenchmarkCommand::compression_only_ = false  [private]
```

9.14.3.5 data_size_

```
size_t filevault::cli::BenchmarkCommand::data_size_ = 1048576  [private]
```

9.14.3.6 engine_

```
core::CryptoEngine& filevault::cli::BenchmarkCommand::engine_  [private]
```

9.14.3.7 hash_only_

```
bool filevault::cli::BenchmarkCommand::hash_only_ = false  [private]
```

9.14.3.8 iterations_

```
int filevault::cli::BenchmarkCommand::iterations_ = 5  [private]
```

9.14.3.9 json_output_

```
bool filevault::cli::BenchmarkCommand::json_output_ = false  [private]
```

9.14.3.10 kdf_only_

```
bool filevault::cli::BenchmarkCommand::kdf_only_ = false  [private]
```

9.14.3.11 output_file_

```
std::string filevault::cli::BenchmarkCommand::output_file_  [private]
```

9.14.3.12 pqc_only_

```
bool filevault::cli::BenchmarkCommand::pqc_only_ = false  [private]
```

9.14.3.13 symmetric_only_

```
bool filevault::cli::BenchmarkCommand::symmetric_only_ = false  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/benchmark_cmd.hpp
- src/cli/commands/benchmark_cmd.cpp

9.15 filevault::cli::BenchmarkResult Struct Reference

```
#include <benchmark_cmd.hpp>
```

Public Attributes

- std::string algorithm
- double encrypt_ms = 0
- double decrypt_ms = 0
- double encrypt_mbps = 0
- double decrypt_mbps = 0
- bool success = false

9.15.1 Member Data Documentation

9.15.1.1 algorithm

```
std::string filevault::cli::BenchmarkResult::algorithm
```

9.15.1.2 decrypt_mbps

```
double filevault::cli::BenchmarkResult::decrypt_mbps = 0
```

9.15.1.3 decrypt_ms

```
double filevault::cli::BenchmarkResult::decrypt_ms = 0
```

9.15.1.4 encrypt_mbps

```
double filevault::cli::BenchmarkResult::encrypt_mbps = 0
```

9.15.1.5 encrypt_ms

```
double filevault::cli::BenchmarkResult::encrypt_ms = 0
```

9.15.1.6 success

```
bool filevault::cli::BenchmarkResult::success = false
```

The documentation for this struct was generated from the following file:

- include/filevault/cli/commands/benchmark_cmd.hpp

9.16 filevault::utils::BlockProgressBar Class Reference

Block-style progress bar.

`#include <progress.hpp>`

Public Member Functions

- BlockProgressBar (const std::string &prefix, size_t max_progress=100)
- void set_progress (size_t progress)
- void set_option_text (const std::string &text)
- void mark_as_completed ()

Private Attributes

- std::unique_ptr< indicators::BlockProgressBar > bar_

9.16.1 Detailed Description

Block-style progress bar.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 BlockProgressBar()

```
filevault::utils::BlockProgressBar::BlockProgressBar (
            const std::string & prefix,
            size_t max_progress = 100)
```

9.16.3 Member Function Documentation

9.16.3.1 mark_as_completed()

```
void filevault::utils::BlockProgressBar::mark_as_completed ()
```

9.16.3.2 set_option_text()

```
void filevault::utils::BlockProgressBar::set_option_text (
            const std::string & text)
```

9.16.3.3 set_progress()

```
void filevault::utils::BlockProgressBar::set_progress (
            size_t progress)
```

9.16.4 Member Data Documentation

9.16.4.1 bar_

```
std::unique_ptr<indicators::BlockProgressBar> filevault::utils::BlockProgressBar::bar_  [private]
```

The documentation for this class was generated from the following files:

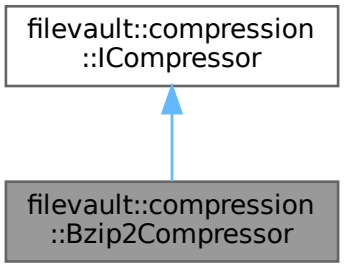
- include/filevault/utils/progress.hpp
- src/utils/progress.cpp

9.17 filevault::compression::Bzip2Compressor Class Reference

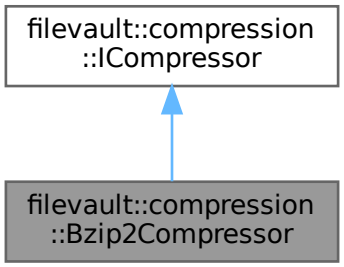
BZIP2 compressor (better ratio, slower).

`#include <compressor.hpp>`

Inheritance diagram for filevault::compression::Bzip2Compressor:



Collaboration diagram for filevault::compression::Bzip2Compressor:



Public Member Functions

- std::string name () const override

  Get compressor name.
- CompressionResult compress (std::span< const uint8_t > input, int level=6) override

  Compress data.
- CompressionResult decompress (std::span< const uint8_t > input) override

  Decompress data.

Public Member Functions inherited from filevault::compression::ICompressor

- virtual ~ICompressor ()=default

9.17.1 Detailed Description

BZIP2 compressor (better ratio, slower).

9.17.2 Member Function Documentation

9.17.2.1 compress()

```
CompressionResult filevault::compression::Bzip2Compressor::compress (
            std::span< const uint8_t > input,
            int level = 6)  [override], [virtual]
```

Compress data.

Parameters

input	Data to compress
level	Compression level (1-9, algorithm-specific)

Returns

Compressed data

Implements filevault::compression::ICompressor.

9.17.2.2 decompress()

```
CompressionResult filevault::compression::Bzip2Compressor::decompress (
            std::span< const uint8_t > input)  [override], [virtual]
```

Decompress data.

Parameters

input	Compressed data

Returns

Decompressed data

Implements filevault::compression::ICompressor.

9.17.2.3 name()

```
std::string filevault::compression::Bzip2Compressor::name () const  [inline], [override],
[virtual]
```

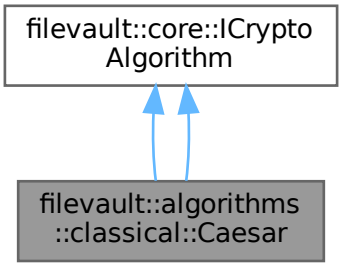
Get compressor name.
Implements filevault::compression::ICompressor.
The documentation for this class was generated from the following files:

- include/filevault/compression/compressor.hpp
- src/compression/compressor.cpp

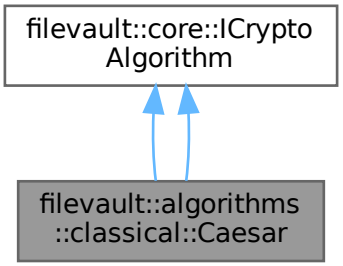
9.18 filevault::algorithms::classical::Caesar Class Reference

Caesar Cipher - Educational only.
`#include <caesar.hpp>`

Inheritance diagram for filevault::algorithms::classical::Caesar:



Collaboration diagram for filevault::algorithms::classical::Caesar:



Public Member Functions

- Caesar (int shift=3)
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- size_t key_size () const override

  Get recommended key size in bytes.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.
- Caesar (int shift=3)
- std::string name () const override

Get algorithm name.

- core::AlgorithmType type () const override

  Get algorithm type.

- size_t key_size () const override

  Get recommended key size in bytes.

- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.

- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.

- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Static Public Member Functions

- static std::string brute_force (const std::string &ciphertext)

  Brute force attack demonstration.

- static std::string brute_force (const std::string &ciphertext)
- static double frequency_analysis (const std::string &text)

Private Member Functions

- char shift_char (char ch, int shift) const
- char shift_char (char ch, int shift) const

Private Attributes

- int shift_

9.18.1 Detailed Description

Caesar Cipher - Educational only.
Simple substitution cipher that shifts characters by a fixed offset. Used by Julius Caesar for military messages (~50 BC).
Security: COMPLETELY BROKEN - Only 26 possible keys! Purpose: Educational demonstration of cryptanalysis

See also

https://en.wikipedia.org/wiki/Caesar_cipher

Simple substitution cipher that shifts characters by a fixed offset. Used by Julius Caesar for military messages (~50 BC).
Security: COMPLETELY BROKEN - Only 26 possible keys! Purpose: Educational demonstration of cryptanalysis

9.18.2 Constructor & Destructor Documentation

9.18.2.1 Caesar() [1/2]

```
filevault::algorithms::classical::Caesar::Caesar (
            int shift = 3)  [explicit]
```

9.18.2.2 Caesar() [2/2]

```
filevault::algorithms::classical::Caesar::Caesar (
            int shift = 3)  [explicit]
```

9.18.3 Member Function Documentation

9.18.3.1 brute_force() [1/2]

```
std::string filevault::algorithms::classical::Caesar::brute_force (
            const std::string & ciphertext)  [static]
```

Brute force attack demonstration.

Parameters

ciphertext	Encrypted text

Returns

All 26 possible decryptions

9.18.3.2 brute_force() [2/2]

```
std::string filevault::algorithms::classical::Caesar::brute_force (
            const std::string & ciphertext)  [static]
```

9.18.3.3 decrypt() [1/2]

```
core::CryptoResult filevault::algorithms::classical::Caesar::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

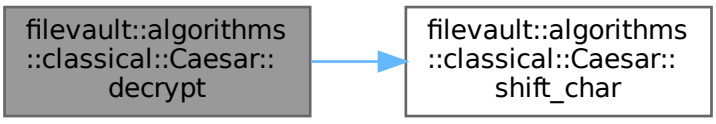
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.18.3.4 decrypt() [2/2]

```
core::CryptoResult filevault::algorithms::classical::Caesar::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

9.18.3.5 encrypt() [1/2]

```
core::CryptoResult filevault::algorithms::classical::Caesar::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

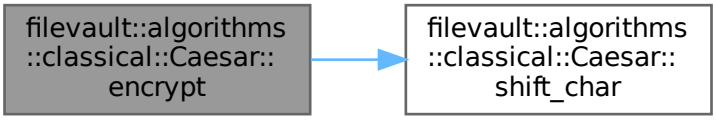
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.18.3.6 encrypt() [2/2]

```
core::CryptoResult filevault::algorithms::classical::Caesar::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.

9.18.3.7 frequency_analysis()

```
double filevault::algorithms::classical::Caesar::frequency_analysis (
            const std::string & text)  [static]
```

9.18.3.8 is_suitable_for() [1/2]

```
bool filevault::algorithms::classical::Caesar::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.18.3.9 is_suitable_for() [2/2]

```
bool filevault::algorithms::classical::Caesar::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.18.3.10 key_size() [1/2]

```
size_t filevault::algorithms::classical::Caesar::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.18.3.11 key_size() [2/2]

```
size_t filevault::algorithms::classical::Caesar::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.18.3.12 name() [1/2]

```
std::string filevault::algorithms::classical::Caesar::name () const  [inline], [override],
[virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.18.3.13 name() [2/2]

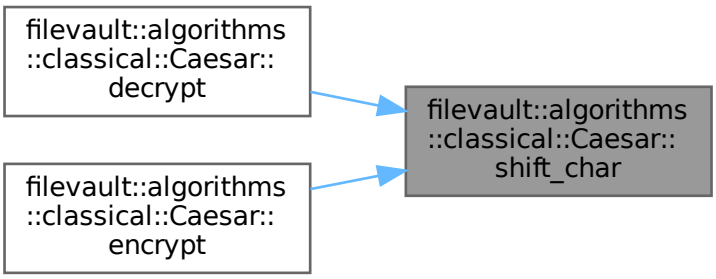
```
std::string filevault::algorithms::classical::Caesar::name () const  [inline], [override],
[virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.18.3.14 shift_char() [1/2]

```
char filevault::algorithms::classical::Caesar::shift_char (
            char ch,
            int shift) const  [private]
```

Here is the caller graph for this function:



9.18.3.15 shift_char() [2/2]

```
char filevault::algorithms::classical::Caesar::shift_char (
            char ch,
            int shift) const  [private]
```

9.18.3.16 type() [1/2]

```
core::AlgorithmType filevault::algorithms::classical::Caesar::type () const  [inline], [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.18.3.17 type() [2/2]

```
core::AlgorithmType filevault::algorithms::classical::Caesar::type () const  [inline], [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.18.4 Member Data Documentation

9.18.4.1 shift_

```
int filevault::algorithms::classical::Caesar::shift_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/classical/caesar.hpp

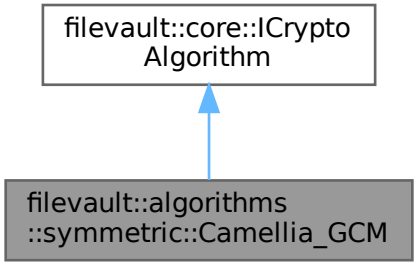
- include/filevault/algorithms/classical/classical_ciphers.hpp
- src/algorithms/classical/caesar.cpp
- src/algorithms/classical/classical_ciphers.cpp

9.19 filevault::algorithms::symmetric::Camellia_GCM Class Reference

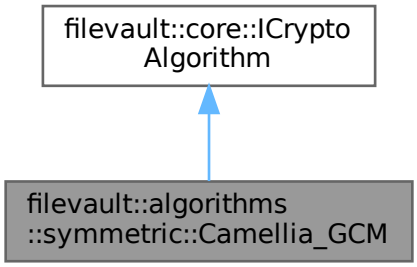
Camellia-GCM AEAD encryption.

`#include <camellia_gcm.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::Camellia_GCM:



Collaboration diagram for filevault::algorithms::symmetric::Camellia_GCM:



Public Member Functions

- Camellia_GCM (size_t key_bits=256)

  Construct Camellia-GCM with specified key size.
- virtual ~Camellia_GCM ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

Encrypt data.

- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.

- size_t key_size () const override

  Get recommended key size in bytes.

- size_t nonce_size () const
- size_t tag_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.19.1 Detailed Description

Camellia-GCM AEAD encryption.
Supports 128, 192, and 256-bit keys. Uses GCM mode for authenticated encryption.

9.19.2 Constructor & Destructor Documentation

9.19.2.1 Camellia_GCM()

```
filevault::algorithms::symmetric::Camellia_GCM::Camellia_GCM (
            size_t key_bits = 256)  [explicit]
```

Construct Camellia-GCM with specified key size.

Parameters

key_bits	Key size in bits (128, 192, or 256)

9.19.2.2 ~Camellia_GCM()

```
virtual filevault::algorithms::symmetric::Camellia_GCM::~Camellia_GCM ()  [virtual], [default]
```

9.19.3 Member Function Documentation

9.19.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::Camellia_GCM::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

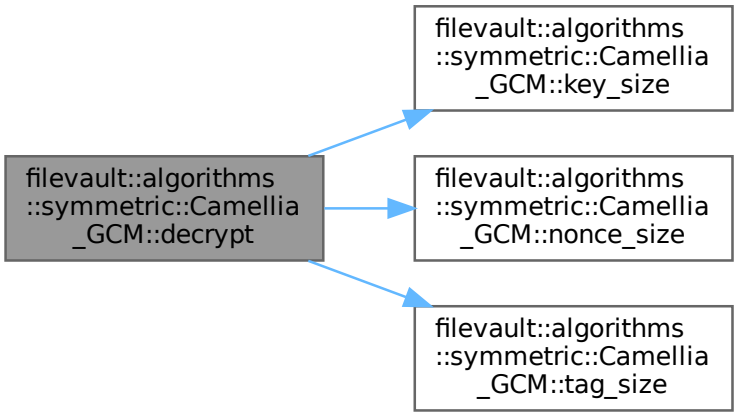
ciphertext	Encrypted data

key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.19.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::Camellia_GCM::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

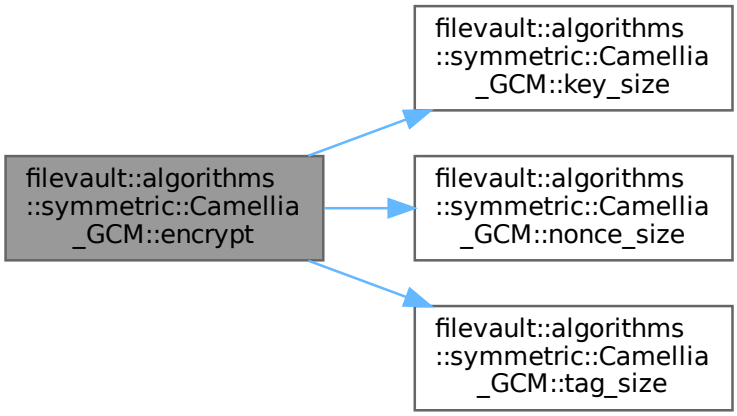
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.19.3.3 is_suitable_for()

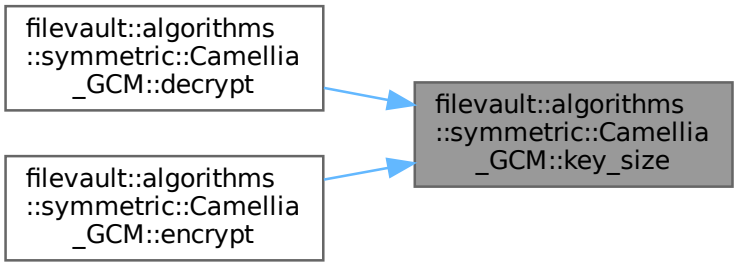
```
bool filevault::algorithms::symmetric::Camellia_GCM::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.19.3.4 key_size()

```
size_t filevault::algorithms::symmetric::Camellia_GCM::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.19.3.5 name()

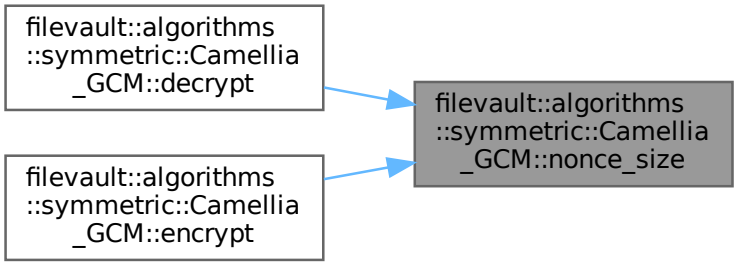
```
std::string filevault::algorithms::symmetric::Camellia_GCM::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.19.3.6 nonce_size()

```
size_t filevault::algorithms::symmetric::Camellia_GCM::nonce_size () const  [inline]
```

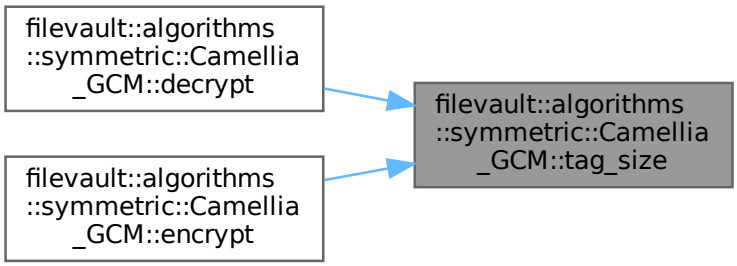
Here is the caller graph for this function:



9.19.3.7 tag_size()

```
size_t filevault::algorithms::symmetric::Camellia_GCM::tag_size () const  [inline]
```

Here is the caller graph for this function:



9.19.3.8 type()

```
core::AlgorithmType filevault::algorithms::symmetric::Camellia_GCM::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.19.4 Member Data Documentation

9.19.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::Camellia_GCM::botan_name_  [private]
```

9.19.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::Camellia_GCM::key_bits_  [private]
```

9.19.4.3 type_

```
core::AlgorithmType filevault::algorithms::symmetric::Camellia_GCM::type_  [private]
```

The documentation for this class was generated from the following files:

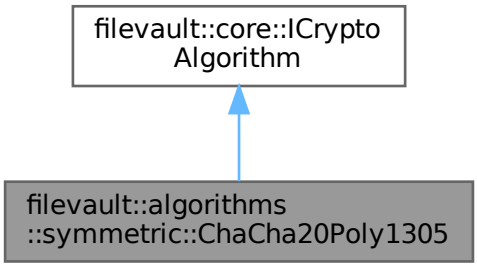
- include/filevault/algorithms/symmetric/camellia_gcm.hpp
- src/algorithms/symmetric/camellia_gcm.cpp

9.20 filevault::algorithms::symmetric::ChaCha20Poly1305 Class Reference

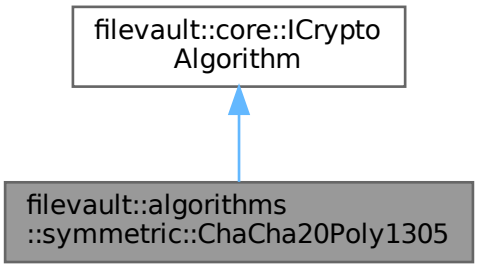
ChaCha20-Poly1305 AEAD encryption.

`#include <chacha20_poly1305.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::ChaCha20Poly1305:



Collaboration diagram for filevault::algorithms::symmetric::ChaCha20Poly1305:



Public Member Functions

- ChaCha20Poly1305 ()
- virtual ~ChaCha20Poly1305 ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t nonce_size () const
- size_t tag_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

9.20.1 Detailed Description

ChaCha20-Poly1305 AEAD encryption.
ChaCha20-Poly1305 is a modern AEAD cipher combining:

- ChaCha20 stream cipher (designed by Daniel J. Bernstein)
- Poly1305 MAC for authentication

Benefits over AES-GCM:

- No need for AES-NI hardware acceleration
- Constant-time implementation easier to achieve
- Better performance on software-only platforms
- Recommended by IETF (RFC 8439)

Security:

- 256-bit key
- 96-bit nonce (12 bytes) - MUST be unique per encryption
- 128-bit authentication tag (16 bytes)
- Provides confidentiality and authenticity

9.20.2 Constructor & Destructor Documentation

9.20.2.1 ChaCha20Poly1305()

```
filevault::algorithms::symmetric::ChaCha20Poly1305::ChaCha20Poly1305 ()
```

9.20.2.2 ~ChaCha20Poly1305()

```
virtual filevault::algorithms::symmetric::ChaCha20Poly1305::~ChaCha20Poly1305 ()  [virtual],
[default]
```

9.20.3 Member Function Documentation

9.20.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::ChaCha20Poly1305::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

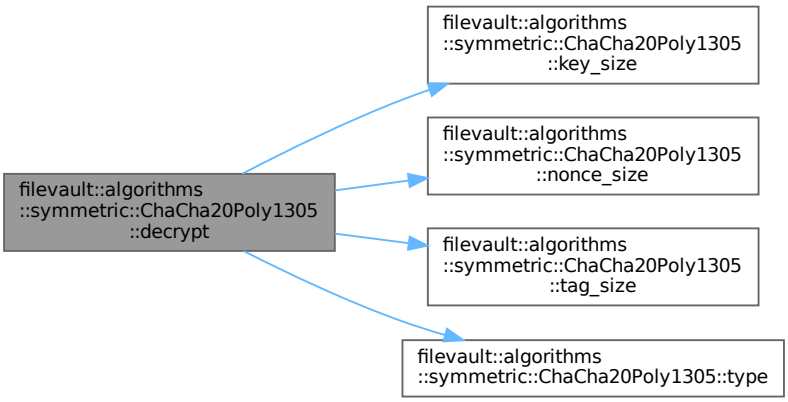
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.20.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::ChaCha20Poly1305::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

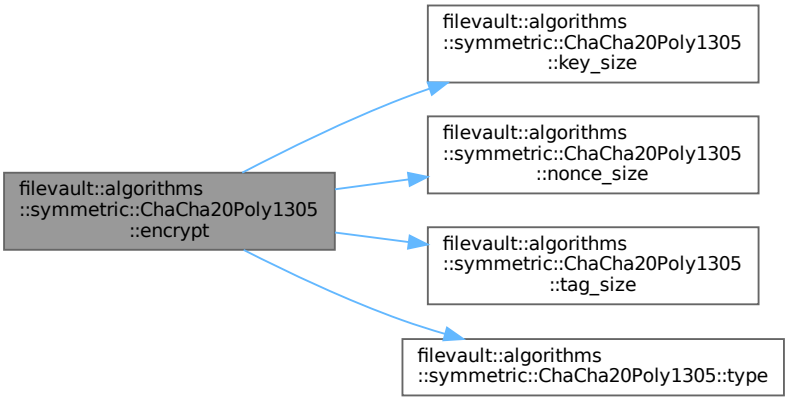
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.20.3.3 is_suitable_for()

```
bool filevault::algorithms::symmetric::ChaCha20Poly1305::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

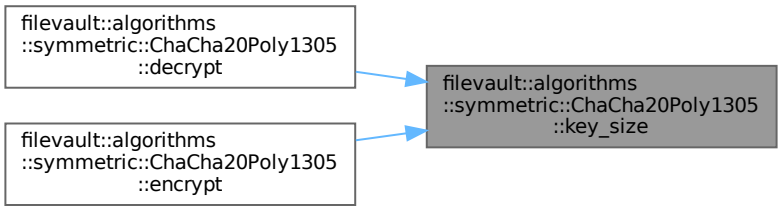
Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.20.3.4 key_size()

```
size_t filevault::algorithms::symmetric::ChaCha20Poly1305::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



9.20.3.5 name()

`std::string filevault::algorithms::symmetric::ChaCha20Poly1305::name () const [override], [virtual]`

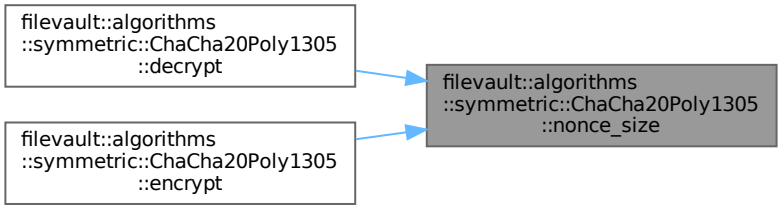
Get algorithm name.

Implements filevault::core::ICryptoAlgorithm.

9.20.3.6 nonce_size()

`size_t filevault::algorithms::symmetric::ChaCha20Poly1305::nonce_size () const [inline]`

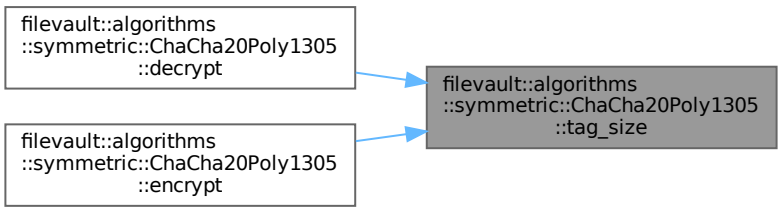
Here is the caller graph for this function:



9.20.3.7 tag_size()

`size_t filevault::algorithms::symmetric::ChaCha20Poly1305::tag_size () const [inline]`

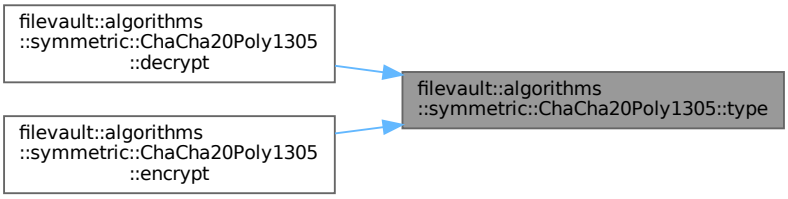
Here is the caller graph for this function:



9.20.3.8 type()

```
core::AlgorithmType filevault::algorithms::symmetric::ChaCha20Poly1305::type () const  [override], [virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/chacha20_poly1305.hpp
- src/algorithms/symmetric/chacha20_poly1305.cpp

9.21 filevault::core::ChunkInfo Struct Reference

Chunk information for streaming encryption.
`#include <streaming.hpp>`

Public Attributes

- size_t chunk_index
- size_t chunk_size
- size_t total_chunks
- size_t bytes_processed
- size_t total_bytes

9.21.1 Detailed Description

Chunk information for streaming encryption.

9.21.2 Member Data Documentation

9.21.2.1 bytes_processed

```
size_t filevault::core::ChunkInfo::bytes_processed
```

9.21.2.2 chunk_index

```
size_t filevault::core::ChunkInfo::chunk_index
```

9.21.2.3 chunk_size

```
size_t filevault::core::ChunkInfo::chunk_size
```

9.21.2.4 total_bytes

```
size_t filevault::core::ChunkInfo::total_bytes
```

9.21.2.5 total_chunks

`size_t filevault::core::ChunkInfo::total_chunks`

The documentation for this struct was generated from the following file:

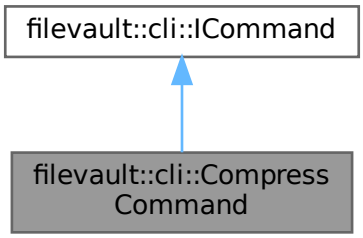
- include/filevault/core/streaming.hpp

9.22 filevault::cli::CompressCommand Class Reference

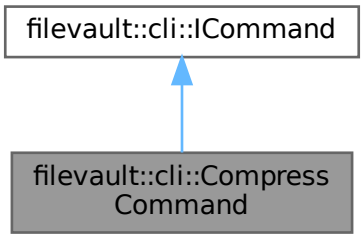
Compress command - standalone compression/decompression.

`#include <compress_cmd.hpp>`

Inheritance diagram for filevault::cli::CompressCommand:



Collaboration diagram for filevault::cli::CompressCommand:



Public Member Functions

- CompressCommand ()=default
- ~CompressCommand () override=default
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- int do_compress ()
- int do_decompress ()
- std::string detect_algorithm (const std::string &path)
- std::string generate_output_path (const std::string &input, bool compressing)

Private Attributes

- CLI::App * subcommand_ = nullptr
- std::string input_file_
- std::string output_file_
- std::string algorithm_ = "zlib"
- int level_ = 6
- bool decompress_ = false
- bool verbose_ = false
- bool benchmark_ = false
- bool auto_detect_ = false

9.22.1 Detailed Description

Compress command - standalone compression/decompression.
Compress or decompress files without encryption. Supports: zlib, bzip2, lzma
Examples: filevault compress bigfile.log -a zlib -l 9 -o bigfile.zlib filevault compress –decompress bigfile.zlib -o bigfile.log

9.22.2 Constructor & Destructor Documentation

9.22.2.1 CompressCommand()

```
filevault::cli::CompressCommand::CompressCommand ()  [default]
```

9.22.2.2 ~CompressCommand()

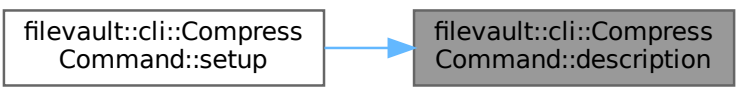
```
filevault::cli::CompressCommand::~CompressCommand ()  [override], [default]
```

9.22.3 Member Function Documentation

9.22.3.1 description()

```
std::string filevault::cli::CompressCommand::description () const  [inline], [override], [virtual]
```

Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:



9.22.3.2 detect_algorithm()

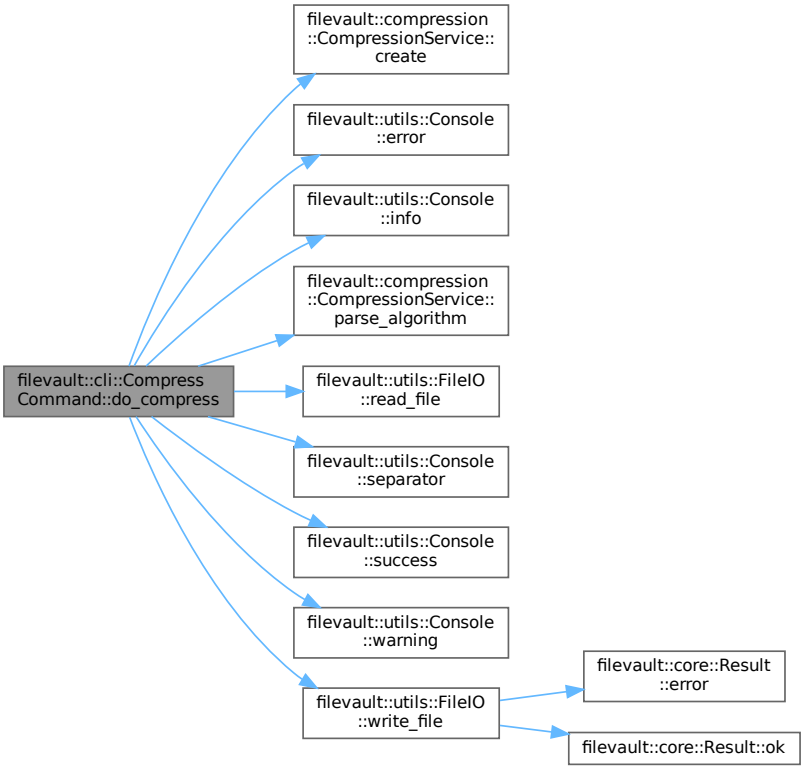
```
std::string filevault::cli::CompressCommand::detect_algorithm (
            const std::string & path)  [private]
```

Here is the caller graph for this function:

9.22.3.3 do_compress()

```
int filevault::cli::CompressCommand::do_compress ()  [private]
```

Here is the call graph for this function:

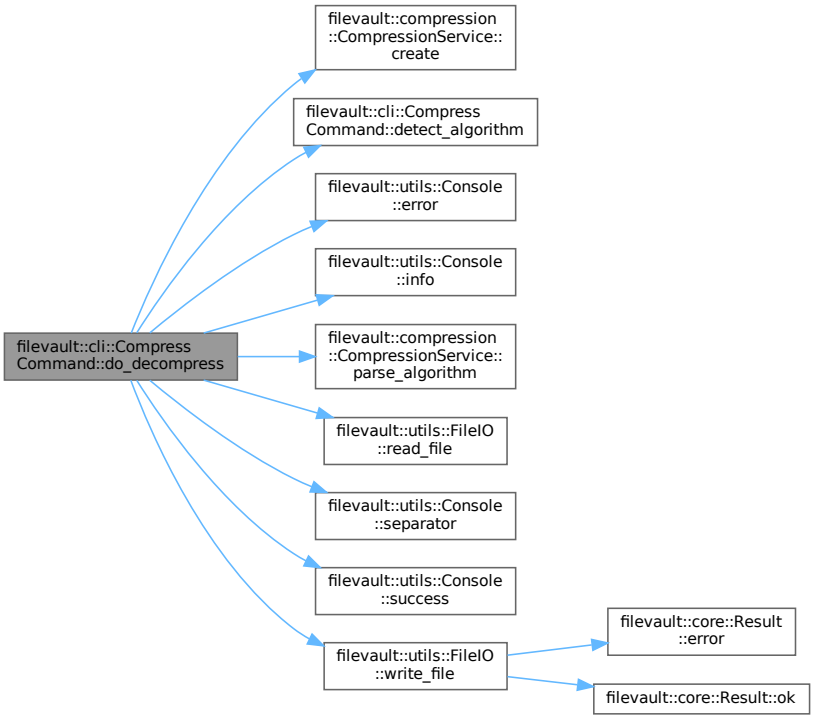


Here is the caller graph for this function:

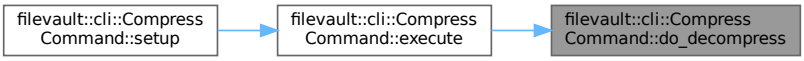
9.22.3.4 do_decompress()

```
int filevault::cli::CompressCommand::do_decompress ()  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



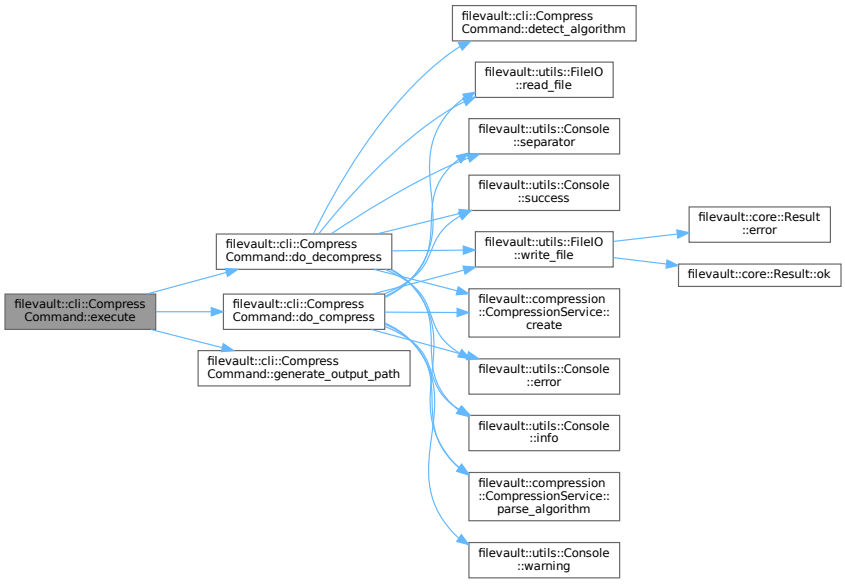
9.22.3.5 execute()

```
int filevault::cli::CompressCommand::execute ()  [override], [virtual]
```

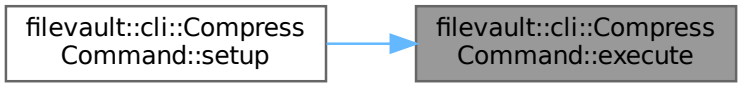
Execute the command.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



9.22.3.6 generate_output_path()

```
std::string filevault::cli::CompressCommand::generate_output_path (
            const std::string & input,
            bool compressing)  [private]
```

Here is the caller graph for this function:

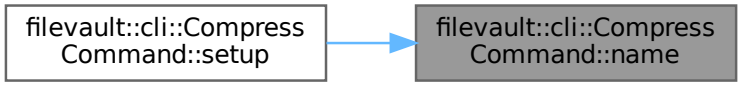
9.22.3.7 name()

```
std::string filevault::cli::CompressCommand::name () const  [inline], [override], [virtual]
```

Get command name.

Implements filevault::cli::ICommand.

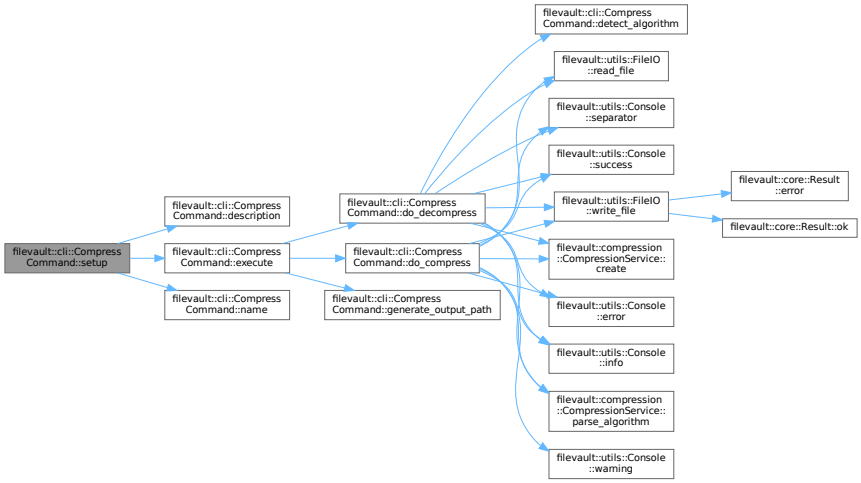
Here is the caller graph for this function:



9.22.3.8 setup()

```
void filevault::cli::CompressCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



9.22.4 Member Data Documentation

9.22.4.1 algorithm_

```
std::string filevault::cli::CompressCommand::algorithm_ = "zlib"  [private]
```

9.22.4.2 auto_detect_

```
bool filevault::cli::CompressCommand::auto_detect_ = false  [private]
```

9.22.4.3 benchmark_

```
bool filevault::cli::CompressCommand::benchmark_ = false  [private]
```

9.22.4.4 decompress_

```
bool filevault::cli::CompressCommand::decompress_ = false  [private]
```

9.22.4.5 input_file_

```
std::string filevault::cli::CompressCommand::input_file_  [private]
```

9.22.4.6 level_

```
int filevault::cli::CompressCommand::level_ = 6  [private]
```

9.22.4.7 output_file_

```
std::string filevault::cli::CompressCommand::output_file_  [private]
```

9.22.4.8 subcommand_

```
CLI::App* filevault::cli::CompressCommand::subcommand_ = nullptr  [private]
```

9.22.4.9 verbose_

```
bool filevault::cli::CompressCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/compress_cmd.hpp
- src/cli/commands/compress_cmd.cpp

9.23 filevault::compression::CompressionResult Struct Reference

Compression result.

`#include <compressor.hpp>`

Public Attributes

- bool success = false
- std::string error_message
- std::vector< uint8_t > data
- size_t original_size = 0
- size_t compressed_size = 0
- double compression_ratio = 0.0
- double processing_time_ms = 0.0

9.23.1 Detailed Description

Compression result.

9.23.2 Member Data Documentation

9.23.2.1 compressed_size

```
size_t filevault::compression::CompressionResult::compressed_size = 0
```

9.23.2.2 compression_ratio

```
double filevault::compression::CompressionResult::compression_ratio = 0.0
```

9.23.2.3 data

```
std::vector<uint8_t> filevault::compression::CompressionResult::data
```

9.23.2.4 error_message

```
std::string filevault::compression::CompressionResult::error_message
```

9.23.2.5 original_size

```
size_t filevault::compression::CompressionResult::original_size = 0
```

9.23.2.6 processing_time_ms

```
double filevault::compression::CompressionResult::processing_time_ms = 0.0
```

9.23.2.7 success

```
bool filevault::compression::CompressionResult::success = false
```

The documentation for this struct was generated from the following file:

- include/filevault/compression/compressor.hpp

9.24 filevault::compression::CompressionService Class Reference

Compression service - factory for compressors.

```
#include <compressor.hpp>
```

Static Public Member Functions

- static std::unique_ptr< ICompressor > create (core::CompressionType type)

  Create compressor for algorithm.
- static std::string get_algorithm_name (core::CompressionType type)

  Get algorithm name.
- static core::CompressionType parse_algorithm (const std::string &name)

  Parse algorithm from string.

9.24.1 Detailed Description

Compression service - factory for compressors.

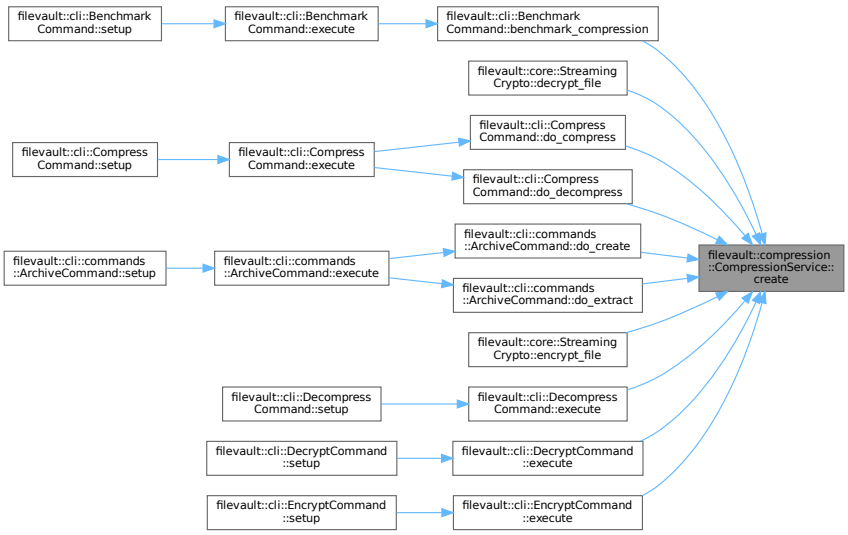
9.24.2 Member Function Documentation

9.24.2.1 create()

```
std::unique_ptr< ICompressor > filevault::compression::CompressionService::create (
            core::CompressionType type)  [static]
```

Create compressor for algorithm.

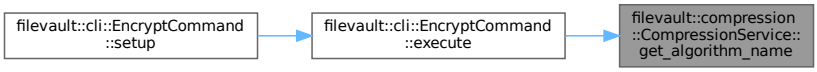
Here is the caller graph for this function:



9.24.2.2 get_algorithm_name()

```
std::string filevault::compression::CompressionService::get_algorithm_name (
            core::CompressionType type)  [static]
```

Get algorithm name.
Here is the caller graph for this function:

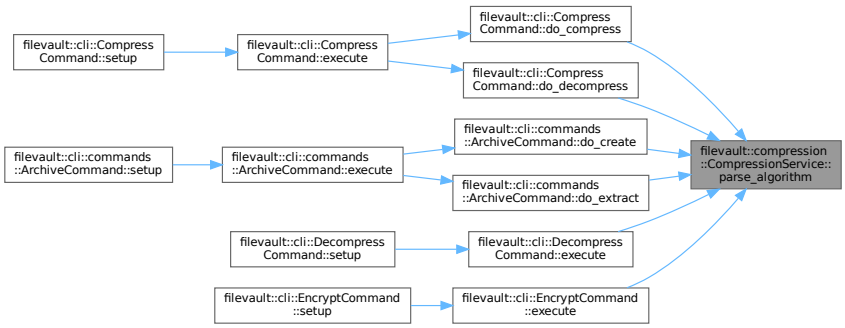


9.24.2.3 parse_algorithm()

```
core::CompressionType filevault::compression::CompressionService::parse_algorithm (
            const std::string & name)  [static]
```

Parse algorithm from string.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/compression/compressor.hpp
- src/compression/compressor.cpp

9.25 filevault::utils::Config Class Reference

Configuration manager for FileVault.

`#include <config.hpp>`

Public Member Functions

- bool save () const

  Save configuration to file.
- void reset ()

  Reset to defaults.
- std::string get_default_mode () const
- std::string get_default_algorithm () const
- std::string get_default_kdf () const
- std::string get_default_compression () const
- int get_compression_level () const
- bool get_show_progress () const
- bool get_verbose () const
- void set_default_mode (const std::string &mode)
- void set_default_algorithm (const std::string &algo)
- void set_default_kdf (const std::string &kdf)
- void set_default_compression (const std::string &comp)
- void set_compression_level (int level)
- void set_show_progress (bool show)
- void set_verbose (bool verbose)
- std::optional< std::string > get (const std::string &key) const

  Get value by key path (e.g., "default.mode").
- bool set (const std::string &key, const std::string &value)

  Set value by key path.
- nlohmann::json to_json () const

  Convert to JSON.

Static Public Member Functions

- static Config load ()

  Load configuration from file.
- static std::filesystem::path get_config_path ()

  Get config file path.
- static Config get_default ()

  Get default config.
- static Config from_json (const nlohmann::json &j)

  Load from JSON.

Private Attributes

- std::string default_mode_ = "standard"
- std::string default_algorithm_ = "aes-256-gcm"
- std::string default_kdf_ = "argon2id"
- std::string default_compression_ = "none"
- int compression_level_ = 6
- bool show_progress_ = true
- bool verbose_ = false

9.25.1 Detailed Description

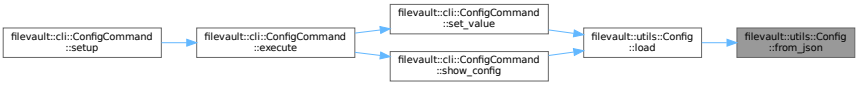
Configuration manager for FileVault.
Manages user preferences stored in ~/.filevault/config.json

9.25.2 Member Function Documentation

9.25.2.1 from_json()

```
Config filevault::utils::Config::from_json (
            const nlohmann::json & j)  [static]
```

Load from JSON.
Here is the caller graph for this function:



9.25.2.2 get()

```
std::optional< std::string > filevault::utils::Config::get (
            const std::string & key) const
```

Get value by key path (e.g., "default.mode").

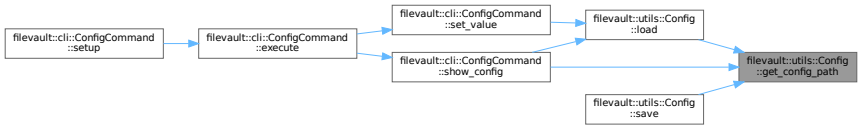
9.25.2.3 get_compression_level()

```
int filevault::utils::Config::get_compression_level () const  [inline]
```

9.25.2.4 get_config_path()

```
std::filesystem::path filevault::utils::Config::get_config_path ()  [static]
```

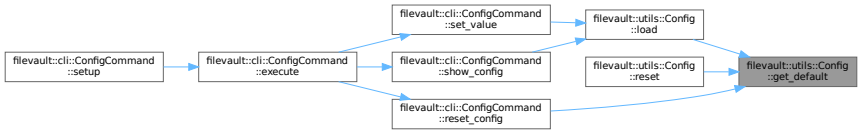
Get config file path.
Here is the caller graph for this function:



9.25.2.5 get_default()

```
Config filevault::utils::Config::get_default ()  [static]
```

Get default config.
Here is the caller graph for this function:



9.25.2.6 get_default_algorithm()

```
std::string filevault::utils::Config::get_default_algorithm () const  [inline]
```

9.25.2.7 get_default_compression()

```
std::string filevault::utils::Config::get_default_compression () const  [inline]
```

9.25.2.8 get_default_kdf()

```
std::string filevault::utils::Config::get_default_kdf () const  [inline]
```

9.25.2.9 get_default_mode()

```
std::string filevault::utils::Config::get_default_mode () const  [inline]
```

9.25.2.10 get_show_progress()

```
bool filevault::utils::Config::get_show_progress () const  [inline]
```

9.25.2.11 get_verbose()

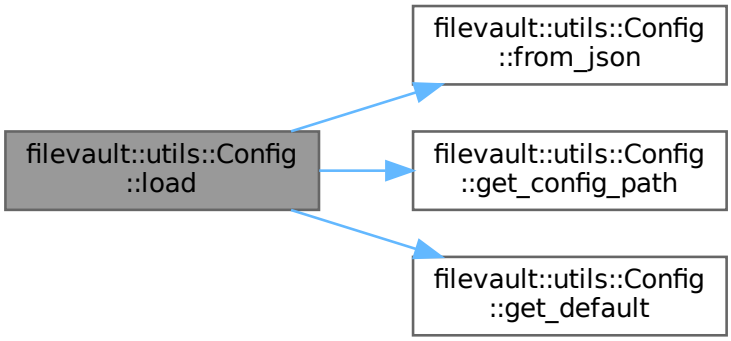
```
bool filevault::utils::Config::get_verbose () const  [inline]
```

9.25.2.12 load()

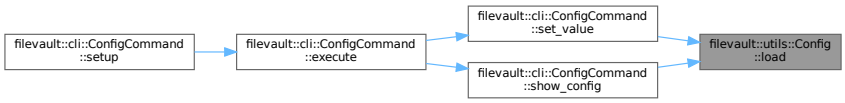
```
Config filevault::utils::Config::load ()  [static]
```

Load configuration from file.

Here is the call graph for this function:



Here is the caller graph for this function:

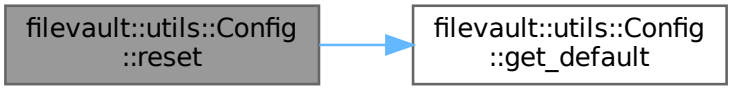


9.25.2.13 reset()

```
void filevault::utils::Config::reset ()
```

Reset to defaults.

Here is the call graph for this function:

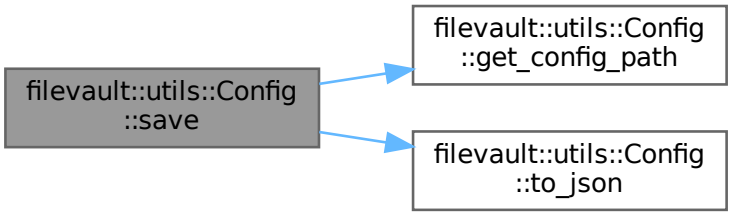


9.25.2.14 save()

```
bool filevault::utils::Config::save () const
```

Save configuration to file.

Here is the call graph for this function:



9.25.2.15 set()

```
bool filevault::utils::Config::set (
            const std::string & key,
            const std::string & value)
```

Set value by key path.

9.25.2.16 set_compression_level()

```
void filevault::utils::Config::set_compression_level (
            int level)  [inline]
```

9.25.2.17 set_default_algorithm()

```
void filevault::utils::Config::set_default_algorithm (
            const std::string & algo)  [inline]
```

9.25.2.18 set_default_compression()

```
void filevault::utils::Config::set_default_compression (
            const std::string & comp)  [inline]
```

9.25.2.19 set_default_kdf()

```
void filevault::utils::Config::set_default_kdf (
            const std::string & kdf)  [inline]
```

9.25.2.20 set_default_mode()

```
void filevault::utils::Config::set_default_mode (
            const std::string & mode)  [inline]
```

9.25.2.21 set_show_progress()

```
void filevault::utils::Config::set_show_progress (
            bool show)  [inline]
```

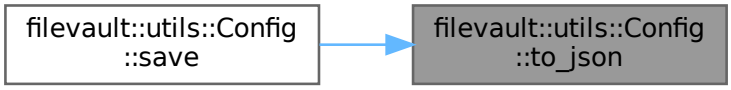
9.25.2.22 set_verbose()

```
void filevault::utils::Config::set_verbose (
            bool verbose)  [inline]
```

9.25.2.23 to_json()

```
nlohmann::json filevault::utils::Config::to_json () const
```

Convert to JSON.
Here is the caller graph for this function:



9.25.3 Member Data Documentation

9.25.3.1 compression_level_

```
int filevault::utils::Config::compression_level_ = 6  [private]
```

9.25.3.2 default_algorithm_

```
std::string filevault::utils::Config::default_algorithm_ = "aes-256-gcm"  [private]
```

9.25.3.3 default_compression_

```
std::string filevault::utils::Config::default_compression_ = "none"  [private]
```

9.25.3.4 default_kdf_

```
std::string filevault::utils::Config::default_kdf_ = "argon2id"  [private]
```

9.25.3.5 default_mode_

```
std::string filevault::utils::Config::default_mode_ = "standard"  [private]
```

9.25.3.6 show_progress_

```
bool filevault::utils::Config::show_progress_ = true  [private]
```

9.25.3.7 verbose_

```
bool filevault::utils::Config::verbose_ = false  [private]
```

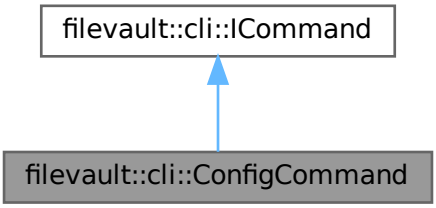
The documentation for this class was generated from the following files:

- include/filevault/utils/config.hpp
- src/utils/config.cpp

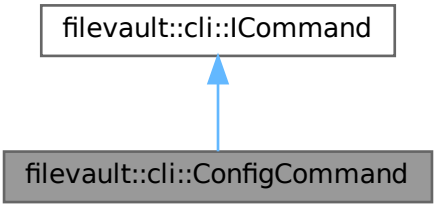
9.26 filevault::cli::ConfigCommand Class Reference

Config command - manage FileVault configuration.
`#include <config_cmd.hpp>`

Inheritance diagram for filevault::cli::ConfigCommand:



Collaboration diagram for filevault::cli::ConfigCommand:



Public Member Functions

- ConfigCommand ()=default
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- int show_config ()
- int set_value ()
- int reset_config ()

Private Attributes

- std::string subcommand_
- std::string key_
- std::string value_

9.26.1 Detailed Description

Config command - manage FileVault configuration.

9.26.2 Constructor & Destructor Documentation

9.26.2.1 ConfigCommand()

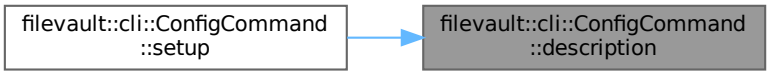
```
filevault::cli::ConfigCommand::ConfigCommand () [default]
```

9.26.3 Member Function Documentation

9.26.3.1 description()

```
std::string filevault::cli::ConfigCommand::description () const [inline], [override], [virtual]
```

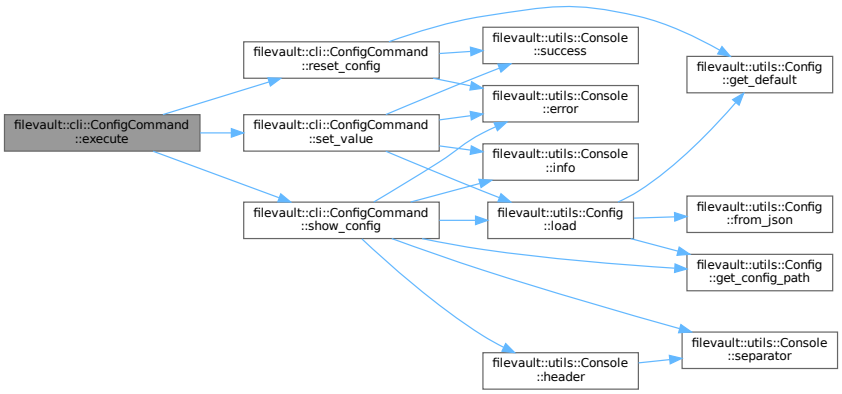
Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:



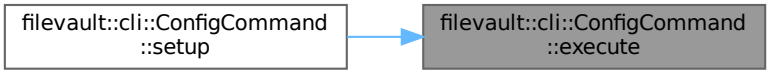
9.26.3.2 execute()

```
int filevault::cli::ConfigCommand::execute () [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



Here is the caller graph for this function:



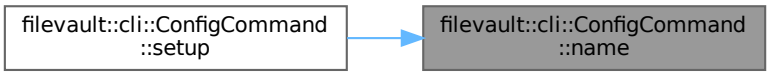
9.26.3.3 name()

`std::string filevault::cli::ConfigCommand::name () const [inline], [override], [virtual]`

Get command name.

Implements filevault::cli::ICommand.

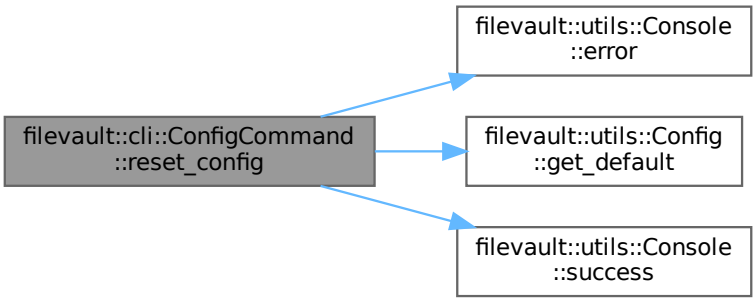
Here is the caller graph for this function:



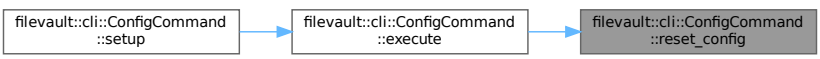
9.26.3.4 reset_config()

`int filevault::cli::ConfigCommand::reset_config () [private]`

Here is the call graph for this function:



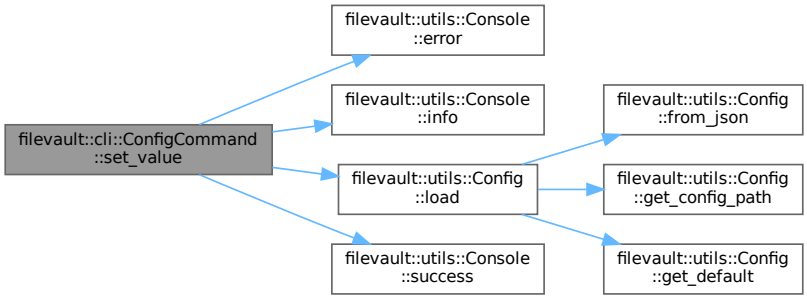
Here is the caller graph for this function:



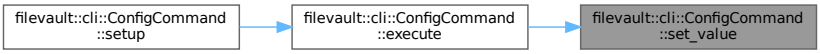
9.26.3.5 set_value()

```
int filevault::cli::ConfigCommand::set_value ()  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



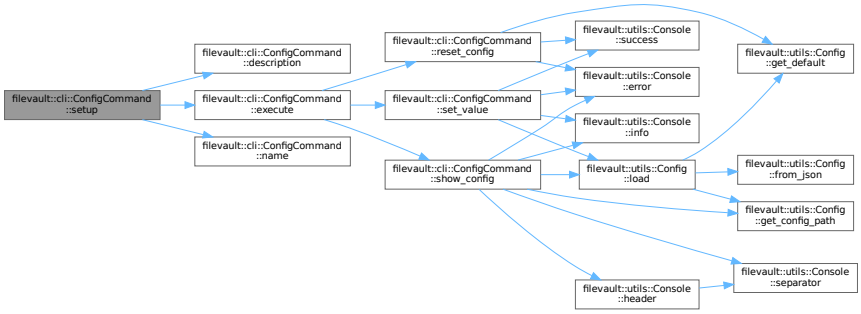
9.26.3.6 setup()

```
void filevault::cli::ConfigCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.

Implements filevault::cli::ICommand.

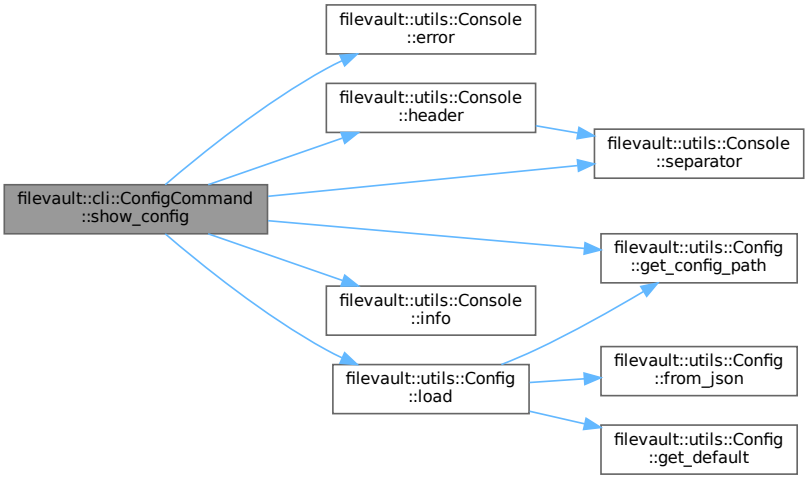
Here is the call graph for this function:



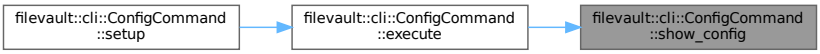
9.26.3.7 show_config()

```
int filevault::cli::ConfigCommand::show_config ()  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.26.4 Member Data Documentation

9.26.4.1 key_

```
std::string filevault::cli::ConfigCommand::key_  [private]
```

9.26.4.2 subcommand_

```
std::string filevault::cli::ConfigCommand::subcommand_  [private]
```

9.26.4.3 value_

```
std::string filevault::cli::ConfigCommand::value_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/config_cmd.hpp
- src/cli/commands/config_cmd.cpp

9.27 filevault::utils::Console Class Reference

Console utilities for colored output.

```
#include <console.hpp>
```

Static Public Member Functions

- static void success (const std::string &msg)
- static void error (const std::string &msg)
- static void warning (const std::string &msg)
- static void info (const std::string &msg)
- static void debug (const std::string &msg)
- static void separator (char ch='=', size_t width=80)
- static void header (const std::string &title)

9.27.1 Detailed Description

Console utilities for colored output.

9.27.2 Member Function Documentation

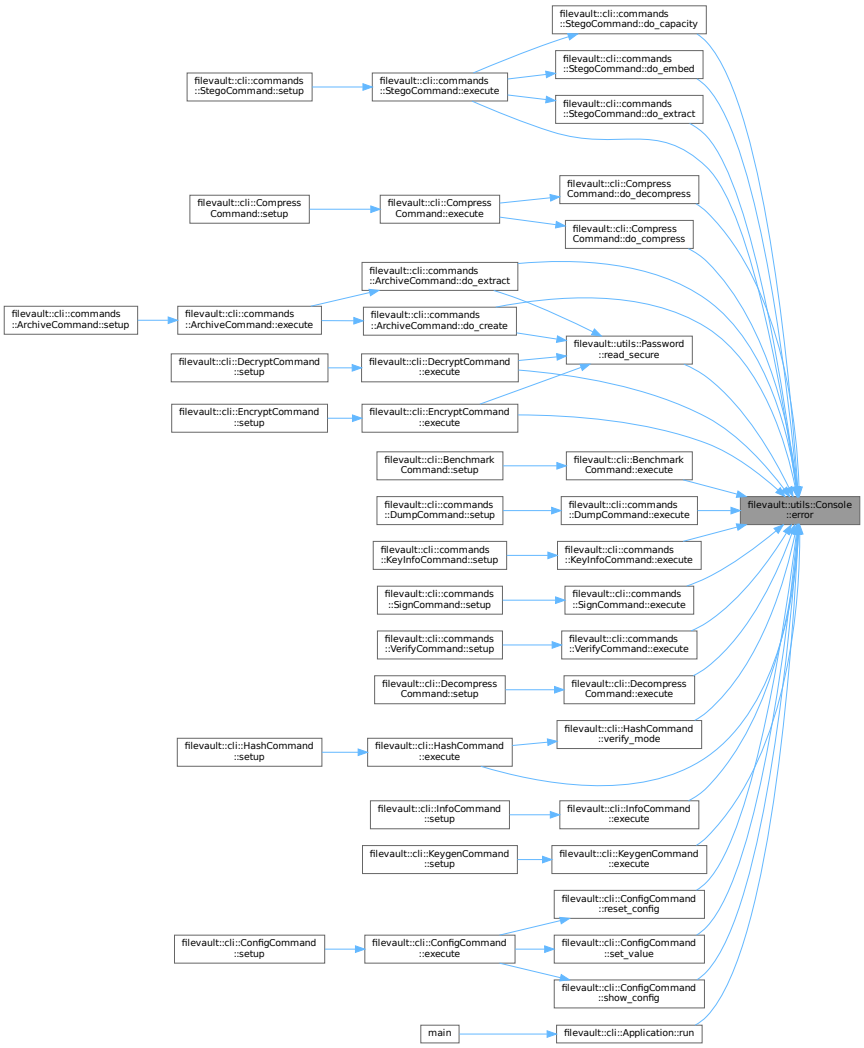
9.27.2.1 debug()

```
void filevault::utils::Console::debug (
            const std::string & msg)  [static]
```

9.27.2.2 error()

```
void filevault::utils::Console::error (
            const std::string & msg)  [static]
```

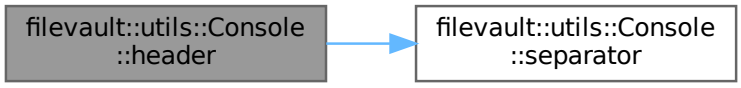
Here is the caller graph for this function:



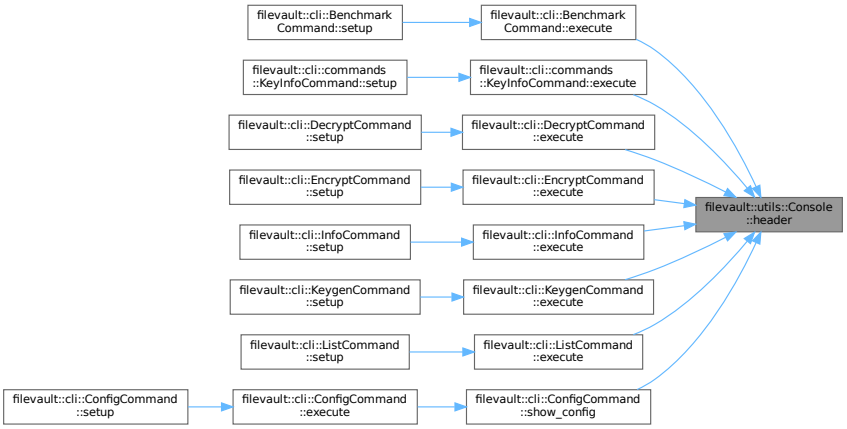
9.27.2.3 header()

```
void filevault::utils::Console::header (
            const std::string & title)  [static]
```

Here is the call graph for this function:



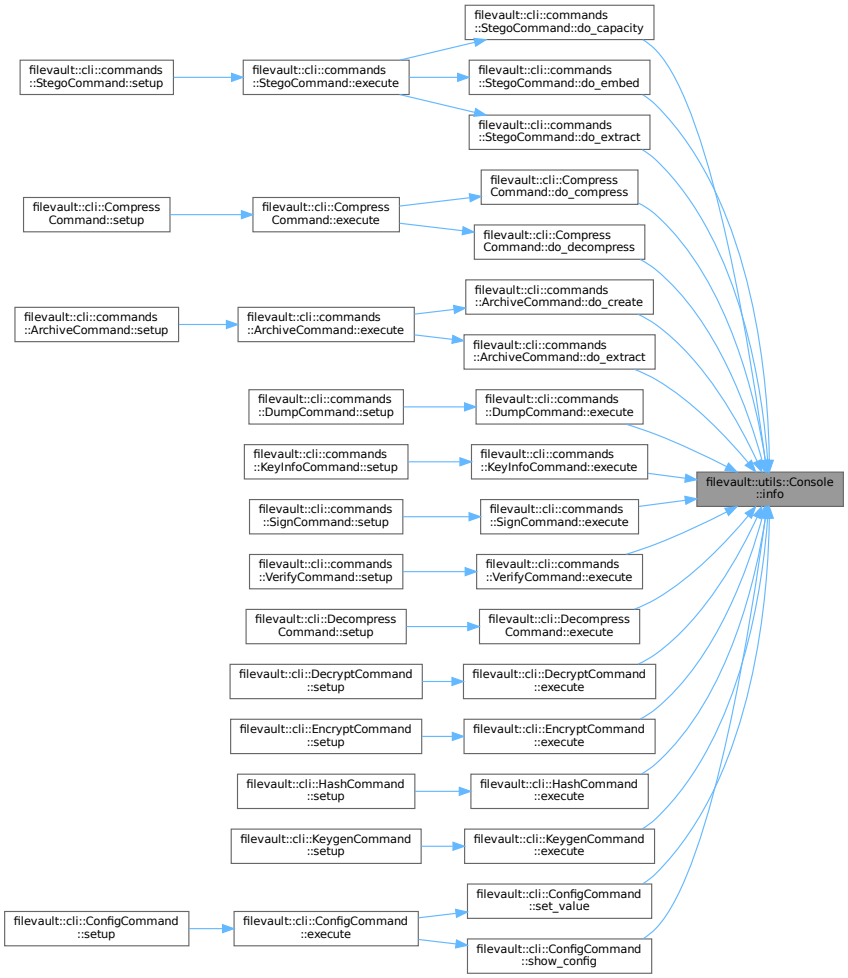
Here is the caller graph for this function:



9.27.2.4 info()

```
void filevault::utils::Console::info (
            const std::string & msg)  [static]
```

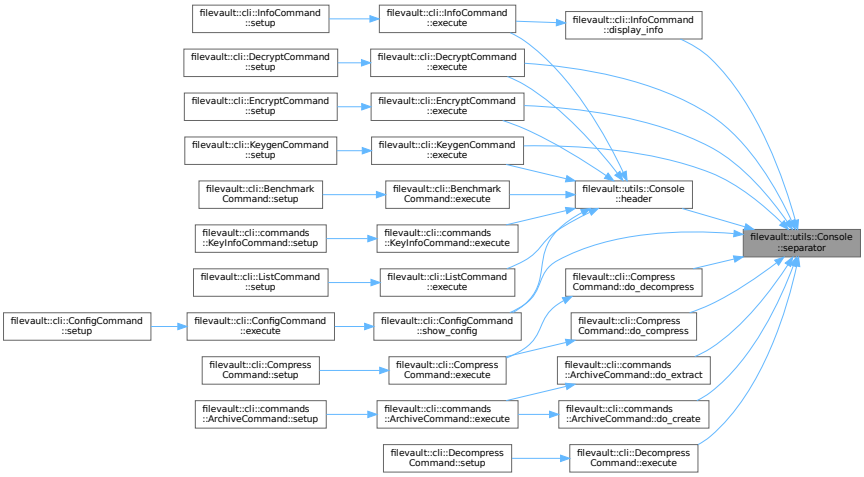
Here is the caller graph for this function:



9.27.2.5 separator()

```
void filevault::utils::Console::separator (
            char ch = '=',
            size_t width = 80)  [static]
```

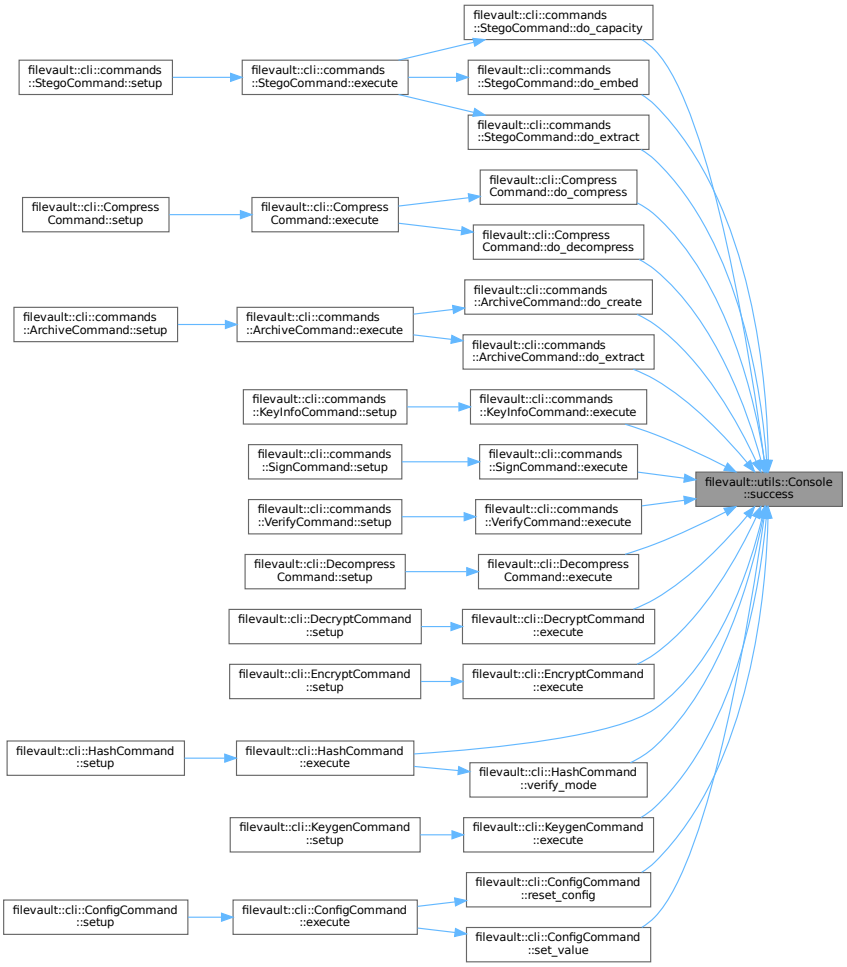
Here is the caller graph for this function:



9.27.2.6 success()

```
void filevault::utils::Console::success (
            const std::string & msg)  [static]
```

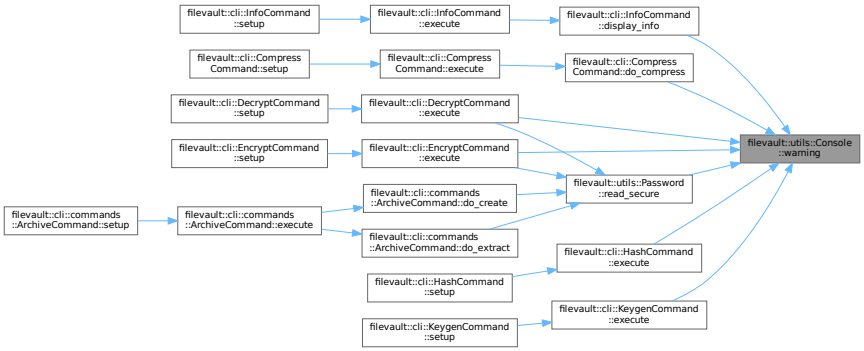
Here is the caller graph for this function:



9.27.2.7 warning()

```
void filevault::utils::Console::warning (
            const std::string & msg)  [static]
```

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/utils/console.hpp
- src/utils/console.cpp

9.28 filevault::core::CryptoEngine Class Reference

Main cryptographic engine Manages algorithms and provides key derivation.
`#include <crypto_engine.hpp>`

Public Member Functions

- CryptoEngine ()
- ~CryptoEngine ()
- CryptoEngine (const CryptoEngine &)=delete
- CryptoEngine & operator= (const CryptoEngine &)=delete
- void initialize ()

  Initialize engine with default algorithms.
- void register_algorithm (std::unique_ptr< ICryptoAlgorithm > algorithm)

  Register a custom algorithm.
- ICryptoAlgorithm * get_algorithm (AlgorithmType type)

  Get algorithm by type.
- std::vector< uint8_t > derive_key (const std::string &password, const std::vector< uint8_t > &salt, const EncryptionConfig &config)

  Derive key from password using KDF.

Static Public Member Functions

- static std::vector< uint8_t > generate_salt (size_t length=32)

  Generate random salt (32 bytes default).
- static std::vector< uint8_t > generate_nonce (size_t length=12)

  Generate random nonce/IV.
- static std::string algorithm_name (AlgorithmType type)

  Get algorithm name from type.
- static std::string kdf_name (KDFType type)

  Get KDF name from type.
- static std::string security_level_name (SecurityLevel level)

  Get security level name from type.
- static std::optional< AlgorithmType > parse_algorithm (const std::string &name)

  Parse algorithm from string.
- static std::optional< KDFType > parse_kdf (const std::string &name)

  Parse KDF from string.
- static std::optional< SecurityLevel > parse_security_level (const std::string &name)

  Parse security level from string.

Private Attributes

- std::map< AlgorithmType, std::unique_ptr< ICryptoAlgorithm > > algorithms_

9.28.1 Detailed Description

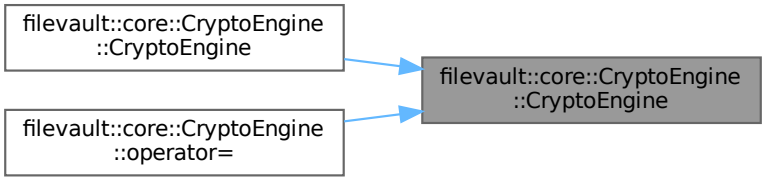
Main cryptographic engine Manages algorithms and provides key derivation.

9.28.2 Constructor & Destructor Documentation

9.28.2.1 CryptoEngine() [1/2]

```
filevault::core::CryptoEngine::CryptoEngine ()
```

Here is the caller graph for this function:



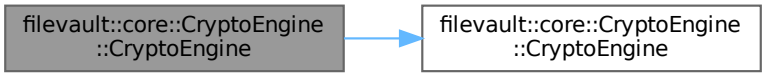
9.28.2.2 ~CryptoEngine()

```
filevault::core::CryptoEngine::~CryptoEngine ()
```

9.28.2.3 CryptoEngine() [2/2]

```
filevault::core::CryptoEngine::CryptoEngine (
            const CryptoEngine & )  [delete]
```

Here is the call graph for this function:



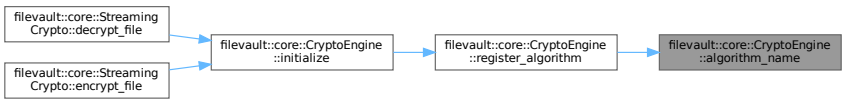
9.28.3 Member Function Documentation

9.28.3.1 algorithm_name()

```
std::string filevault::core::CryptoEngine::algorithm_name (
            AlgorithmType type)  [static]
```

Get algorithm name from type.

Here is the caller graph for this function:



9.28.3.2 derive_key()

```
std::vector< uint8_t > filevault::core::CryptoEngine::derive_key (
            const std::string & password,
            const std::vector< uint8_t > & salt,
            const EncryptionConfig & config)
```

Derive key from password using KDF.

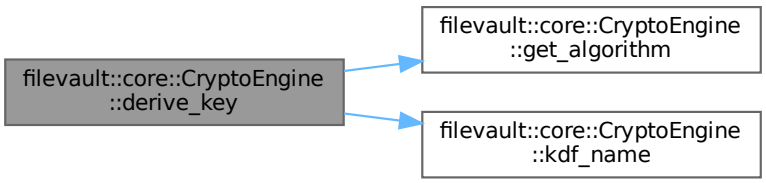
Parameters

password	User password
salt	Random salt (MUST be unique per file)
config	Configuration containing KDF parameters

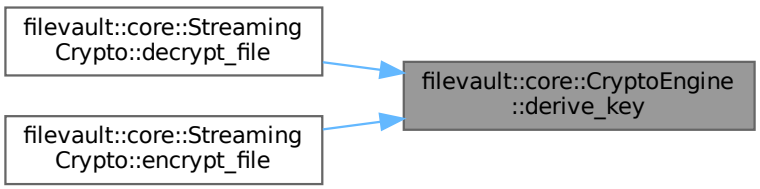
Returns

Derived key

Here is the call graph for this function:



Here is the caller graph for this function:



9.28.3.3 generate_nonce()

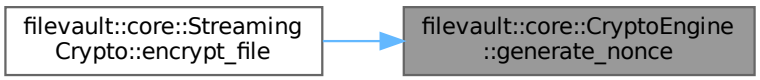
```
std::vector< uint8_t > filevault::core::CryptoEngine::generate_nonce (
            size_t length = 12)  [static]
```

Generate random nonce/IV.

Parameters

length	Nonce length (12 for GCM, 16 for CBC)

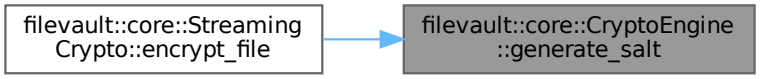
Here is the caller graph for this function:



9.28.3.4 generate_salt()

```
std::vector< uint8_t > filevault::core::CryptoEngine::generate_salt (
            size_t length = 32)  [static]
```

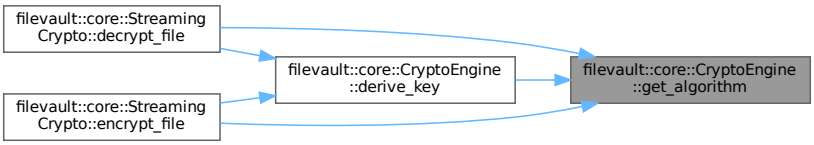
Generate random salt (32 bytes default).
Here is the caller graph for this function:



9.28.3.5 get_algorithm()

```
ICryptoAlgorithm * filevault::core::CryptoEngine::get_algorithm (
            AlgorithmType type)
```

Get algorithm by type.
Here is the caller graph for this function:

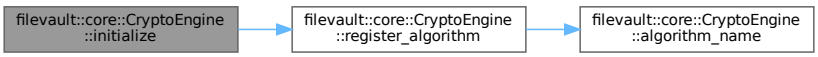


9.28.3.6 initialize()

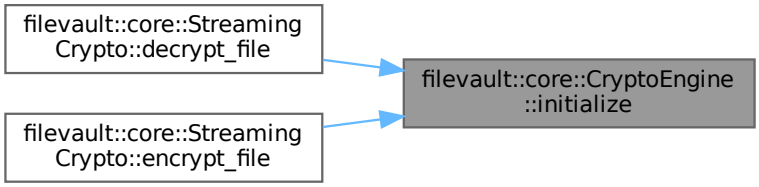
```
void filevault::core::CryptoEngine::initialize ()
```

Initialize engine with default algorithms.

Here is the call graph for this function:



Here is the caller graph for this function:



9.28.3.7 kdf_name()

```
std::string filevault::core::CryptoEngine::kdf_name (
            KDFType type)  [static]
```

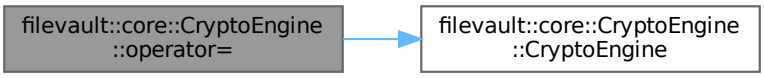
Get KDF name from type.

Here is the caller graph for this function:

9.28.3.8 operator=()

```
CryptoEngine & filevault::core::CryptoEngine::operator= (
            const CryptoEngine & )  [delete]
```

Here is the call graph for this function:



9.28.3.9 parse_algorithm()

```
std::optional< AlgorithmType > filevault::core::CryptoEngine::parse_algorithm (
            const std::string & name)  [static]
```

Parse algorithm from string.

9.28.3.10 parse_kdf()

```
std::optional< KDFType > filevault::core::CryptoEngine::parse_kdf (
            const std::string & name)  [static]
```

Parse KDF from string.

9.28.3.11 parse_security_level()

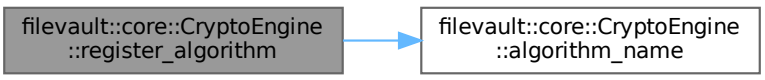
```
std::optional< SecurityLevel > filevault::core::CryptoEngine::parse_security_level (
            const std::string & name)  [static]
```

Parse security level from string.

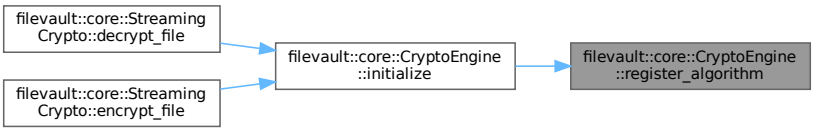
9.28.3.12 register_algorithm()

```
void filevault::core::CryptoEngine::register_algorithm (
            std::unique_ptr< ICryptoAlgorithm > algorithm)
```

Register a custom algorithm.
Here is the call graph for this function:



Here is the caller graph for this function:



9.28.3.13 security_level_name()

```
std::string filevault::core::CryptoEngine::security_level_name (
            SecurityLevel level)  [static]
```

Get security level name from type.

9.28.4 Member Data Documentation

9.28.4.1 algorithms_

```
std::map<AlgorithmType, std::unique_ptr<ICryptoAlgorithm> > filevault::core::CryptoEngine↩
::algorithms_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/core/crypto_engine.hpp
- src/core/crypto_engine.cpp

9.29 filevault::core::CryptoResult Struct Reference

Result of cryptographic operations.
`#include <result.hpp>`

Public Attributes

- bool success = false
- std::string error_message
- std::vector< uint8_t > data
- AlgorithmType algorithm_used
- size_t original_size = 0
- size_t final_size = 0
- double processing_time_ms = 0.0
- std::optional< std::vector< uint8_t > > salt
- std::optional< std::vector< uint8_t > > nonce
- std::optional< std::vector< uint8_t > > tag

9.29.1 Detailed Description

Result of cryptographic operations.

9.29.2 Member Data Documentation

9.29.2.1 algorithm_used

```
AlgorithmType filevault::core::CryptoResult::algorithm_used
```

9.29.2.2 data

```
std::vector<uint8_t> filevault::core::CryptoResult::data
```

9.29.2.3 error_message

```
std::string filevault::core::CryptoResult::error_message
```

9.29.2.4 final_size

```
size_t filevault::core::CryptoResult::final_size = 0
```

9.29.2.5 nonce

```
std::optional<std::vector<uint8_t> > filevault::core::CryptoResult::nonce
```

9.29.2.6 original_size

```
size_t filevault::core::CryptoResult::original_size = 0
```

9.29.2.7 processing_time_ms

```
double filevault::core::CryptoResult::processing_time_ms = 0.0
```

9.29.2.8 salt

```
std::optional<std::vector<uint8_t> > filevault::core::CryptoResult::salt
```

9.29.2.9 success

```
bool filevault::core::CryptoResult::success = false
```

9.29.2.10 tag

```
std::optional<std::vector<uint8_t> > filevault::core::CryptoResult::tag
```

The documentation for this struct was generated from the following file:

- include/filevault/core/result.hpp

9.30 filevault::utils::CryptoUtils Class Reference

Crypto utility functions.

```
#include <crypto_utils.hpp>
```

Static Public Member Functions

- static std::string hex_encode (std::span< const uint8_t > data)

  Encode bytes to hexadecimal string.
- static std::vector< uint8_t > hex_decode (const std::string &hex)

  Decode hexadecimal string to bytes.
- static std::string format_bytes (size_t bytes)

  Format bytes as human-readable size.

9.30.1 Detailed Description

Crypto utility functions.

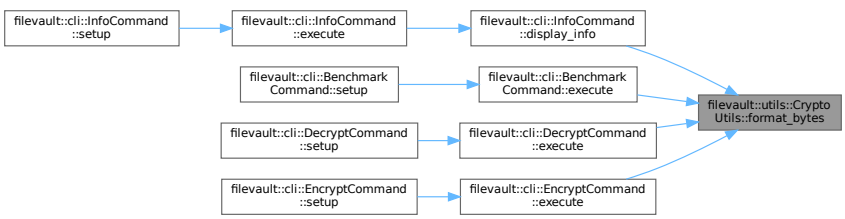
9.30.2 Member Function Documentation

9.30.2.1 format_bytes()

```
std::string filevault::utils::CryptoUtils::format_bytes (
            size_t bytes)  [static]
```

Format bytes as human-readable size.

Here is the caller graph for this function:



9.30.2.2 hex_decode()

```
std::vector< uint8_t > filevault::utils::CryptoUtils::hex_decode (
            const std::string & hex)  [static]
```

Decode hexadecimal string to bytes.

9.30.2.3 hex_encode()

```
std::string filevault::utils::CryptoUtils::hex_encode (
            std::span< const uint8_t > data)  [static]
```

Encode bytes to hexadecimal string.

The documentation for this class was generated from the following files:

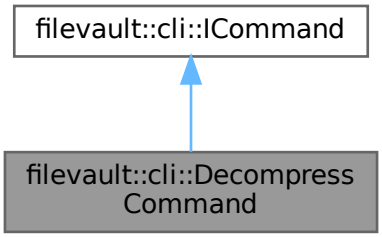
- include/filevault/utils/crypto_utils.hpp
- src/utils/crypto_utils.cpp

9.31 filevault::cli::DecompressCommand Class Reference

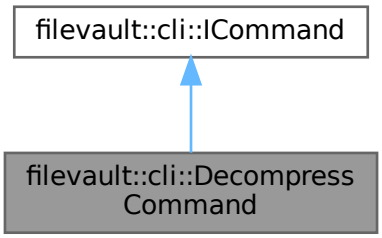
Decompress command - alias for compress -d.

`#include <decompress_cmd.hpp>`

Inheritance diagram for filevault::cli::DecompressCommand:



Collaboration diagram for filevault::cli::DecompressCommand:



Public Member Functions

- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.

- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- std::string detect_algorithm (const std::string &path)
- std::string generate_output_path (const std::string &input)

Private Attributes

- CLI::App * subcommand_ = nullptr
- std::string input_file_
- std::string output_file_
- std::string algorithm_
- bool auto_detect_ = true
- bool verbose_ = false
- bool benchmark_ = false

9.31.1 Detailed Description

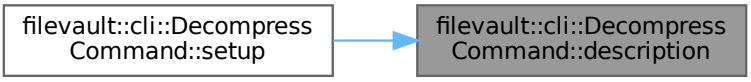
Decompress command - alias for compress -d.
Provides a direct `decompress` command for better UX

9.31.2 Member Function Documentation

9.31.2.1 description()

```
std::string filevault::cli::DecompressCommand::description () const  [inline], [override],
[virtual]
```

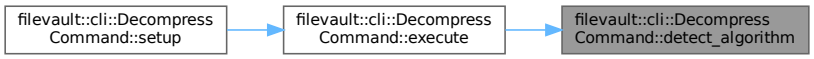
Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:



9.31.2.2 detect_algorithm()

```
std::string filevault::cli::DecompressCommand::detect_algorithm (
            const std::string & path)  [private]
```

Here is the caller graph for this function:



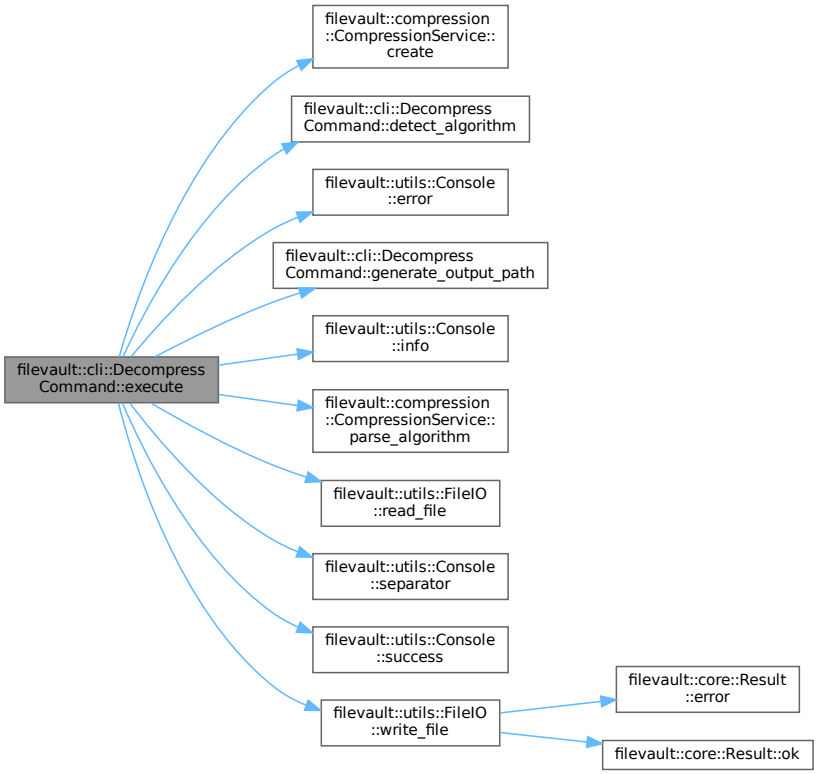
9.31.2.3 execute()

`int filevault::cli::DecompressCommand::execute () [override], [virtual]`

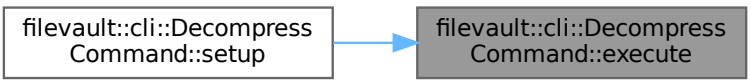
Execute the command.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



9.31.2.4 generate_output_path()

```
std::string filevault::cli::DecompressCommand::generate_output_path (
            const std::string & input)  [private]
```

Here is the caller graph for this function:

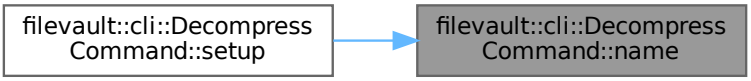
9.31.2.5 name()

```
std::string filevault::cli::DecompressCommand::name () const  [inline], [override], [virtual]
```

Get command name.

Implements filevault::cli::ICommand.

Here is the caller graph for this function:



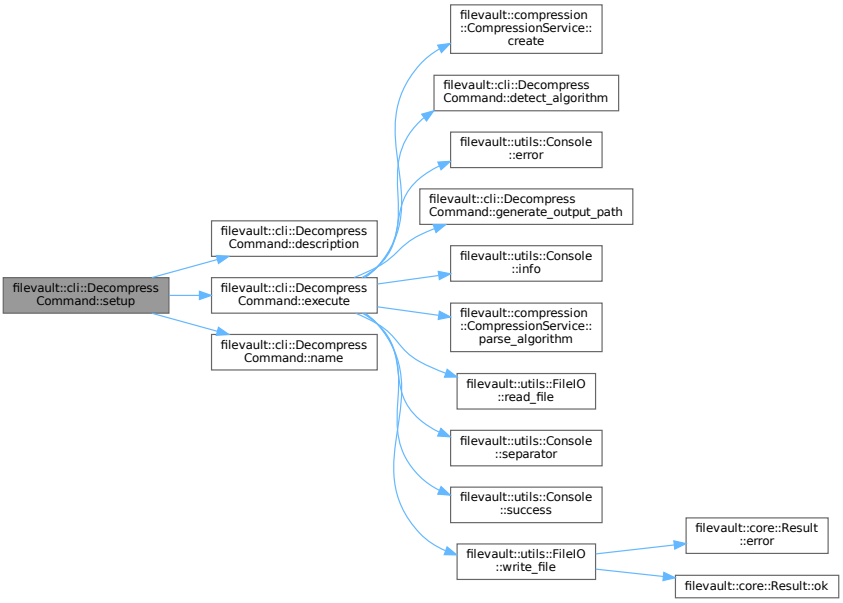
9.31.2.6 setup()

```
void filevault::cli::DecompressCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



9.31.3 Member Data Documentation

9.31.3.1 algorithm_

```
std::string filevault::cli::DecompressCommand::algorithm_  [private]
```

9.31.3.2 auto_detect_

```
bool filevault::cli::DecompressCommand::auto_detect_ = true  [private]
```

9.31.3.3 benchmark_

```
bool filevault::cli::DecompressCommand::benchmark_ = false  [private]
```

9.31.3.4 input_file_

```
std::string filevault::cli::DecompressCommand::input_file_  [private]
```

9.31.3.5 output_file_

```
std::string filevault::cli::DecompressCommand::output_file_  [private]
```

9.31.3.6 subcommand_

```
CLI::App* filevault::cli::DecompressCommand::subcommand_ = nullptr  [private]
```

9.31.3.7 verbose_

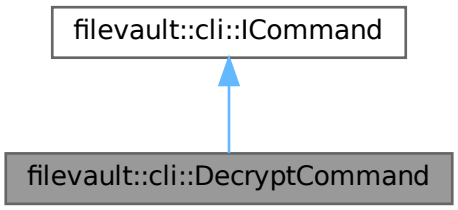
```
bool filevault::cli::DecompressCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

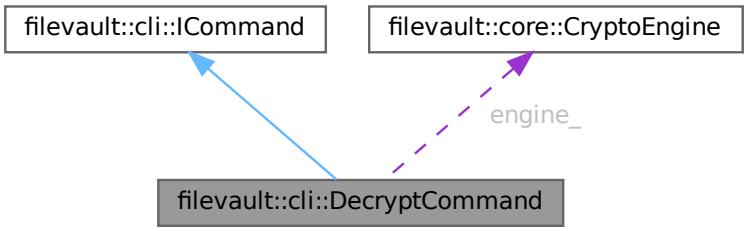
- include/filevault/cli/commands/decompress_cmd.hpp
- src/cli/commands/decompress_cmd.cpp

9.32 filevault::cli::DecryptCommand Class Reference

`#include <decrypt_cmd.hpp>`
Inheritance diagram for filevault::cli::DecryptCommand:



Collaboration diagram for filevault::cli::DecryptCommand:



Public Member Functions

- DecryptCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- core::CryptoEngine & engine_
- std::string input_file_
- std::string output_file_

- std::string password_
- bool verbose_ = false
- bool no_progress_ = false

9.32.1 Constructor & Destructor Documentation

9.32.1.1 DecryptCommand()

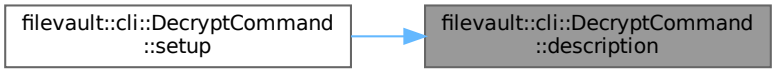
```
filevault::cli::DecryptCommand::DecryptCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.32.2 Member Function Documentation

9.32.2.1 description()

```
std::string filevault::cli::DecryptCommand::description () const  [inline], [override], [virtual]
```

Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:

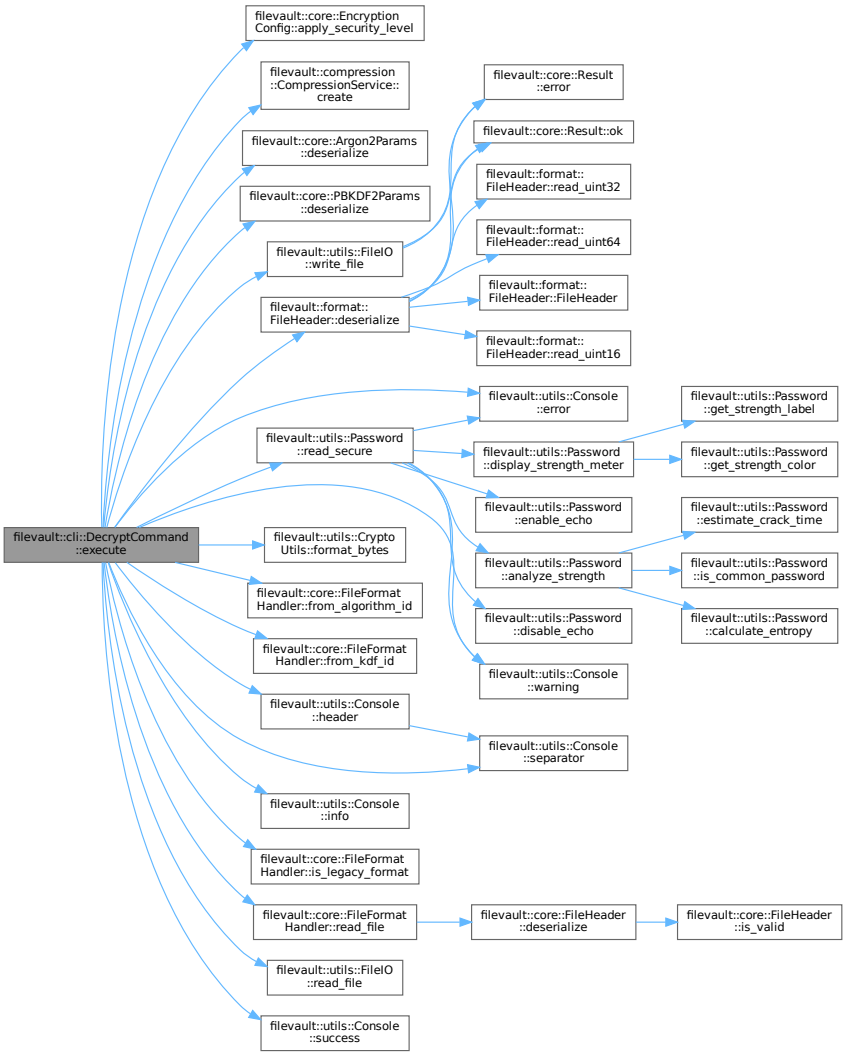


9.32.2.2 execute()

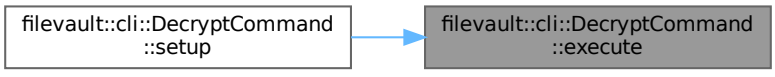
```
int filevault::cli::DecryptCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



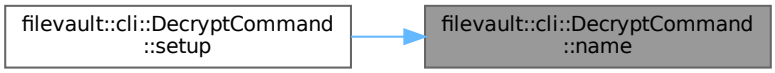
9.32.2.3 name()

`std::string filevault::cli::DecryptCommand::name () const [inline], [override], [virtual]`

Get command name.

Implements filevault::cli::ICommand.

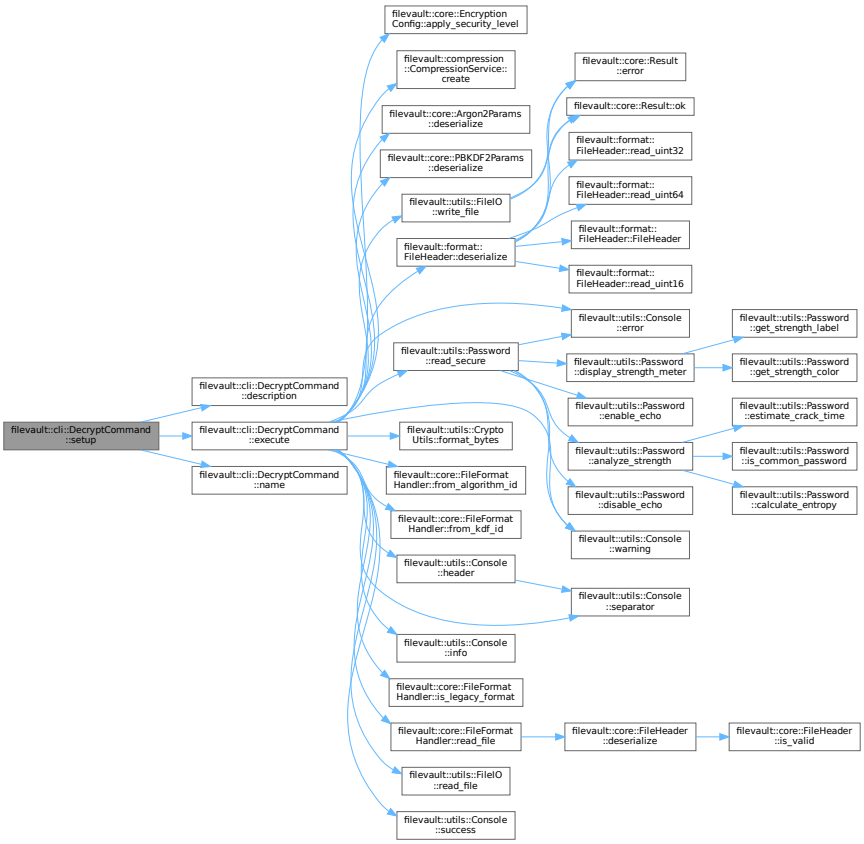
Here is the caller graph for this function:



9.32.2.4 setup()

```
void filevault::cli::DecryptCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



9.32.3 Member Data Documentation

9.32.3.1 engine_

```
core::CryptoEngine& filevault::cli::DecryptCommand::engine_  [private]
```

9.32.3.2 input_file_

```
std::string filevault::cli::DecryptCommand::input_file_  [private]
```

9.32.3.3 no_progress_

```
bool filevault::cli::DecryptCommand::no_progress_ = false  [private]
```

9.32.3.4 output_file_

```
std::string filevault::cli::DecryptCommand::output_file_  [private]
```

9.32.3.5 password_

```
std::string filevault::cli::DecryptCommand::password_  [private]
```

9.32.3.6 verbose_

```
bool filevault::cli::DecryptCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/decrypt_cmd.hpp
- src/cli/commands/decrypt_cmd.cpp

9.33 filevault::algorithms::pqc::Dilithium Class Reference

Dilithium Digital Signature Algorithm.

```
#include <post_quantum.hpp>
```

Public Types

- enum class Variant { Dilithium2 , Dilithium3 , Dilithium5 }

Public Member Functions

- Dilithium (Variant variant=Variant::Dilithium3)
- std::string name () const
- PQKeyPair generate_keypair ()

  Generate Dilithium key pair.
- std::vector< uint8_t > sign (std::span< const uint8_t > message, std::span< const uint8_t > private_key)

  Sign a message.
- bool verify (std::span< const uint8_t > message, std::span< const uint8_t > signature, std::span< const uint8_t > public_key)

  Verify a signature.
- size_t public_key_size () const

  Get public key size for this variant.
- size_t private_key_size () const

  Get private key size for this variant.
- size_t signature_size () const

  Get signature size for this variant.

Private Attributes

- Variant variant_
- std::string botan_name_

9.33.1 Detailed Description

Dilithium Digital Signature Algorithm.
NIST FIPS 204 (ML-DSA) - Post-quantum secure digital signatures
Dilithium is a lattice-based signature scheme that provides:

- EUF-CMA security
- Resistance to quantum computer attacks
- Efficient signing and verification

Variants:

- Dilithium2: ∼128-bit post-quantum security
- Dilithium3: ∼192-bit post-quantum security (recommended)
- Dilithium5: ∼256-bit post-quantum security

9.33.2 Member Enumeration Documentation

9.33.2.1 Variant

```
enum class filevault::algorithms::pqc::Dilithium::Variant  [strong]
```

Enumerator

Dilithium2	
Dilithium3	
Dilithium5	

9.33.3 Constructor & Destructor Documentation

9.33.3.1 Dilithium()

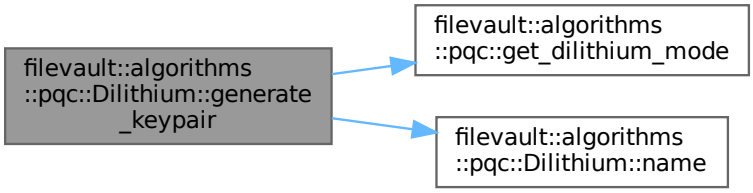
```
filevault::algorithms::pqc::Dilithium::Dilithium (
            Variant variant = Variant::Dilithium3)  [explicit]
```

9.33.4 Member Function Documentation

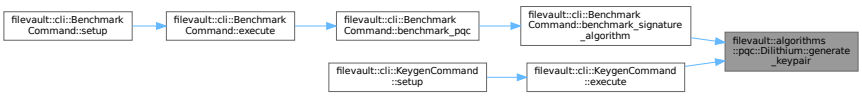
9.33.4.1 generate_keypair()

```
PQKeyPair filevault::algorithms::pqc::Dilithium::generate_keypair ()
```

Generate Dilithium key pair.
Here is the call graph for this function:



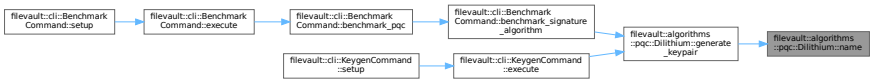
Here is the caller graph for this function:



9.33.4.2 name()

```
std::string filevault::algorithms::pqc::Dilithium::name () const
```

Here is the caller graph for this function:



9.33.4.3 private_key_size()

```
size_t filevault::algorithms::pqc::Dilithium::private_key_size () const
```

Get private key size for this variant.

9.33.4.4 public_key_size()

```
size_t filevault::algorithms::pqc::Dilithium::public_key_size () const
```

Get public key size for this variant.

9.33.4.5 sign()

```
std::vector< uint8_t > filevault::algorithms::pqc::Dilithium::sign (
            std::span< const uint8_t > message,
            std::span< const uint8_t > private_key)
```

Sign a message.

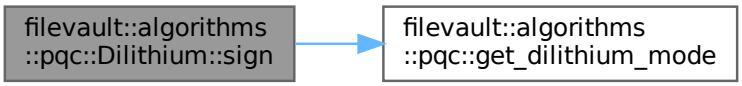
Parameters

message	Message to sign
private_key	Signer's private key

Returns

Signature

Here is the call graph for this function:



Here is the caller graph for this function:

9.33.4.6 signature_size()

```
size_t filevault::algorithms::pqc::Dilithium::signature_size () const
```

Get signature size for this variant.

9.33.4.7 verify()

```
bool filevault::algorithms::pqc::Dilithium::verify (
            std::span< const uint8_t > message,
            std::span< const uint8_t > signature,
            std::span< const uint8_t > public_key)
```

Verify a signature.

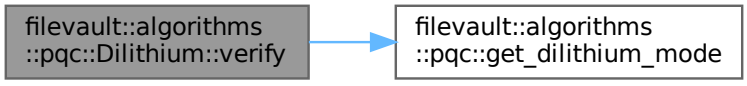
Parameters

message	Original message
signature	Signature to verify
public_key	Signer's public key

Returns

true if signature is valid

Here is the call graph for this function:



Here is the caller graph for this function:

9.33.5 Member Data Documentation

9.33.5.1 botan_name_

```
std::string filevault::algorithms::pqc::Dilithium::botan_name_  [private]
```

9.33.5.2 variant_

```
Variant filevault::algorithms::pqc::Dilithium::variant_  [private]
```

The documentation for this class was generated from the following files:

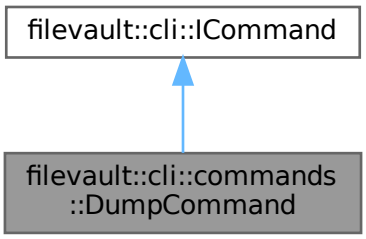
- include/filevault/algorithms/pqc/post_quantum.hpp
- src/algorithms/pqc/post_quantum.cpp

9.34 filevault::cli::commands::DumpCommand Class Reference

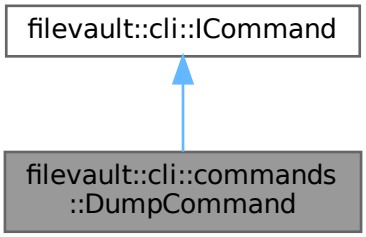
Command to dump file content in various formats (hex, binary, base64).

`#include <dump_cmd.hpp>`

Inheritance diagram for filevault::cli::commands::DumpCommand:



Collaboration diagram for filevault::cli::commands::DumpCommand:



Public Member Functions

- DumpCommand ()
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- std::string name_ = "dump"
- std::string description_ = "View file content in hex, binary, or base64 format"
- std::string file_path_
- std::string format_ = "hex"
- size_t max_bytes_ = 0
- bool show_offset_ = true
- bool show_ascii_ = true

9.34.1 Detailed Description

Command to dump file content in various formats (hex, binary, base64).

9.34.2 Constructor & Destructor Documentation

9.34.2.1 DumpCommand()

```
filevault::cli::commands::DumpCommand::DumpCommand ()  [default]
```

9.34.3 Member Function Documentation

9.34.3.1 description()

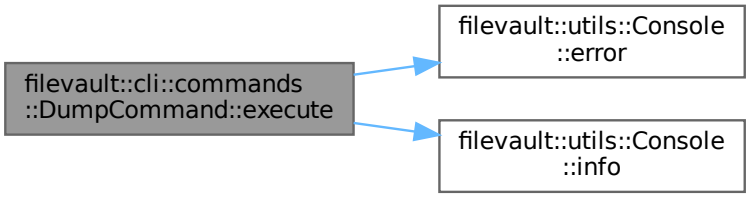
```
std::string filevault::cli::commands::DumpCommand::description () const  [inline], [override],
[virtual]
```

Get command description.
Implements filevault::cli::ICommand.

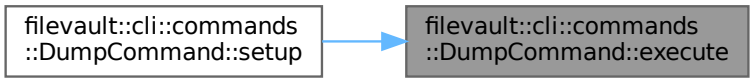
9.34.3.2 execute()

```
int filevault::cli::commands::DumpCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



Here is the caller graph for this function:



9.34.3.3 name()

```
std::string filevault::cli::commands::DumpCommand::name () const  [inline], [override], [virtual]
```

Get command name.
Implements filevault::cli::ICommand.

9.34.3.4 setup()

```
void filevault::cli::commands::DumpCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.

Here is the call graph for this function:

9.34.4 Member Data Documentation

9.34.4.1 description_

```
std::string filevault::cli::commands::DumpCommand::description_ = "View file content in hex,
binary, or base64 format"  [private]
```

9.34.4.2 file_path_

```
std::string filevault::cli::commands::DumpCommand::file_path_  [private]
```

9.34.4.3 format_

```
std::string filevault::cli::commands::DumpCommand::format_ = "hex"  [private]
```

9.34.4.4 max_bytes_

```
size_t filevault::cli::commands::DumpCommand::max_bytes_ = 0  [private]
```

9.34.4.5 name_

```
std::string filevault::cli::commands::DumpCommand::name_ = "dump"  [private]
```

9.34.4.6 show_ascii_

```
bool filevault::cli::commands::DumpCommand::show_ascii_ = true  [private]
```

9.34.4.7 show_offset_

```
bool filevault::cli::commands::DumpCommand::show_offset_ = true  [private]
```

The documentation for this class was generated from the following files:

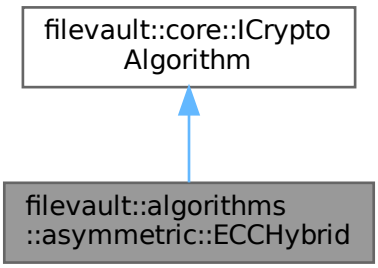
- include/filevault/cli/commands/dump_cmd.hpp
- src/cli/commands/dump_cmd.cpp

9.35 filevault::algorithms::asymmetric::ECCHybrid Class Reference

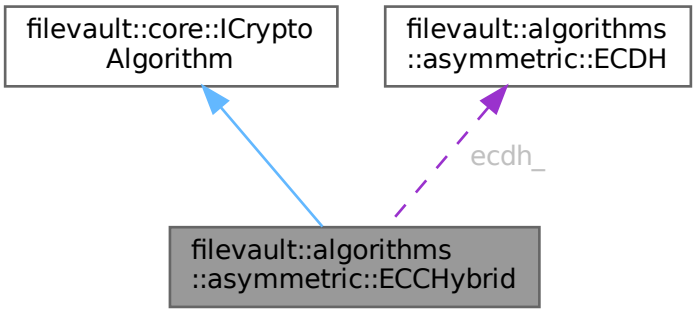
Hybrid encryption using ECDH + AES-GCM.

```
#include <ecc.hpp>
```

Inheritance diagram for filevault::algorithms::asymmetric::ECCHybrid:



Collaboration diagram for filevault::algorithms::asymmetric::ECCHybrid:



Public Member Functions

- ECCHybrid (ECCurve curve=ECCurve::SECP256R1)

  Construct ECC hybrid encryption.
- ~ECCHybrid () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt with recipient's public key.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt with own private key.
- ECCKeyPair generate_key_pair ()

  Generate a new ECC key pair for this hybrid scheme.
- size_t key_size () const override

Get recommended key size in bytes.

- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- ECCurve curve_
- std::string botan_curve_name_
- core::AlgorithmType type_
- ECDH ecdh_

9.35.1 Detailed Description

Hybrid encryption using ECDH + AES-GCM.
Combines the security of ECC key exchange with the efficiency of symmetric encryption for encrypting arbitrary-sized data.
Encryption flow:

1. Generate ephemeral ECDH key pair
2. Derive shared secret with recipient's public key
3. Use KDF to derive AES key from shared secret
4. Encrypt data with AES-GCM
5. Output: ephemeral public key + nonce + ciphertext + tag

9.35.2 Constructor & Destructor Documentation

9.35.2.1 ECCHybrid()

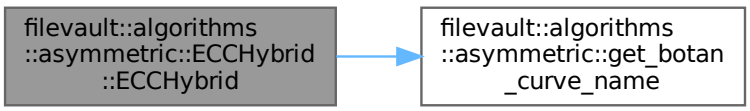
```
filevault::algorithms::asymmetric::ECCHybrid::ECCHybrid (
            ECCurve curve = ECCurve::SECP256R1)  [explicit]
```

Construct ECC hybrid encryption.

Parameters

curve	Elliptic curve to use

Here is the call graph for this function:



9.35.2.2 ~ECCHybrid()

```
filevault::algorithms::asymmetric::ECCHybrid::~ECCHybrid () [override], [default]
```

9.35.3 Member Function Documentation

9.35.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::asymmetric::ECCHybrid::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config) [override], [virtual]
```

Decrypt with own private key.

Parameters

ciphertext	Encrypted data (includes ephemeral public key)
key	Own private key
config	Encryption configuration

Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:

9.35.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::asymmetric::ECCHybrid::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config) [override], [virtual]
```

Encrypt with recipient's public key.

Parameters

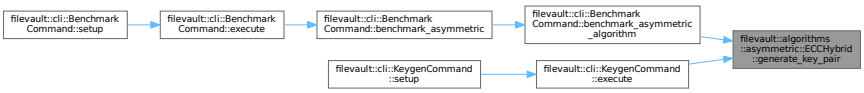
plaintext	Data to encrypt
key	Recipient's public key
config	Encryption configuration

Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:

9.35.3.3 generate_key_pair()

```
ECCKeyPair filevault::algorithms::asymmetric::ECCHybrid::generate_key_pair ()
```

Generate a new ECC key pair for this hybrid scheme.
Here is the caller graph for this function:



9.35.3.4 is_suitable_for()

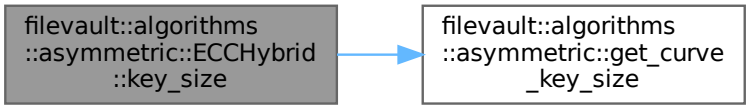
```
bool filevault::algorithms::asymmetric::ECCHybrid::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.35.3.5 key_size()

```
size_t filevault::algorithms::asymmetric::ECCHybrid::key_size () const  [override], [virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.35.3.6 name()

```
std::string filevault::algorithms::asymmetric::ECCHybrid::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:

9.35.3.7 type()

```
core::AlgorithmType filevault::algorithms::asymmetric::ECCHybrid::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.35.4 Member Data Documentation

9.35.4.1 botan_curve_name_

```
std::string filevault::algorithms::asymmetric::ECCHybrid::botan_curve_name_  [private]
```

9.35.4.2 curve_

```
ECCurve filevault::algorithms::asymmetric::ECCHybrid::curve_  [private]
```

9.35.4.3 ecdh_

```
ECDH filevault::algorithms::asymmetric::ECCHybrid::ecdh_  [private]
```

9.35.4.4 type_

```
core::AlgorithmType filevault::algorithms::asymmetric::ECCHybrid::type_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/asymmetric/ecc.hpp
- src/algorithms/asymmetric/ecc.cpp

9.36 filevault::algorithms::asymmetric::ECCKeyPair Struct Reference

ECC key pair (ECDH or ECDSA).

`#include <ecc.hpp>`

Public Attributes

- std::vector< uint8_t > public_key
- std::vector< uint8_t > private_key
- ECCurve curve
- std::string curve_name

9.36.1 Detailed Description

ECC key pair (ECDH or ECDSA).

9.36.2 Member Data Documentation

9.36.2.1 curve

```
ECCurve filevault::algorithms::asymmetric::ECCKeyPair::curve
```

9.36.2.2 curve_name

```
std::string filevault::algorithms::asymmetric::ECCKeyPair::curve_name
```

9.36.2.3 private_key

```
std::vector<uint8_t> filevault::algorithms::asymmetric::ECCKeyPair::private_key
```

9.36.2.4 public_key

```
std::vector<uint8_t> filevault::algorithms::asymmetric::ECCKeyPair::public_key
```

The documentation for this struct was generated from the following file:

- include/filevault/algorithms/asymmetric/ecc.hpp

9.37 filevault::algorithms::asymmetric::ECDH Class Reference

Elliptic Curve Diffie-Hellman (ECDH) key exchange.
`#include <ecc.hpp>`

Public Member Functions

- ECDH (ECCurve curve=ECCurve::SECP256R1)

  Construct ECDH with specified curve.
- ~ECDH ()=default
- ECCKeyPair generate_key_pair ()

  Generate a new ECDH key pair.
- ECDHResult derive_shared_secret (std::span< const uint8_t > own_private_key, std::span< const uint8_t > peer_public_key)

  Derive shared secret from own private key and peer's public key.
- std::string name () const
- std::string curve_name () const
- size_t key_size () const

Private Attributes

- ECCurve curve_
- std::string botan_curve_name_

9.37.1 Detailed Description

Elliptic Curve Diffie-Hellman (ECDH) key exchange.
Used for secure key agreement between two parties. Each party generates a key pair, exchanges public keys, and derives the same shared secret.

9.37.2 Constructor & Destructor Documentation

9.37.2.1 ECDH()

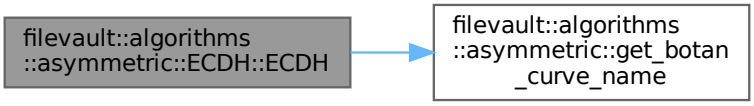
```
filevault::algorithms::asymmetric::ECDH::ECDH (
            ECCurve curve = ECCurve::SECP256R1)  [explicit]
```

Construct ECDH with specified curve.

Parameters

curve	Elliptic curve to use

Here is the call graph for this function:



9.37.2.2 ~ECDH()

```
filevault::algorithms::asymmetric::ECDH::~ECDH () [default]
```

9.37.3 Member Function Documentation

9.37.3.1 curve_name()

```
std::string filevault::algorithms::asymmetric::ECDH::curve_name () const
```

9.37.3.2 derive_shared_secret()

```
ECDHResult filevault::algorithms::asymmetric::ECDH::derive_shared_secret (
            std::span< const uint8_t > own_private_key,
            std::span< const uint8_t > peer_public_key)
```

Derive shared secret from own private key and peer's public key.

Parameters

own_private_key	Our private key
peer_public_key	Peer's public key

Returns

Shared secret that both parties can derive

Here is the call graph for this function:

9.37.3.3 generate_key_pair()

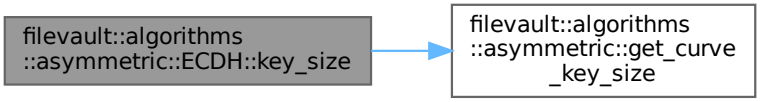
```
ECCKeyPair filevault::algorithms::asymmetric::ECDH::generate_key_pair ()
```

Generate a new ECDH key pair.

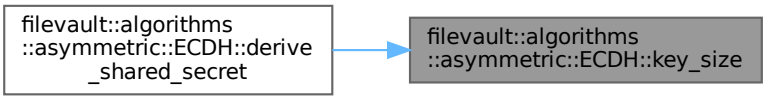
9.37.3.4 key_size()

```
size_t filevault::algorithms::asymmetric::ECDH::key_size () const
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.37.3.5 name()

```
std::string filevault::algorithms::asymmetric::ECDH::name () const
```

9.37.4 Member Data Documentation

9.37.4.1 botan_curve_name_

```
std::string filevault::algorithms::asymmetric::ECDH::botan_curve_name_  [private]
```

9.37.4.2 curve_

```
ECCurve filevault::algorithms::asymmetric::ECDH::curve_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/asymmetric/ecc.hpp
- src/algorithms/asymmetric/ecc.cpp

9.38 filevault::algorithms::asymmetric::ECDHResult Struct Reference

ECDH Key Exchange result.

```
#include <ecc.hpp>
```

Public Attributes

- bool success
- std::vector< uint8_t > shared_secret
- std::string error_message

9.38.1 Detailed Description

ECDH Key Exchange result.

9.38.2 Member Data Documentation

9.38.2.1 error_message

```
std::string filevault::algorithms::asymmetric::ECDHResult::error_message
```

9.38.2.2 shared_secret

```
std::vector<uint8_t> filevault::algorithms::asymmetric::ECDHResult::shared_secret
```

9.38.2.3 success

```
bool filevault::algorithms::asymmetric::ECDHResult::success
```

The documentation for this struct was generated from the following file:

- include/filevault/algorithms/asymmetric/ecc.hpp

9.39 filevault::algorithms::asymmetric::ECDSA Class Reference

Elliptic Curve Digital Signature Algorithm (ECDSA).
`#include <ecc.hpp>`

Public Member Functions

- ECDSA (ECCurve curve=ECCurve::SECP256R1)

  Construct ECDSA with specified curve.
- ~ECDSA ()=default
- ECCKeyPair generate_key_pair ()

  Generate a new ECDSA key pair.
- ECDSASignResult sign (std::span< const uint8_t > data, std::span< const uint8_t > private_key)

  Sign data with private key.
- bool verify (std::span< const uint8_t > data, std::span< const uint8_t > signature, std::span< const uint8_t > public_key)

  Verify signature with public key.
- std::string name () const
- std::string curve_name () const
- size_t key_size () const
- size_t signature_size () const

Private Attributes

- ECCurve curve_
- std::string botan_curve_name_

9.39.1 Detailed Description

Elliptic Curve Digital Signature Algorithm (ECDSA).
Used for digital signatures - proving authenticity and integrity.

9.39.2 Constructor & Destructor Documentation

9.39.2.1 ECDSA()

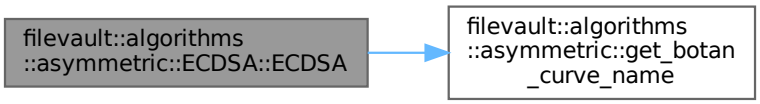
```
filevault::algorithms::asymmetric::ECDSA::ECDSA (
            ECCurve curve = ECCurve::SECP256R1)  [explicit]
```

Construct ECDSA with specified curve.

Parameters

curve	Elliptic curve to use

Here is the call graph for this function:



9.39.2.2 ~ECDSA()

```
filevault::algorithms::asymmetric::ECDSA::~ECDSA ()  [default]
```

9.39.3 Member Function Documentation

9.39.3.1 curve_name()

```
std::string filevault::algorithms::asymmetric::ECDSA::curve_name () const
```

9.39.3.2 generate_key_pair()

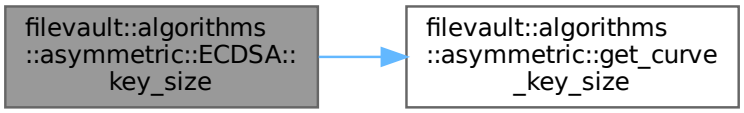
```
ECCKeyPair filevault::algorithms::asymmetric::ECDSA::generate_key_pair ()
```

Generate a new ECDSA key pair.

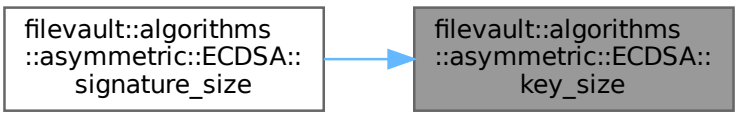
9.39.3.3 key_size()

```
size_t filevault::algorithms::asymmetric::ECDSA::key_size () const
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.39.3.4 name()

```
std::string filevault::algorithms::asymmetric::ECDSA::name () const
```

9.39.3.5 sign()

```
ECDSASignResult filevault::algorithms::asymmetric::ECDSA::sign (
            std::span< const uint8_t > data,
            std::span< const uint8_t > private_key)
```

Sign data with private key.

Parameters

data	Data to sign
private_key	ECDSA private key

Returns

Signature

9.39.3.6 signature_size()

```
size_t filevault::algorithms::asymmetric::ECDSA::signature_size () const
```

Here is the call graph for this function:

9.39.3.7 verify()

```
bool filevault::algorithms::asymmetric::ECDSA::verify (
            std::span< const uint8_t > data,
            std::span< const uint8_t > signature,
            std::span< const uint8_t > public_key)
```

Verify signature with public key.

Parameters

data	Original data
signature	Signature to verify
public_key	ECDSA public key

Returns

true if signature is valid

9.39.4 Member Data Documentation

9.39.4.1 botan_curve_name_

```
std::string filevault::algorithms::asymmetric::ECDSA::botan_curve_name_  [private]
```

9.39.4.2 curve_

```
ECCurve filevault::algorithms::asymmetric::ECDSA::curve_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/asymmetric/ecc.hpp
- src/algorithms/asymmetric/ecc.cpp

9.40 filevault::algorithms::asymmetric::ECDSASignResult Struct Reference

ECDSA signature result.

```
#include <ecc.hpp>
```

Public Attributes

- bool success
- std::vector< uint8_t > signature
- std::string error_message

9.40.1 Detailed Description

ECDSA signature result.

9.40.2 Member Data Documentation

9.40.2.1 error_message

```
std::string filevault::algorithms::asymmetric::ECDSASignResult::error_message
```

9.40.2.2 signature

```
std::vector<uint8_t> filevault::algorithms::asymmetric::ECDSASignResult::signature
```

9.40.2.3 success

```
bool filevault::algorithms::asymmetric::ECDSASignResult::success
```

The documentation for this struct was generated from the following file:

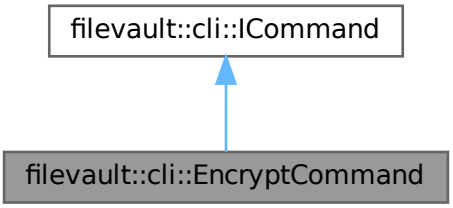
- include/filevault/algorithms/asymmetric/ecc.hpp

9.41 filevault::cli::EncryptCommand Class Reference

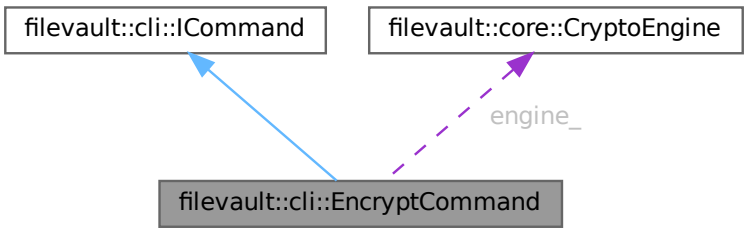
Encrypt command.

```
#include <encrypt_cmd.hpp>
```

Inheritance diagram for filevault::cli::EncryptCommand:



Collaboration diagram for filevault::cli::EncryptCommand:



Public Member Functions

- EncryptCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- core::CryptoEngine & engine_
- std::string input_file_
- std::string output_file_
- std::string password_
- std::string mode_

- std::string algorithm_ = "aes-256-gcm"
- std::string security_level_ = "medium"
- std::string kdf_ = "argon2id"
- std::string compression_type_ = "none"
- int compression_level_ = 6
- bool verbose_ = false
- bool no_progress_ = false
- bool force_weak_password_ = false

9.41.1 Detailed Description

Encrypt command.

9.41.2 Constructor & Destructor Documentation

9.41.2.1 EncryptCommand()

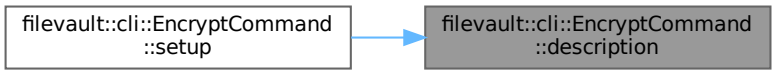
```
filevault::cli::EncryptCommand::EncryptCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.41.3 Member Function Documentation

9.41.3.1 description()

```
std::string filevault::cli::EncryptCommand::description () const  [inline], [override], [virtual]
```

Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:

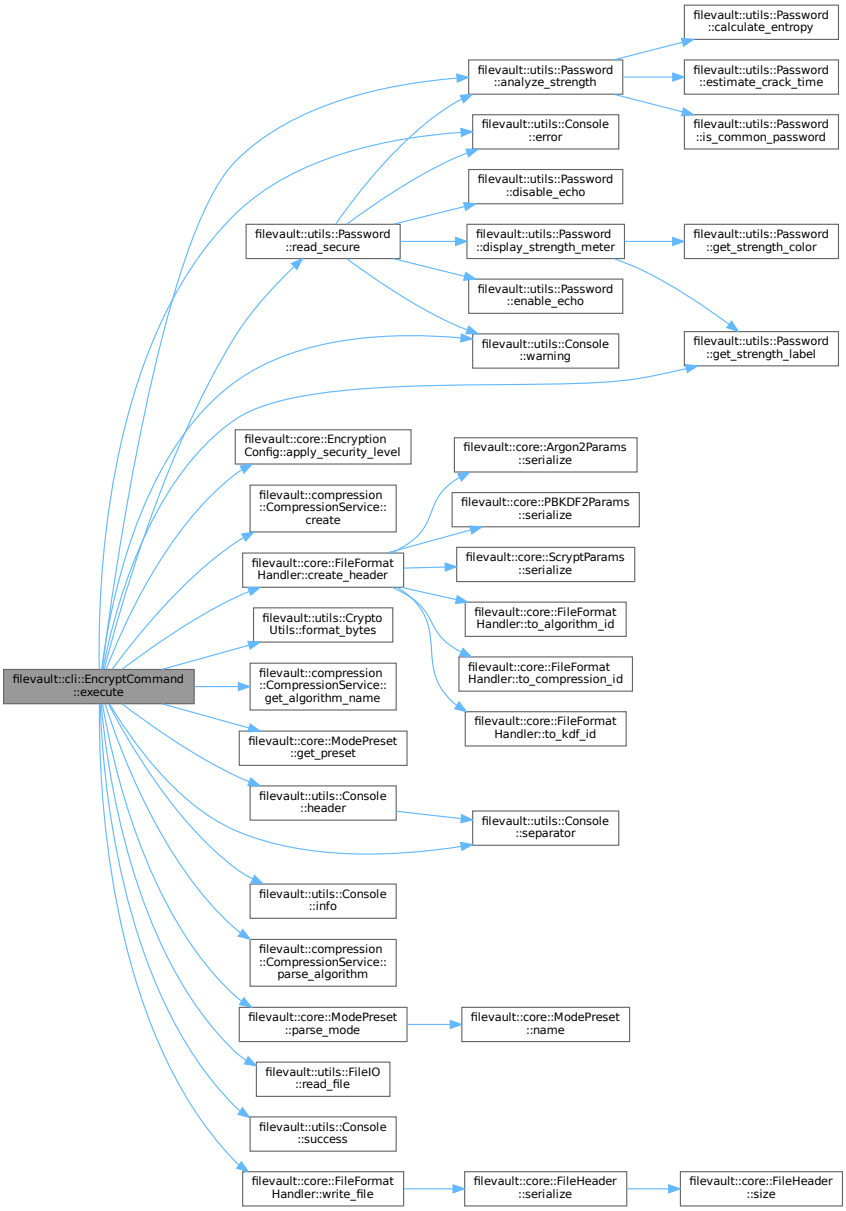


9.41.3.2 execute()

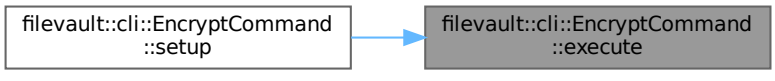
```
int filevault::cli::EncryptCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



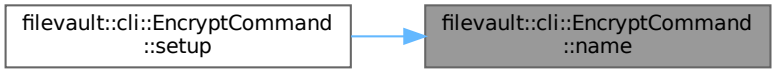
9.41.3.3 name()

`std::string filevault::cli::EncryptCommand::name () const [inline], [override], [virtual]`

Get command name.

Implements filevault::cli::ICommand.

Here is the caller graph for this function:



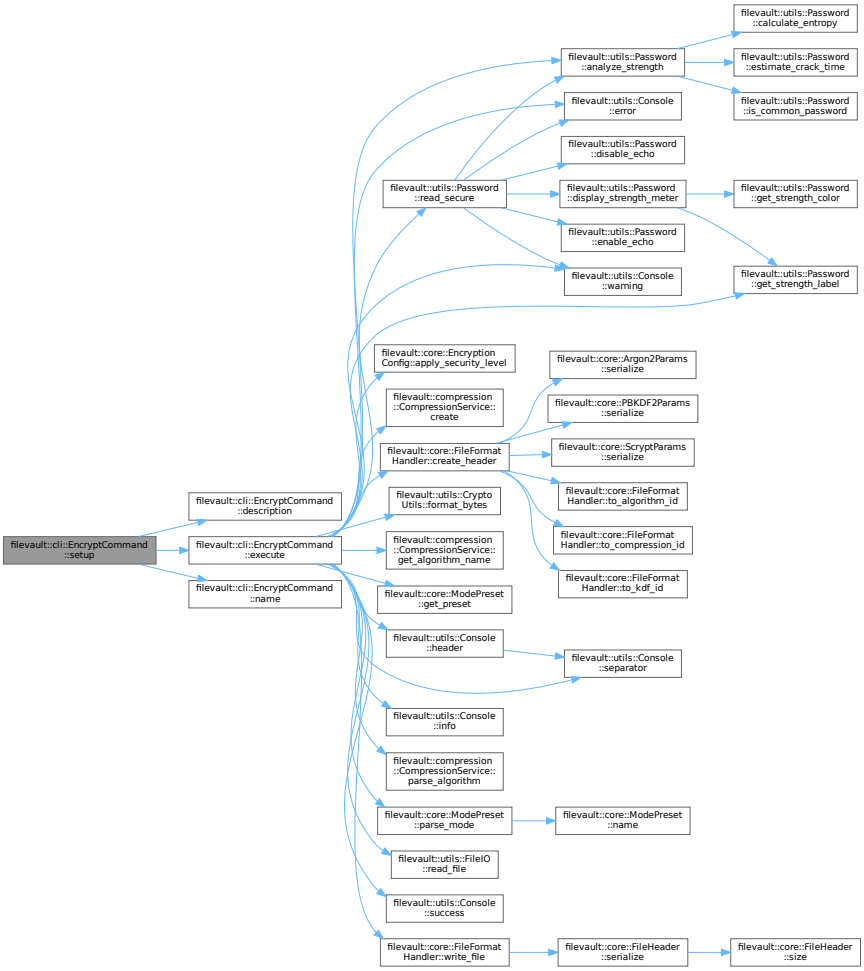
9.41.3.4 setup()

`void filevault::cli::EncryptCommand::setup (`
`CLI::App & app) [override], [virtual]`

Setup CLI11 subcommand.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



9.41.4 Member Data Documentation

9.41.4.1 algorithm_

```
std::string filevault::cli::EncryptCommand::algorithm_ = "aes-256-gcm"  [private]
```

9.41.4.2 compression_level_

```
int filevault::cli::EncryptCommand::compression_level_ = 6  [private]
```

9.41.4.3 compression_type_

```
std::string filevault::cli::EncryptCommand::compression_type_ = "none"  [private]
```

9.41.4.4 engine_

```
core::CryptoEngine& filevault::cli::EncryptCommand::engine_  [private]
```

9.41.4.5 force_weak_password_

```
bool filevault::cli::EncryptCommand::force_weak_password_ = false  [private]
```

9.41.4.6 input_file_

```
std::string filevault::cli::EncryptCommand::input_file_  [private]
```

9.41.4.7 kdf_

```
std::string filevault::cli::EncryptCommand::kdf_ = "argon2id"  [private]
```

9.41.4.8 mode_

```
std::string filevault::cli::EncryptCommand::mode_  [private]
```

9.41.4.9 no_progress_

```
bool filevault::cli::EncryptCommand::no_progress_ = false  [private]
```

9.41.4.10 output_file_

```
std::string filevault::cli::EncryptCommand::output_file_  [private]
```

9.41.4.11 password_

```
std::string filevault::cli::EncryptCommand::password_  [private]
```

9.41.4.12 security_level_

```
std::string filevault::cli::EncryptCommand::security_level_ = "medium"  [private]
```

9.41.4.13 verbose_

```
bool filevault::cli::EncryptCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/encrypt_cmd.hpp
- src/cli/commands/encrypt_cmd.cpp

9.42 filevault::core::EncryptionConfig Struct Reference

Configuration for encryption operations.
`#include <types.hpp>`

Public Member Functions

- void apply_security_level ()

  Apply security level parameters.
- void apply_user_mode ()

  Apply user mode defaults.

Public Attributes

- AlgorithmType algorithm = AlgorithmType::AES_256_GCM
- KDFType kdf = KDFType::ARGON2ID
- SecurityLevel level = SecurityLevel::MEDIUM
- UserMode mode = UserMode::PROFESSIONAL
- uint32_t kdf_iterations = 100000
- uint32_t kdf_memory_kb = 65536
- uint32_t kdf_parallelism = 4
- std::vector< uint8_t > salt
- std::optional< std::vector< uint8_t > > nonce
- std::optional< std::vector< uint8_t > > tag
- std::optional< std::vector< uint8_t > > associated_data
- CompressionType compression = CompressionType::NONE
- int compression_level = 6
- bool include_metadata = true
- std::string comment
- bool show_progress = true
- bool verbose = false

9.42.1 Detailed Description

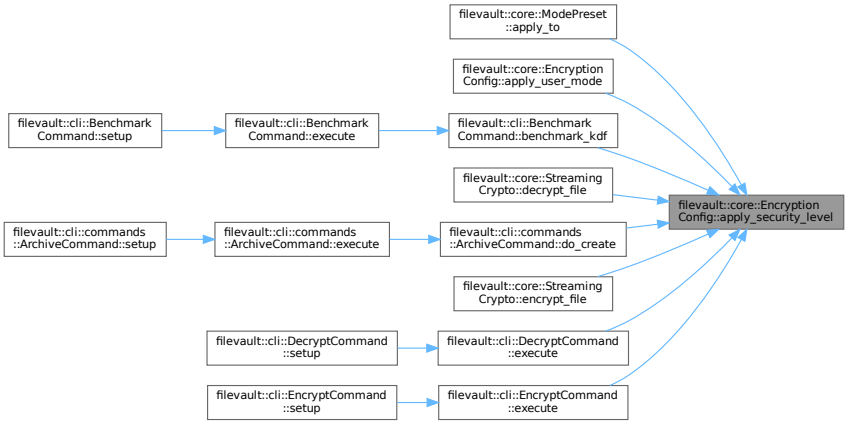
Configuration for encryption operations.

9.42.2 Member Function Documentation

9.42.2.1 apply_security_level()

`void filevault::core::EncryptionConfig::apply_security_level ()`
Apply security level parameters.

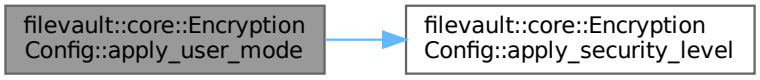
Here is the caller graph for this function:



9.42.2.2 apply_user_mode()

```
void filevault::core::EncryptionConfig::apply_user_mode ()
```

Apply user mode defaults.
Here is the call graph for this function:



9.42.3 Member Data Documentation

9.42.3.1 algorithm

```
AlgorithmType filevault::core::EncryptionConfig::algorithm = AlgorithmType::AES_256_GCM
```

9.42.3.2 associated_data

```
std::optional<std::vector<uint8_t> > filevault::core::EncryptionConfig::associated_data
```

9.42.3.3 comment

```
std::string filevault::core::EncryptionConfig::comment
```

9.42.3.4 compression

```
CompressionType filevault::core::EncryptionConfig::compression = CompressionType::NONE
```

9.42.3.5 compression_level

```
int filevault::core::EncryptionConfig::compression_level = 6
```

9.42.3.6 include_metadata

```
bool filevault::core::EncryptionConfig::include_metadata = true
```

9.42.3.7 kdf

```
KDFType filevault::core::EncryptionConfig::kdf = KDFType::ARGON2ID
```

9.42.3.8 kdf_iterations

```
uint32_t filevault::core::EncryptionConfig::kdf_iterations = 100000
```

9.42.3.9 kdf_memory_kb

```
uint32_t filevault::core::EncryptionConfig::kdf_memory_kb = 65536
```

9.42.3.10 kdf_parallelism

```
uint32_t filevault::core::EncryptionConfig::kdf_parallelism = 4
```

9.42.3.11 level

```
SecurityLevel filevault::core::EncryptionConfig::level = SecurityLevel::MEDIUM
```

9.42.3.12 mode

```
UserMode filevault::core::EncryptionConfig::mode = UserMode::PROFESSIONAL
```

9.42.3.13 nonce

```
std::optional<std::vector<uint8_t> > filevault::core::EncryptionConfig::nonce
```

9.42.3.14 salt

```
std::vector<uint8_t> filevault::core::EncryptionConfig::salt
```

9.42.3.15 show_progress

```
bool filevault::core::EncryptionConfig::show_progress = true
```

9.42.3.16 tag

```
std::optional<std::vector<uint8_t> > filevault::core::EncryptionConfig::tag
```

9.42.3.17 verbose

```
bool filevault::core::EncryptionConfig::verbose = false
```

The documentation for this struct was generated from the following files:

- include/filevault/core/types.hpp
- src/core/types.cpp

9.43 filevault::archive::FileEntry Struct Reference

Archive file entry metadata.

`#include <archive_format.hpp>`

Public Member Functions

- std::vector< uint8_t > serialize () const

Static Public Member Functions

- static FileEntry deserialize (std::span< const uint8_t > data, size_t &offset)

Public Attributes

- std::string filename
- uint64_t file_size
- uint64_t offset
- uint64_t modified_time
- uint32_t permissions

9.43.1 Detailed Description

Archive file entry metadata.

9.43.2 Member Function Documentation

9.43.2.1 deserialize()

```
FileEntry filevault::archive::FileEntry::deserialize (
            std::span< const uint8_t > data,
            size_t & offset)  [static]
```

Here is the caller graph for this function:

9.43.2.2 serialize()

```
std::vector< uint8_t > filevault::archive::FileEntry::serialize () const
```

9.43.3 Member Data Documentation

9.43.3.1 file_size

```
uint64_t filevault::archive::FileEntry::file_size
```

9.43.3.2 filename

```
std::string filevault::archive::FileEntry::filename
```

9.43.3.3 modified_time

```
uint64_t filevault::archive::FileEntry::modified_time
```

9.43.3.4 offset

```
uint64_t filevault::archive::FileEntry::offset
```

9.43.3.5 permissions

```
uint32_t filevault::archive::FileEntry::permissions
```

The documentation for this struct was generated from the following files:

- include/filevault/archive/archive_format.hpp
- src/archive/archive_format.cpp

9.44 filevault::core::FileFormatHandler Class Reference

File format handler.

`#include <file_format.hpp>`

Static Public Member Functions

- static FileHeader create_header (AlgorithmType algo_type, KDFType kdf_type, const EncryptionConfig &config, std::span< const uint8_t > salt, std::span< const uint8_t > nonce, bool compressed)

  Create header from encryption config.
- static bool write_file (const std::string &path, const FileHeader &header, std::span< const uint8_t > ciphertext, std::span< const uint8_t > auth_tag)

  Write encrypted file with header.
- static std::tuple< FileHeader, std::vector< uint8_t >, std::vector< uint8_t > > read_file (const std::string &path)

  Read encrypted file and parse header.
- static AlgorithmID to_algorithm_id (AlgorithmType type)

  Convert AlgorithmType to AlgorithmID.
- static AlgorithmType from_algorithm_id (AlgorithmID id)

  Convert AlgorithmID to AlgorithmType.
- static KDFID to_kdf_id (KDFType type)

  Convert KDFType to KDFID.
- static KDFType from_kdf_id (KDFID id)

  Convert KDFID to KDFType.
- static CompressionID to_compression_id (const std::string &type)

  Convert compression type to CompressionID.
- static std::string from_compression_id (CompressionID id)

  Convert CompressionID to string.
- static bool is_legacy_format (const std::string &path)

  Detect if file is legacy format (no magic bytes).
- static std::tuple< std::vector< uint8_t >, std::vector< uint8_t >, std::vector< uint8_t > > read_legacy_file (const std::string &path)

  Read legacy format file (old format without header).

9.44.1 Detailed Description

File format handler.

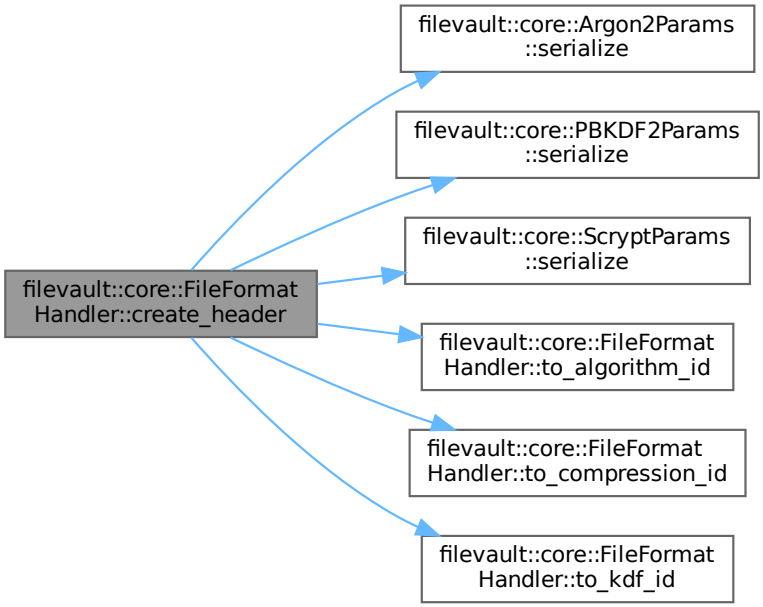
9.44.2 Member Function Documentation

9.44.2.1 create_header()

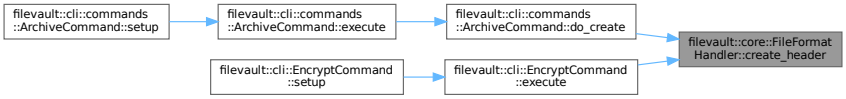
```
FileHeader filevault::core::FileFormatHandler::create_header (
            AlgorithmType algo_type,
            KDFType kdf_type,
            const EncryptionConfig & config,
            std::span< const uint8_t > salt,
            std::span< const uint8_t > nonce,
            bool compressed)  [static]
```

Create header from encryption config.

Here is the call graph for this function:



Here is the caller graph for this function:

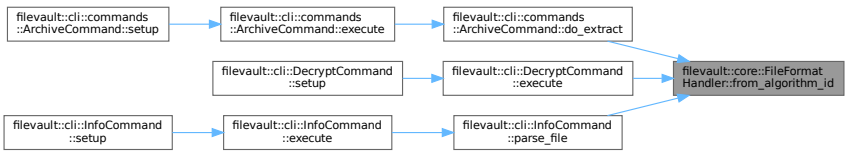


9.44.2.2 from_algorithm_id()

```
AlgorithmType filevault::core::FileFormatHandler::from_algorithm_id (
            AlgorithmID id)  [static]
```

Convert AlgorithmID to AlgorithmType.

Here is the caller graph for this function:



9.44.2.3 from_compression_id()

```
std::string filevault::core::FileFormatHandler::from_compression_id (
```

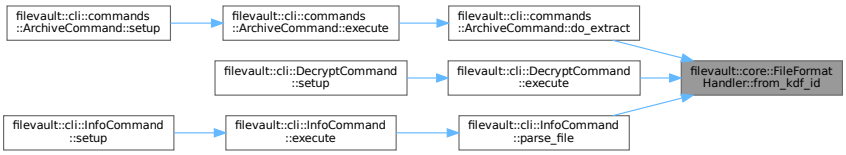
```
            CompressionID id)  [static]
```

Convert CompressionID to string.
Here is the caller graph for this function:

9.44.2.4 from_kdf_id()

```
KDFType filevault::core::FileFormatHandler::from_kdf_id (
            KDFID id)  [static]
```

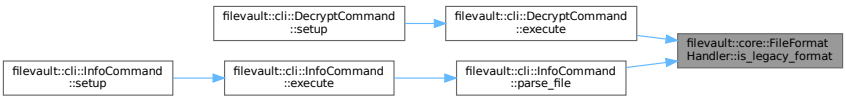
Convert KDFID to KDFType.
Here is the caller graph for this function:



9.44.2.5 is_legacy_format()

```
bool filevault::core::FileFormatHandler::is_legacy_format (
            const std::string & path)  [static]
```

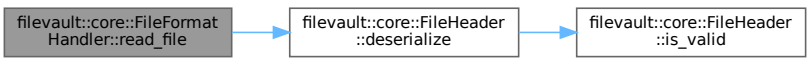
Detect if file is legacy format (no magic bytes).
Here is the caller graph for this function:



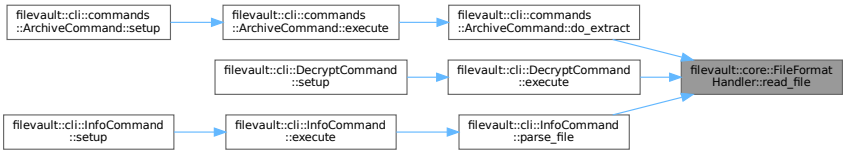
9.44.2.6 read_file()

```
std::tuple< FileHeader, std::vector< uint8_t >, std::vector< uint8_t > > filevault::core::↩
FileFormatHandler::read_file (
            const std::string & path)  [static]
```

Read encrypted file and parse header.
Here is the call graph for this function:



Here is the caller graph for this function:



9.44.2.7 read_legacy_file()

```
std::tuple< std::vector< uint8_t >, std::vector< uint8_t >, std::vector< uint8_t > > filevault↩
::core::FileFormatHandler::read_legacy_file (
            const std::string & path)  [static]
```

Read legacy format file (old format without header).

9.44.2.8 to_algorithm_id()

```
AlgorithmID filevault::core::FileFormatHandler::to_algorithm_id (
            AlgorithmType type)  [static]
```

Convert AlgorithmType to AlgorithmID.
Here is the caller graph for this function:

9.44.2.9 to_compression_id()

```
CompressionID filevault::core::FileFormatHandler::to_compression_id (
            const std::string & type)  [static]
```

Convert compression type to CompressionID.
Here is the caller graph for this function:

9.44.2.10 to_kdf_id()

```
KDFID filevault::core::FileFormatHandler::to_kdf_id (
            KDFType type)  [static]
```

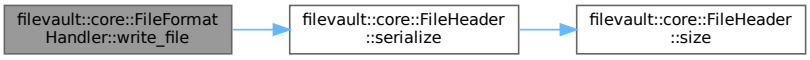
Convert KDFType to KDFID.

Here is the caller graph for this function:

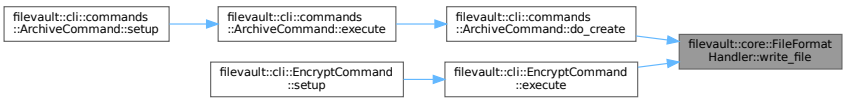
9.44.2.11 write_file()

```
bool filevault::core::FileFormatHandler::write_file (
            const std::string & path,
            const FileHeader & header,
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > auth_tag)  [static]
```

Write encrypted file with header.
Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/core/file_format.hpp
- src/format/file_format.cpp

9.45 filevault::core::FileHeader Struct Reference

File format header.
`#include <file_format.hpp>`

Public Member Functions

- bool is_valid () const

  Check if magic bytes are valid.
- size_t size () const

  Get total header size in bytes.
- std::vector< uint8_t > serialize () const

  Serialize header to bytes.

Static Public Member Functions

- static std::pair< FileHeader, size_t > deserialize (std::span< const uint8_t > data)

  Deserialize header from bytes.

Public Attributes

- uint8_t magic [8]
- uint8_t version_major
- uint8_t version_minor
- AlgorithmID algorithm
- KDFID kdf
- CompressionID compression
- uint8_t reserved [3]
- std::vector< uint8_t > salt
- std::vector< uint8_t > kdf_params
- std::vector< uint8_t > nonce
- bool compressed

9.45.1 Detailed Description

File format header.

9.45.2 Member Function Documentation

9.45.2.1 deserialize()

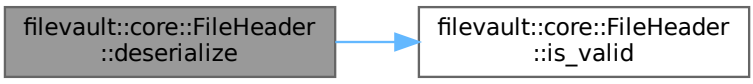
```
std::pair< FileHeader, size_t > filevault::core::FileHeader::deserialize (
            std::span< const uint8_t > data)  [static]
```

Deserialize header from bytes.

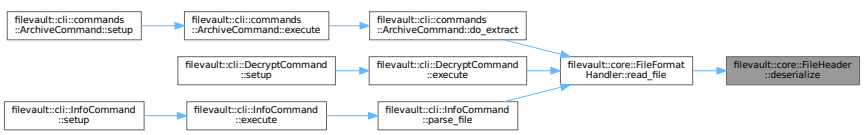
Returns

Header and number of bytes consumed

Here is the call graph for this function:

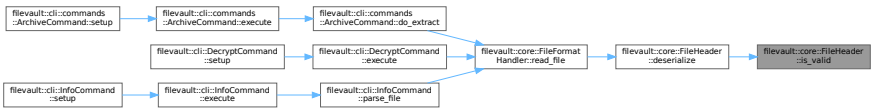


Here is the caller graph for this function:



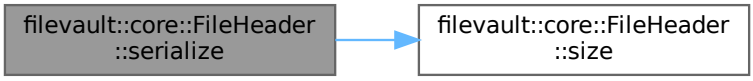
9.45.2.2 is_valid()

`bool filevault::core::FileHeader::is_valid () const`
Check if magic bytes are valid.
Here is the caller graph for this function:



9.45.2.3 serialize()

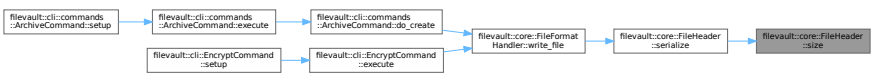
`std::vector< uint8_t > filevault::core::FileHeader::serialize () const`
Serialize header to bytes.
Here is the call graph for this function:



Here is the caller graph for this function:

9.45.2.4 size()

`size_t filevault::core::FileHeader::size () const`
Get total header size in bytes.
Here is the caller graph for this function:



9.45.3 Member Data Documentation

9.45.3.1 algorithm

`AlgorithmID filevault::core::FileHeader::algorithm`

9.45.3.2 compressed

```
bool filevault::core::FileHeader::compressed
```

9.45.3.3 compression

```
CompressionID filevault::core::FileHeader::compression
```

9.45.3.4 kdf

```
KDFID filevault::core::FileHeader::kdf
```

9.45.3.5 kdf_params

```
std::vector<uint8_t> filevault::core::FileHeader::kdf_params
```

9.45.3.6 magic

```
uint8_t filevault::core::FileHeader::magic[8]
```

9.45.3.7 nonce

```
std::vector<uint8_t> filevault::core::FileHeader::nonce
```

9.45.3.8 reserved

```
uint8_t filevault::core::FileHeader::reserved[3]
```

9.45.3.9 salt

```
std::vector<uint8_t> filevault::core::FileHeader::salt
```

9.45.3.10 version_major

```
uint8_t filevault::core::FileHeader::version_major
```

9.45.3.11 version_minor

```
uint8_t filevault::core::FileHeader::version_minor
```

The documentation for this struct was generated from the following files:

- include/filevault/core/file_format.hpp
- src/format/file_format.cpp

9.46 filevault::format::FileHeader Class Reference

FileVault encrypted file format.

`#include <file_header.hpp>`

Public Member Functions

- FileHeader ()
- void set_algorithm (core::AlgorithmType algo)
- void set_kdf (core::KDFType kdf)
- void set_security_level (core::SecurityLevel level)
- void set_salt (const std::vector< uint8_t > &salt)
- void set_nonce (const std::vector< uint8_t > &nonce)
- void set_tag (const std::vector< uint8_t > &tag)
- void set_original_size (uint64_t size)

- void set_encrypted_size (uint64_t size)
- void set_timestamp (uint64_t ts)
- void set_compressed (bool compressed)
- core::AlgorithmType algorithm () const
- core::KDFType kdf () const
- core::SecurityLevel security_level () const
- const std::vector< uint8_t > & salt () const
- const std::vector< uint8_t > & nonce () const
- const std::vector< uint8_t > & tag () const
- uint64_t original_size () const
- uint64_t encrypted_size () const
- uint64_t timestamp () const
- bool is_compressed () const
- uint32_t flags () const
- std::vector< uint8_t > serialize () const
- bool validate () const
- size_t total_size () const

Static Public Member Functions

- static core::Result< FileHeader > deserialize (std::span< const uint8_t > data)

Static Public Attributes

- static constexpr uint32_t MAGIC = 0x544C5646
- static constexpr uint8_t VERSION_MAJOR = 1
- static constexpr uint8_t VERSION_MINOR = 0
- static constexpr size_t MIN_HEADER_SIZE = 64
- static constexpr uint32_t FLAG_COMPRESSED = 0x00000001
- static constexpr uint32_t FLAG_METADATA = 0x00000002

Static Private Member Functions

- static void write_uint16 (std::vector< uint8_t > &buf, uint16_t val)
- static void write_uint32 (std::vector< uint8_t > &buf, uint32_t val)
- static void write_uint64 (std::vector< uint8_t > &buf, uint64_t val)
- static uint16_t read_uint16 (std::span< const uint8_t > data, size_t &offset)
- static uint32_t read_uint32 (std::span< const uint8_t > data, size_t &offset)
- static uint64_t read_uint64 (std::span< const uint8_t > data, size_t &offset)

Private Attributes

- core::AlgorithmType algorithm_
- core::KDFType kdf_
- core::SecurityLevel security_level_
- std::vector< uint8_t > salt_
- std::vector< uint8_t > nonce_
- std::vector< uint8_t > tag_
- uint64_t original_size_
- uint64_t encrypted_size_
- uint64_t timestamp_
- uint32_t flags_
- std::array< uint8_t, 16 > reserved_

9.46.1 Detailed Description

FileVault encrypted file format.
Structure:

- Magic bytes (4): "FVLT"
- Version (2): major.minor
- Algorithm (1): AlgorithmType enum
- KDF (1): KDFType enum
- Security level (1): SecurityLevel enum
- Salt length (2): uint16_t
- Salt (variable)
- Nonce length (2): uint16_t
- Nonce (variable)
- Tag length (2): uint16_t
- Tag (variable, for AEAD)
- Original size (8): uint64_t
- Encrypted size (8): uint64_t
- Timestamp (8): uint64_t (Unix epoch)
- Flags (4): compression, etc.
- Reserved (16): future use
- Encrypted data (variable)

9.46.2 Constructor & Destructor Documentation

9.46.2.1 FileHeader()

```
filevault::format::FileHeader::FileHeader ()
```

Here is the caller graph for this function:

9.46.3 Member Function Documentation

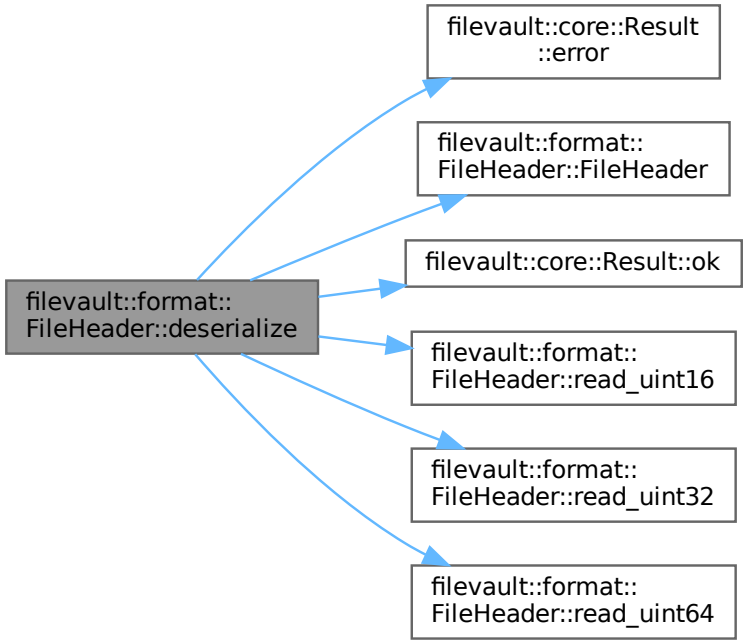
9.46.3.1 algorithm()

```
core::AlgorithmType filevault::format::FileHeader::algorithm () const  [inline]
```

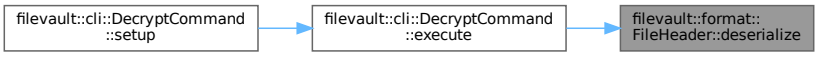
9.46.3.2 deserialize()

```
core::Result< FileHeader > filevault::format::FileHeader::deserialize (
            std::span< const uint8_t > data)  [static]
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.46.3.3 encrypted_size()

```
uint64_t filevault::format::FileHeader::encrypted_size () const  [inline]
```

9.46.3.4 flags()

```
uint32_t filevault::format::FileHeader::flags () const  [inline]
```

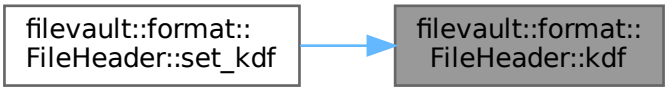
9.46.3.5 is_compressed()

```
bool filevault::format::FileHeader::is_compressed () const  [inline]
```

9.46.3.6 kdf()

```
core::KDFType filevault::format::FileHeader::kdf () const  [inline]
```

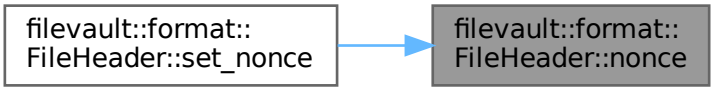
Here is the caller graph for this function:



9.46.3.7 nonce()

```
const std::vector< uint8_t > & filevault::format::FileHeader::nonce () const  [inline]
```

Here is the caller graph for this function:



9.46.3.8 original_size()

```
uint64_t filevault::format::FileHeader::original_size () const  [inline]
```

9.46.3.9 read_uint16()

```
uint16_t filevault::format::FileHeader::read_uint16 (
            std::span< const uint8_t > data,
            size_t & offset)  [static], [private]
```

Here is the caller graph for this function:

9.46.3.10 read_uint32()

```
uint32_t filevault::format::FileHeader::read_uint32 (
            std::span< const uint8_t > data,
            size_t & offset)  [static], [private]
```

Here is the caller graph for this function:

9.46.3.11 read_uint64()

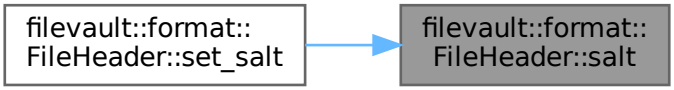
```
uint64_t filevault::format::FileHeader::read_uint64 (
            std::span< const uint8_t > data,
            size_t & offset)  [static], [private]
```

Here is the caller graph for this function:

9.46.3.12 salt()

```
const std::vector< uint8_t > & filevault::format::FileHeader::salt () const  [inline]
```

Here is the caller graph for this function:



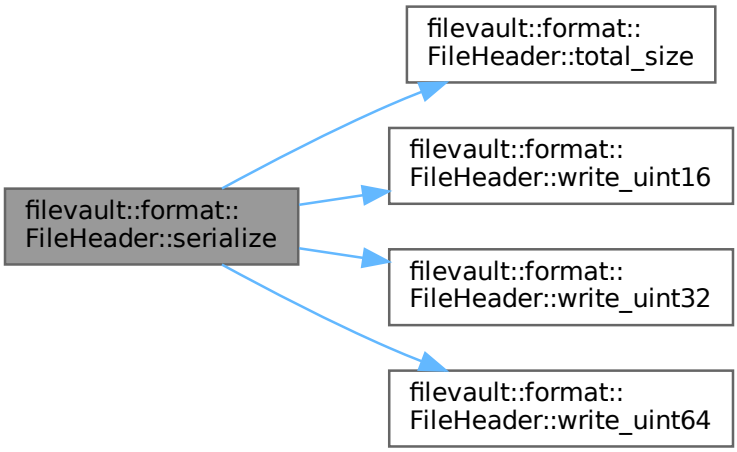
9.46.3.13 security_level()

```
core::SecurityLevel filevault::format::FileHeader::security_level () const  [inline]
```

9.46.3.14 serialize()

```
std::vector< uint8_t > filevault::format::FileHeader::serialize () const
```

Here is the call graph for this function:



9.46.3.15 set_algorithm()

```
void filevault::format::FileHeader::set_algorithm (
            core::AlgorithmType algo)  [inline]
```

9.46.3.16 set_compressed()

```
void filevault::format::FileHeader::set_compressed (
            bool compressed)
```

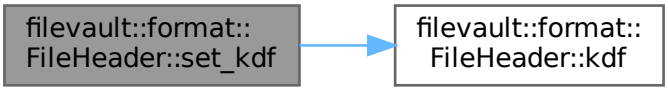
9.46.3.17 set_encrypted_size()

```
void filevault::format::FileHeader::set_encrypted_size (
            uint64_t size)  [inline]
```

9.46.3.18 set_kdf()

```
void filevault::format::FileHeader::set_kdf (
            core::KDFType kdf)  [inline]
```

Here is the call graph for this function:



9.46.3.19 set_nonce()

```
void filevault::format::FileHeader::set_nonce (
            const std::vector< uint8_t > & nonce)  [inline]
```

Here is the call graph for this function:

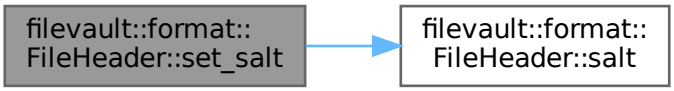
9.46.3.20 set_original_size()

```
void filevault::format::FileHeader::set_original_size (
            uint64_t size)  [inline]
```

9.46.3.21 set_salt()

```
void filevault::format::FileHeader::set_salt (
            const std::vector< uint8_t > & salt)  [inline]
```

Here is the call graph for this function:



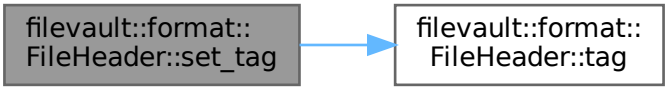
9.46.3.22 set_security_level()

```
void filevault::format::FileHeader::set_security_level (
            core::SecurityLevel level)  [inline]
```

9.46.3.23 set_tag()

```
void filevault::format::FileHeader::set_tag (
            const std::vector< uint8_t > & tag)  [inline]
```

Here is the call graph for this function:



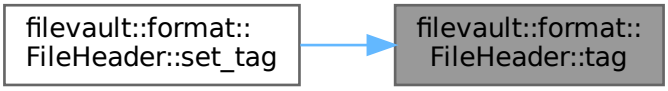
9.46.3.24 set_timestamp()

```
void filevault::format::FileHeader::set_timestamp (
            uint64_t ts)  [inline]
```

9.46.3.25 tag()

```
const std::vector< uint8_t > & filevault::format::FileHeader::tag () const  [inline]
```

Here is the caller graph for this function:



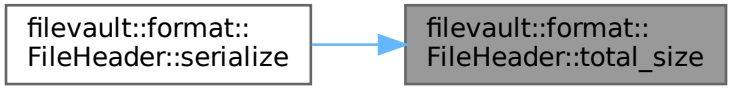
9.46.3.26 timestamp()

```
uint64_t filevault::format::FileHeader::timestamp () const  [inline]
```

9.46.3.27 total_size()

```
size_t filevault::format::FileHeader::total_size () const
```

Here is the caller graph for this function:



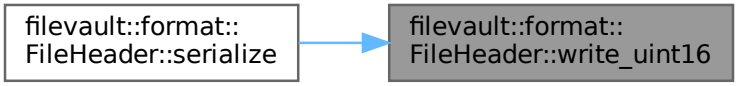
9.46.3.28 validate()

```
bool filevault::format::FileHeader::validate () const
```

9.46.3.29 write_uint16()

```
void filevault::format::FileHeader::write_uint16 (
            std::vector< uint8_t > & buf,
            uint16_t val)  [static], [private]
```

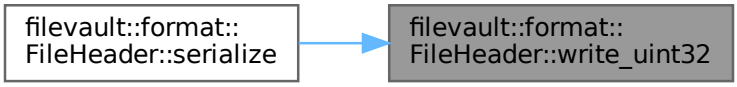
Here is the caller graph for this function:



9.46.3.30 write_uint32()

```
void filevault::format::FileHeader::write_uint32 (
            std::vector< uint8_t > & buf,
            uint32_t val)  [static], [private]
```

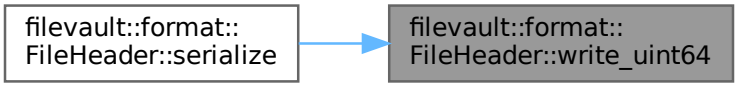
Here is the caller graph for this function:



9.46.3.31 write_uint64()

```
void filevault::format::FileHeader::write_uint64 (
            std::vector< uint8_t > & buf,
            uint64_t val)  [static], [private]
```

Here is the caller graph for this function:



9.46.4 Member Data Documentation

9.46.4.1 algorithm_

```
core::AlgorithmType filevault::format::FileHeader::algorithm_ [private]
```

9.46.4.2 encrypted_size_

```
uint64_t filevault::format::FileHeader::encrypted_size_ [private]
```

9.46.4.3 FLAG_COMPRESSED

```
uint32_t filevault::format::FileHeader::FLAG_COMPRESSED = 0x00000001 [static], [constexpr]
```

9.46.4.4 FLAG_METADATA

```
uint32_t filevault::format::FileHeader::FLAG_METADATA = 0x00000002 [static], [constexpr]
```

9.46.4.5 flags_

```
uint32_t filevault::format::FileHeader::flags_ [private]
```

9.46.4.6 kdf_

```
core::KDFType filevault::format::FileHeader::kdf_ [private]
```

9.46.4.7 MAGIC

```
uint32_t filevault::format::FileHeader::MAGIC = 0x544C5646 [static], [constexpr]
```

9.46.4.8 MIN_HEADER_SIZE

```
size_t filevault::format::FileHeader::MIN_HEADER_SIZE = 64 [static], [constexpr]
```

9.46.4.9 nonce_

```
std::vector<uint8_t> filevault::format::FileHeader::nonce_ [private]
```

9.46.4.10 original_size_

```
uint64_t filevault::format::FileHeader::original_size_ [private]
```

9.46.4.11 reserved_

```
std::array<uint8_t, 16> filevault::format::FileHeader::reserved_ [private]
```

9.46.4.12 salt_

```
std::vector<uint8_t> filevault::format::FileHeader::salt_ [private]
```

9.46.4.13 security_level_

```
core::SecurityLevel filevault::format::FileHeader::security_level_ [private]
```

9.46.4.14 tag_

```
std::vector<uint8_t> filevault::format::FileHeader::tag_ [private]
```

9.46.4.15 timestamp_

```
uint64_t filevault::format::FileHeader::timestamp_ [private]
```

9.46.4.16 VERSION_MAJOR

```
uint8_t filevault::format::FileHeader::VERSION_MAJOR = 1  [static], [constexpr]
```

9.46.4.17 VERSION_MINOR

```
uint8_t filevault::format::FileHeader::VERSION_MINOR = 0  [static], [constexpr]
```

The documentation for this class was generated from the following files:

- include/filevault/format/file_header.hpp
- src/format/file_header.cpp

9.47 filevault::cli::InfoCommand::FileInfo Struct Reference

Parse encrypted file header.

Public Attributes

- size_t file_size = 0
- size_t salt_size = 0
- size_t nonce_size = 0
- size_t tag_size = 0
- size_t data_size = 0
- size_t header_size = 0
- bool has_header = false
- std::string version
- std::string algorithm
- std::string kdf
- std::string compression
- bool compressed = false

9.47.1 Detailed Description

Parse encrypted file header.

9.47.2 Member Data Documentation

9.47.2.1 algorithm

```
std::string filevault::cli::InfoCommand::FileInfo::algorithm
```

9.47.2.2 compressed

```
bool filevault::cli::InfoCommand::FileInfo::compressed = false
```

9.47.2.3 compression

```
std::string filevault::cli::InfoCommand::FileInfo::compression
```

9.47.2.4 data_size

```
size_t filevault::cli::InfoCommand::FileInfo::data_size = 0
```

9.47.2.5 file_size

```
size_t filevault::cli::InfoCommand::FileInfo::file_size = 0
```

9.47.2.6 has_header

```
bool filevault::cli::InfoCommand::FileInfo::has_header = false
```

9.47.2.7 header_size

```
size_t filevault::cli::InfoCommand::FileInfo::header_size = 0
```

9.47.2.8 kdf

```
std::string filevault::cli::InfoCommand::FileInfo::kdf
```

9.47.2.9 nonce_size

```
size_t filevault::cli::InfoCommand::FileInfo::nonce_size = 0
```

9.47.2.10 salt_size

```
size_t filevault::cli::InfoCommand::FileInfo::salt_size = 0
```

9.47.2.11 tag_size

```
size_t filevault::cli::InfoCommand::FileInfo::tag_size = 0
```

9.47.2.12 version

```
std::string filevault::cli::InfoCommand::FileInfo::version
```

The documentation for this struct was generated from the following file:

- include/filevault/cli/commands/info_cmd.hpp

9.48 filevault::utils::FileIO Class Reference

File I/O utilities.

`#include <file_io.hpp>`

Static Public Member Functions

- static core::Result< std::vector< uint8_t > > read_file (const std::string &path)

  Read entire file into memory.
- static core::Result< void > write_file (const std::string &path, std::span< const uint8_t > data)

  Write data to file.
- static bool file_exists (const std::string &path)

  Check if file exists.
- static size_t file_size (const std::string &path)

  Get file size.

9.48.1 Detailed Description

File I/O utilities.

9.48.2 Member Function Documentation

9.48.2.1 file_exists()

```
bool filevault::utils::FileIO::file_exists (
            const std::string & path)  [static]
```

Check if file exists.

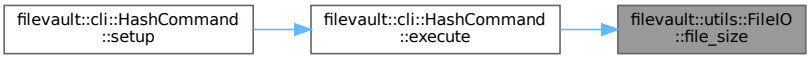
Here is the caller graph for this function:

9.48.2.2 file_size()

```
size_t filevault::utils::FileIO::file_size (
            const std::string & path)  [static]
```

Get file size.
Here is the call graph for this function:

Here is the caller graph for this function:

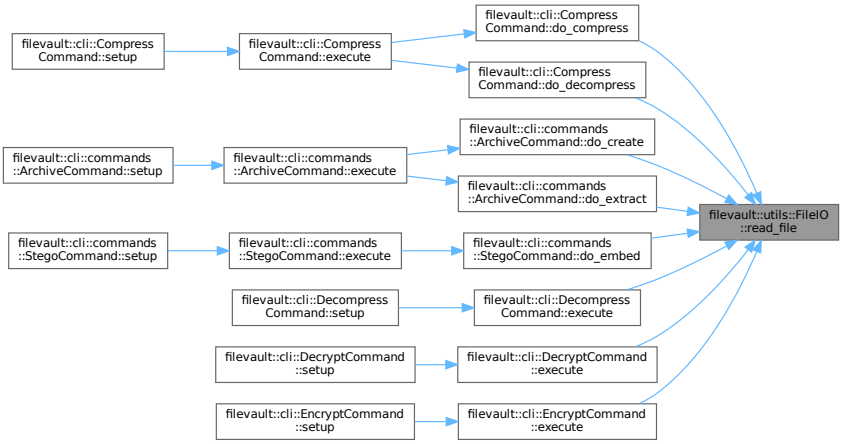


9.48.2.3 read_file()

```
core::Result< std::vector< uint8_t > > filevault::utils::FileIO::read_file (
            const std::string & path)  [static]
```

Read entire file into memory.

Here is the caller graph for this function:

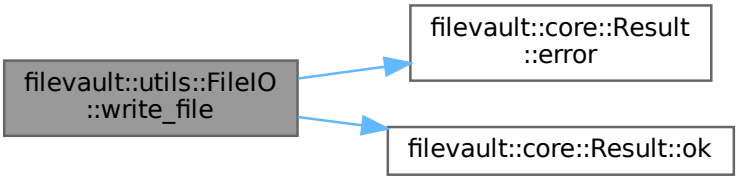


9.48.2.4 write_file()

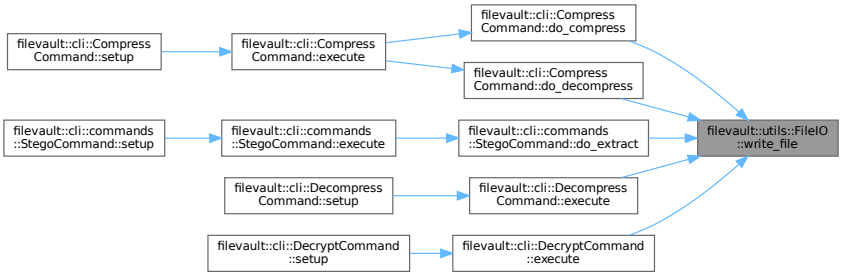
```
core::Result< void > filevault::utils::FileIO::write_file (
            const std::string & path,
            std::span< const uint8_t > data)  [static]
```

Write data to file.

Here is the call graph for this function:



Here is the caller graph for this function:

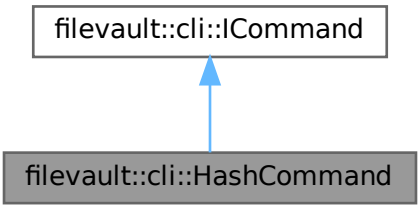


The documentation for this class was generated from the following files:

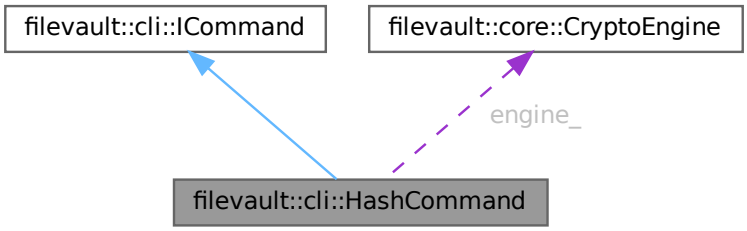
- include/filevault/utils/file_io.hpp
- src/utils/file_io.cpp

9.49 filevault::cli::HashCommand Class Reference

Hash command - Calculate cryptographic hashes.
`#include <hash_cmd.hpp>`
Inheritance diagram for filevault::cli::HashCommand:



Collaboration diagram for filevault::cli::HashCommand:



Public Member Functions

- HashCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- std::string get_botan_algorithm_name (const std::string &algo)
- bool is_secure_algorithm (const std::string &algo)
- std::string calculate_file_hash (const std::string &filepath, const std::string &algorithm)
- std::string calculate_file_hmac (const std::string &filepath, const std::string &hash_algorithm, const std::string &key)
- int verify_mode (const std::string &calculated_hash)

Private Attributes

- core::CryptoEngine & engine_
- std::string input_file_
- std::string output_file_
- std::string algorithm_ = "sha256"
- std::string output_format_ = "hex"
- std::string verify_hash_
- std::string hmac_key_
- bool uppercase_ = false
- bool no_filename_ = false
- bool verbose_ = false
- bool benchmark_ = false

9.49.1 Detailed Description

Hash command - Calculate cryptographic hashes.
Supports: MD5, SHA1, SHA2 family (224/256/384/512), SHA3 family (224/256/384/512), BLAKE2 (b/s variants)
Features:

- HMAC mode with key
- Hash verification
- Batch processing
- Performance benchmarking

9.49.2 Constructor & Destructor Documentation

9.49.2.1 HashCommand()

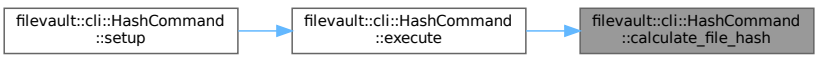
```
filevault::cli::HashCommand::HashCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.49.3 Member Function Documentation

9.49.3.1 calculate_file_hash()

```
std::string filevault::cli::HashCommand::calculate_file_hash (
            const std::string & filepath,
            const std::string & algorithm)  [private]
```

Here is the caller graph for this function:



9.49.3.2 calculate_file_hmac()

```
std::string filevault::cli::HashCommand::calculate_file_hmac (
            const std::string & filepath,
            const std::string & hash_algorithm,
            const std::string & key)  [private]
```

Here is the caller graph for this function:

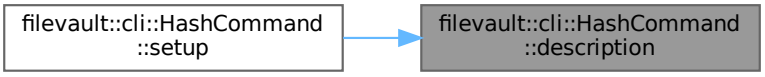
9.49.3.3 description()

```
std::string filevault::cli::HashCommand::description () const  [inline], [override], [virtual]
```

Get command description.

Implements filevault::cli::ICommand.

Here is the caller graph for this function:



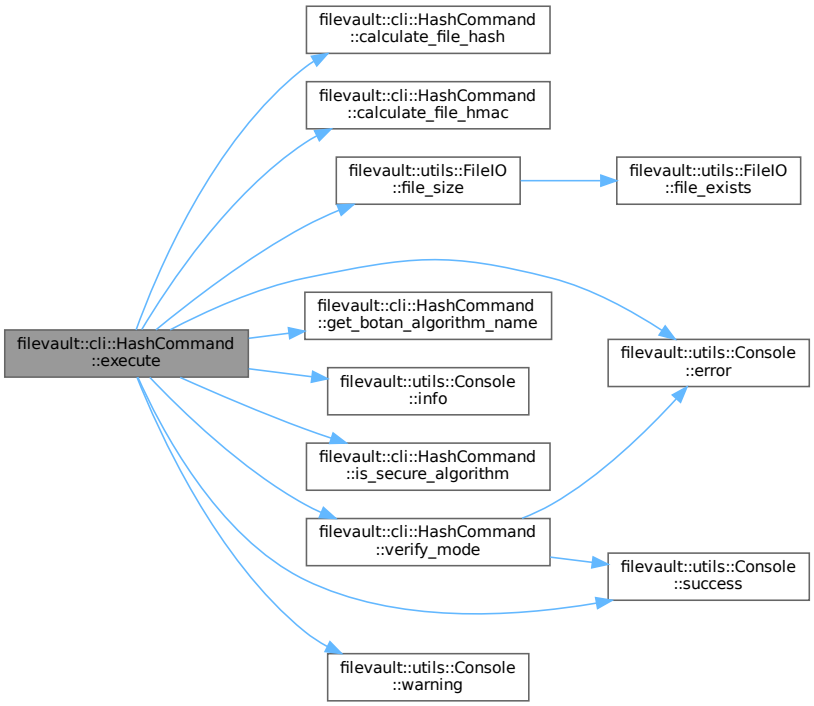
9.49.3.4 execute()

```
int filevault::cli::HashCommand::execute ()  [override], [virtual]
```

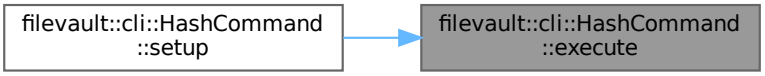
Execute the command.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



9.49.3.5 get_botan_algorithm_name()

```
std::string filevault::cli::HashCommand::get_botan_algorithm_name (
            const std::string & algo)  [private]
```

Here is the caller graph for this function:

9.49.3.6 is_secure_algorithm()

```
bool filevault::cli::HashCommand::is_secure_algorithm (
```

```
            const std::string & algo)  [private]
```

Here is the caller graph for this function:

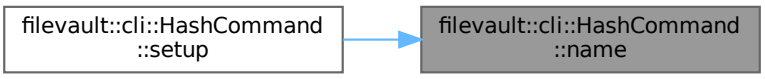
9.49.3.7 name()

```
std::string filevault::cli::HashCommand::name () const  [inline], [override], [virtual]
```

Get command name.

Implements filevault::cli::ICommand.

Here is the caller graph for this function:



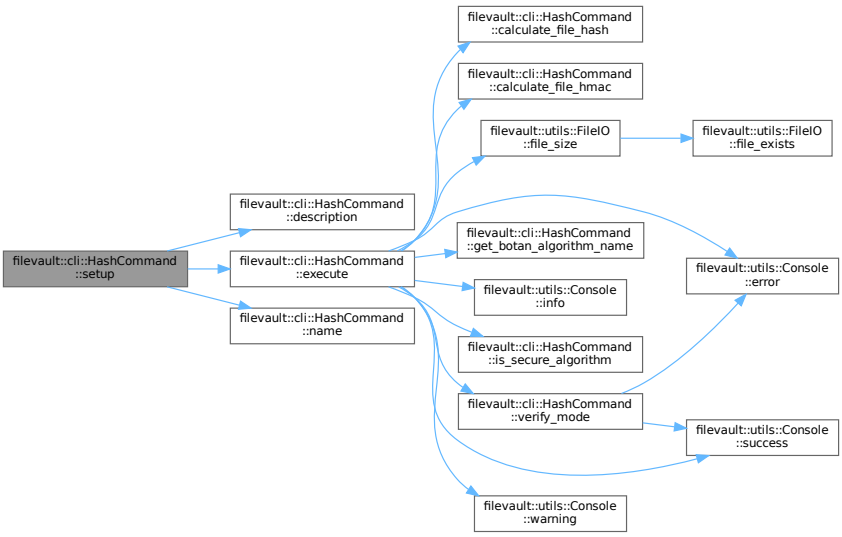
9.49.3.8 setup()

```
void filevault::cli::HashCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.

Implements filevault::cli::ICommand.

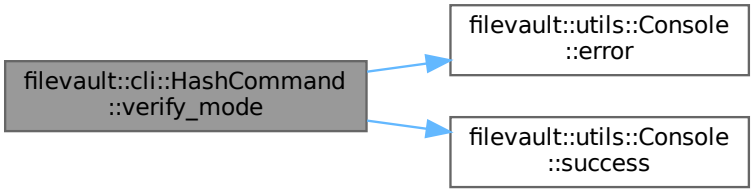
Here is the call graph for this function:



9.49.3.9 verify_mode()

```
int filevault::cli::HashCommand::verify_mode (
            const std::string & calculated_hash)  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.49.4 Member Data Documentation

9.49.4.1 algorithm_

```
std::string filevault::cli::HashCommand::algorithm_ = "sha256"  [private]
```

9.49.4.2 benchmark_

```
bool filevault::cli::HashCommand::benchmark_ = false  [private]
```

9.49.4.3 engine_

```
core::CryptoEngine& filevault::cli::HashCommand::engine_  [private]
```

9.49.4.4 hmac_key_

```
std::string filevault::cli::HashCommand::hmac_key_  [private]
```

9.49.4.5 input_file_

```
std::string filevault::cli::HashCommand::input_file_  [private]
```

9.49.4.6 no_filename_

```
bool filevault::cli::HashCommand::no_filename_ = false  [private]
```

9.49.4.7 output_file_

```
std::string filevault::cli::HashCommand::output_file_  [private]
```

9.49.4.8 output_format_

```
std::string filevault::cli::HashCommand::output_format_ = "hex"  [private]
```

9.49.4.9 uppercase_

```
bool filevault::cli::HashCommand::uppercase_ = false  [private]
```

9.49.4.10 verbose_

```
bool filevault::cli::HashCommand::verbose_ = false  [private]
```

9.49.4.11 verify_hash_

```
std::string filevault::cli::HashCommand::verify_hash_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/hash_cmd.hpp
- src/cli/commands/hash_cmd.cpp

9.50 filevault::core::HashConfig Struct Reference

Configuration for hashing operations.

`#include <types.hpp>`

Public Attributes

- HashType algorithm = HashType::SHA256
- bool hmac_mode = false
- std::vector< uint8_t > hmac_key
- bool verify_mode = false
- std::string expected_hash
- bool uppercase = false
- bool include_filename = true

9.50.1 Detailed Description

Configuration for hashing operations.

9.50.2 Member Data Documentation

9.50.2.1 algorithm

```
HashType filevault::core::HashConfig::algorithm = HashType::SHA256
```

9.50.2.2 expected_hash

```
std::string filevault::core::HashConfig::expected_hash
```

9.50.2.3 hmac_key

```
std::vector<uint8_t> filevault::core::HashConfig::hmac_key
```

9.50.2.4 hmac_mode

```
bool filevault::core::HashConfig::hmac_mode = false
```

9.50.2.5 include_filename

```
bool filevault::core::HashConfig::include_filename = true
```

9.50.2.6 uppercase

```
bool filevault::core::HashConfig::uppercase = false
```

9.50.2.7 verify_mode

```
bool filevault::core::HashConfig::verify_mode = false
```

The documentation for this struct was generated from the following file:

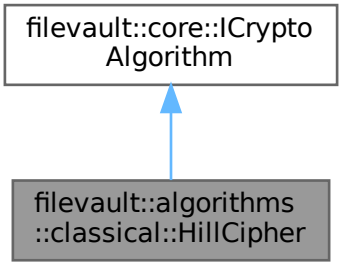
- include/filevault/core/types.hpp

9.51 filevault::algorithms::classical::HillCipher Class Reference

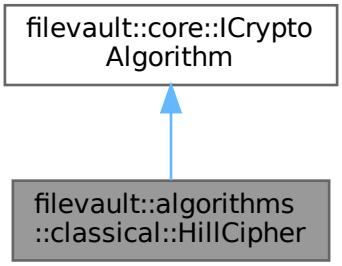
Hill cipher (2x2 matrix).

`#include <hill.hpp>`

Inheritance diagram for filevault::algorithms::classical::HillCipher:



Collaboration diagram for filevault::algorithms::classical::HillCipher:



Public Member Functions

- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.

- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Types

- using Matrix2x2 = std::array<int, 4>

Private Member Functions

- Matrix2x2 parse_key (std::span< const uint8_t > key)
- Matrix2x2 invert_matrix (const Matrix2x2 &matrix)
- int mod_inverse (int a, int m)
- int determinant (const Matrix2x2 &matrix)
- bool is_valid_key (const Matrix2x2 &matrix)
- std::string encrypt_block (const std::string &block, const Matrix2x2 &key_matrix)
- std::string decrypt_block (const std::string &block, const Matrix2x2 &inv_matrix)

9.51.1 Detailed Description

Hill cipher (2x2 matrix).
Educational cipher using matrix multiplication Key: 4 integers forming invertible 2x2 matrix mod 26

9.51.2 Member Typedef Documentation

9.51.2.1 Matrix2x2

```
using filevault::algorithms::classical::HillCipher::Matrix2x2 = std::array<int, 4>  [private]
```

9.51.3 Member Function Documentation

9.51.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::classical::HillCipher::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

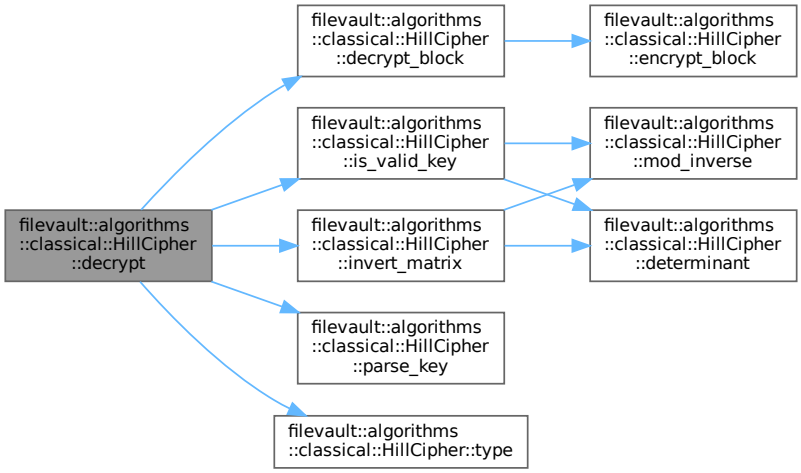
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

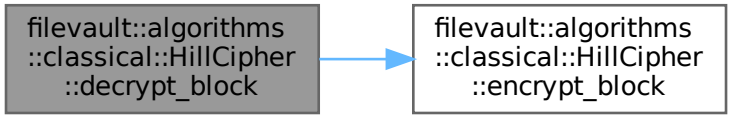
Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



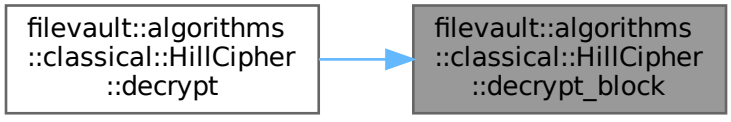
9.51.3.2 decrypt_block()

```
std::string filevault::algorithms::classical::HillCipher::decrypt_block (
            const std::string & block,
            const Matrix2x2 & inv_matrix)  [private]
```

Here is the call graph for this function:



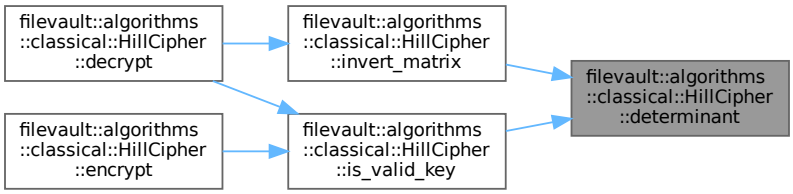
Here is the caller graph for this function:



9.51.3.3 determinant()

```
int filevault::algorithms::classical::HillCipher::determinant (
            const Matrix2x2 & matrix)  [private]
```

Here is the caller graph for this function:



9.51.3.4 encrypt()

```
core::CryptoResult filevault::algorithms::classical::HillCipher::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

Parameters

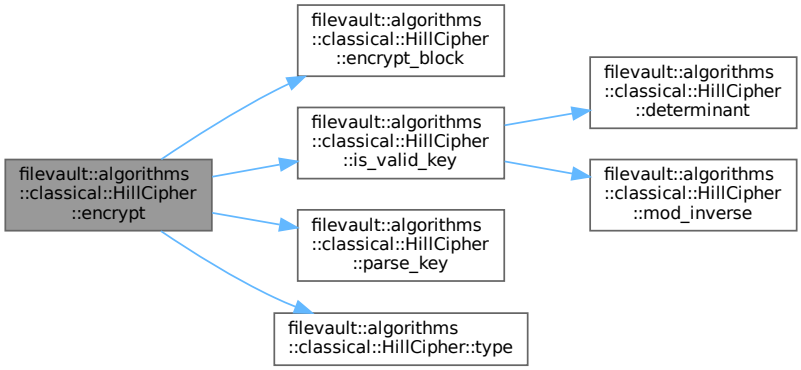
plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.

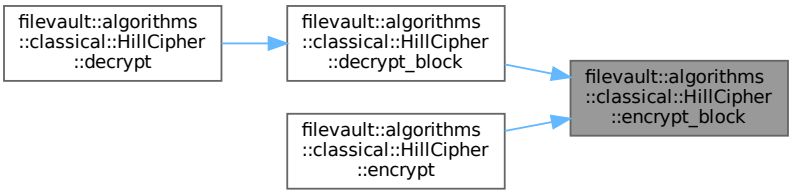
Here is the call graph for this function:



9.51.3.5 encrypt_block()

```
std::string filevault::algorithms::classical::HillCipher::encrypt_block (
            const std::string & block,
            const Matrix2x2 & key_matrix)  [private]
```

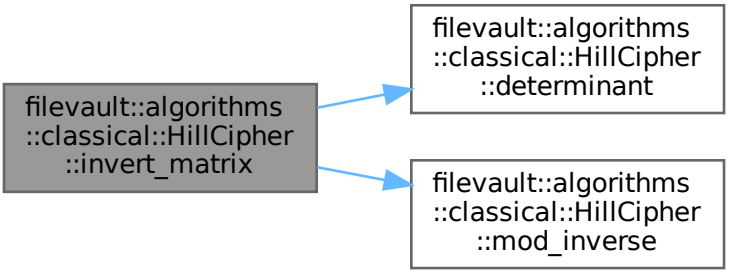
Here is the caller graph for this function:



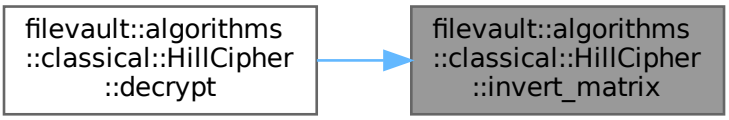
9.51.3.6 invert_matrix()

```
HillCipher::Matrix2x2 filevault::algorithms::classical::HillCipher::invert_matrix (
            const Matrix2x2 & matrix)  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.51.3.7 is_suitable_for()

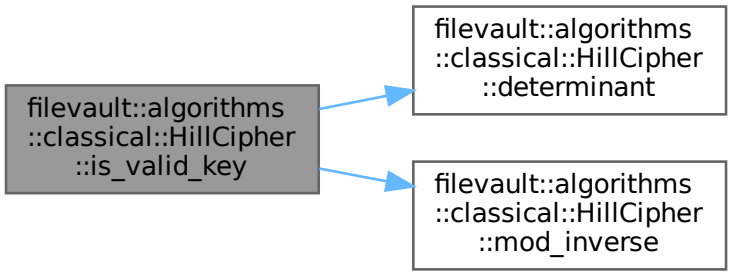
```
bool filevault::algorithms::classical::HillCipher::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

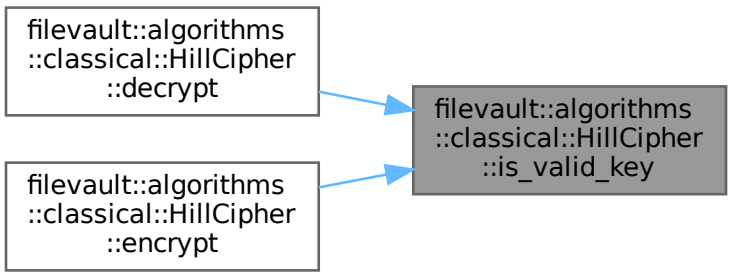
9.51.3.8 is_valid_key()

```
bool filevault::algorithms::classical::HillCipher::is_valid_key (
            const Matrix2x2 & matrix)  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.51.3.9 key_size()

```
size_t filevault::algorithms::classical::HillCipher::key_size () const  [inline], [override],
[virtual]
```

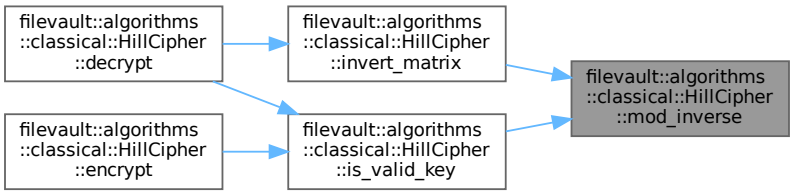
Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.51.3.10 mod_inverse()

```
int filevault::algorithms::classical::HillCipher::mod_inverse (
```

```
        int a,
        int m)  [private]
```

Here is the caller graph for this function:



9.51.3.11 name()

```
std::string filevault::algorithms::classical::HillCipher::name () const  [inline], [override],
[virtual]
```

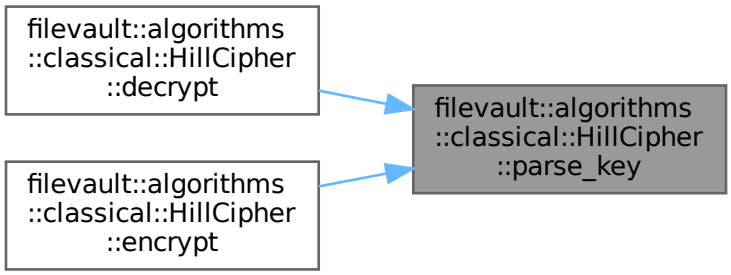
Get algorithm name.

Implements filevault::core::ICryptoAlgorithm.

9.51.3.12 parse_key()

```
HillCipher::Matrix2x2 filevault::algorithms::classical::HillCipher::parse_key (
        std::span< const uint8_t > key)  [private]
```

Here is the caller graph for this function:



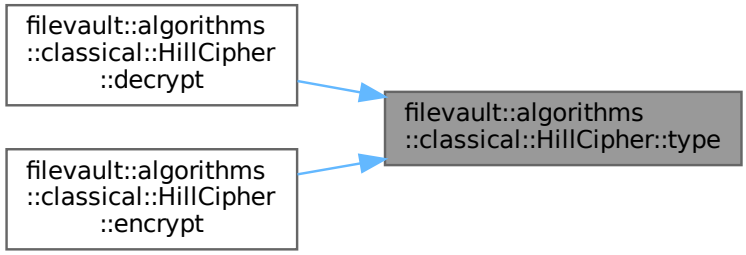
9.51.3.13 type()

```
core::AlgorithmType filevault::algorithms::classical::HillCipher::type () const  [inline],
[override], [virtual]
```

Get algorithm type.

Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

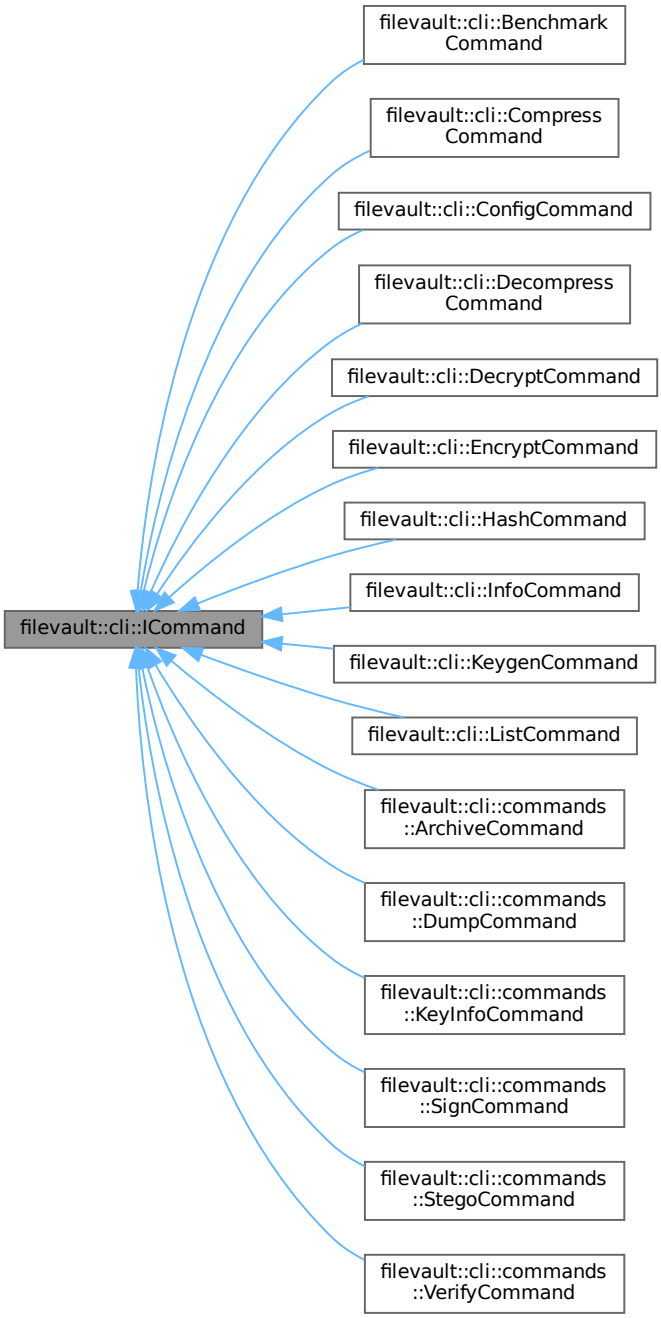
- include/filevault/algorithms/classical/hill.hpp
- src/algorithms/classical/hill.cpp

9.52 filevault::cli::ICommand Class Reference

Base interface for CLI commands.

`#include <command.hpp>`

Inheritance diagram for filevault::cli::ICommand:



Public Member Functions

- virtual ~ICommand ()=default
- virtual std::string name () const =0

  Get command name.
- virtual std::string description () const =0

  Get command description.
- virtual void setup (CLI::App &app)=0

Setup CLI11 subcommand.

- virtual int execute ()=0

  Execute the command.

9.52.1 Detailed Description

Base interface for CLI commands.

9.52.2 Constructor & Destructor Documentation

9.52.2.1 ~ICommand()

```
virtual filevault::cli::ICommand::~ICommand ()  [virtual], [default]
```

9.52.3 Member Function Documentation

9.52.3.1 description()

```
virtual std::string filevault::cli::ICommand::description () const  [pure virtual]
```

Get command description.

Implemented in filevault::cli::BenchmarkCommand, filevault::cli::commands::ArchiveCommand, filevault::cli::commands::DumpComma filevault::cli::commands::KeyInfoCommand, filevault::cli::commands::SignCommand, filevault::cli::commands::StegoCommand, filevault::cli::commands::VerifyCommand, filevault::cli::CompressCommand, filevault::cli::ConfigCommand, filevault::cli::DecompressCommand, filevault::cli::DecryptCommand, filevault::cli::EncryptCommand, filevault::cli::HashCommand, filevault::cli::InfoCommand, filevault::cli::KeygenCommand, and filevault::cli::ListCommand.

9.52.3.2 execute()

```
virtual int filevault::cli::ICommand::execute ()  [pure virtual]
```

Execute the command.

Implemented in filevault::cli::BenchmarkCommand, filevault::cli::commands::ArchiveCommand, filevault::cli::commands::DumpComma filevault::cli::commands::KeyInfoCommand, filevault::cli::commands::SignCommand, filevault::cli::commands::StegoCommand, filevault::cli::commands::VerifyCommand, filevault::cli::CompressCommand, filevault::cli::ConfigCommand, filevault::cli::DecompressCommand, filevault::cli::DecryptCommand, filevault::cli::EncryptCommand, filevault::cli::HashCommand, filevault::cli::InfoCommand, filevault::cli::KeygenCommand, and filevault::cli::ListCommand.

9.52.3.3 name()

```
virtual std::string filevault::cli::ICommand::name () const  [pure virtual]
```

Get command name.

Implemented in filevault::cli::BenchmarkCommand, filevault::cli::commands::ArchiveCommand, filevault::cli::commands::DumpComma filevault::cli::commands::KeyInfoCommand, filevault::cli::commands::SignCommand, filevault::cli::commands::StegoCommand, filevault::cli::commands::VerifyCommand, filevault::cli::CompressCommand, filevault::cli::ConfigCommand, filevault::cli::DecompressCommand, filevault::cli::DecryptCommand, filevault::cli::EncryptCommand, filevault::cli::HashCommand, filevault::cli::InfoCommand, filevault::cli::KeygenCommand, and filevault::cli::ListCommand.

9.52.3.4 setup()

```
virtual void filevault::cli::ICommand::setup (
            CLI::App & app)  [pure virtual]
```

Setup CLI11 subcommand.

Implemented in filevault::cli::BenchmarkCommand, filevault::cli::commands::ArchiveCommand, filevault::cli::commands::DumpComma filevault::cli::commands::KeyInfoCommand, filevault::cli::commands::SignCommand, filevault::cli::commands::StegoCommand, filevault::cli::commands::VerifyCommand, filevault::cli::CompressCommand, filevault::cli::ConfigCommand, filevault::cli::DecompressCommand, filevault::cli::DecryptCommand, filevault::cli::EncryptCommand, filevault::cli::HashCommand, filevault::cli::InfoCommand, filevault::cli::KeygenCommand, and filevault::cli::ListCommand.

The documentation for this class was generated from the following file:

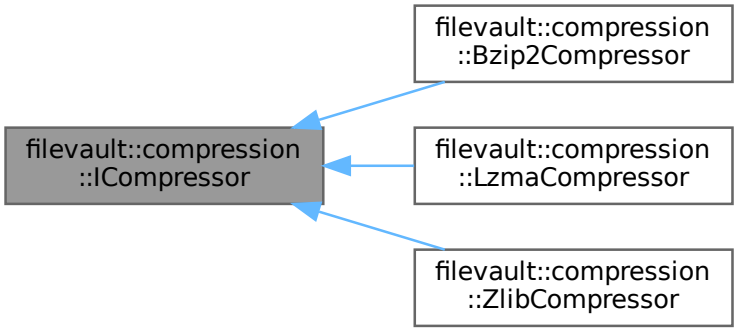
- include/filevault/cli/command.hpp

9.53 filevault::compression::ICompressor Class Reference

Interface for compression algorithms.

`#include <compressor.hpp>`

Inheritance diagram for filevault::compression::ICompressor:



Public Member Functions

- virtual ~ICompressor ()=default
- virtual std::string name () const =0

  Get compressor name.
- virtual CompressionResult compress (std::span< const uint8_t > input, int level=6)=0

  Compress data.
- virtual CompressionResult decompress (std::span< const uint8_t > input)=0

  Decompress data.

9.53.1 Detailed Description

Interface for compression algorithms.

9.53.2 Constructor & Destructor Documentation

9.53.2.1 ~ICompressor()

```
virtual filevault::compression::ICompressor::~ICompressor ()  [virtual], [default]
```

9.53.3 Member Function Documentation

9.53.3.1 compress()

```
virtual CompressionResult filevault::compression::ICompressor::compress (
            std::span< const uint8_t > input,
            int level = 6)  [pure virtual]
```

Compress data.

Parameters

input	Data to compress
level	Compression level (1-9, algorithm-specific)

Returns

Compressed data

Implemented in filevault::compression::Bzip2Compressor, filevault::compression::LzmaCompressor, and filevault::compression::ZlibCompressor.

9.53.3.2 decompress()

```
virtual CompressionResult filevault::compression::ICompressor::decompress (
            std::span< const uint8_t > input)  [pure virtual]
```

Decompress data.

Parameters

input	Compressed data

Returns

Decompressed data

Implemented in filevault::compression::Bzip2Compressor, filevault::compression::LzmaCompressor, and filevault::compression::ZlibCompressor.

9.53.3.3 name()

```
virtual std::string filevault::compression::ICompressor::name () const  [pure virtual]
```

Get compressor name.

Implemented in filevault::compression::Bzip2Compressor, filevault::compression::LzmaCompressor, and filevault::compression::ZlibCompressor.

The documentation for this class was generated from the following file:

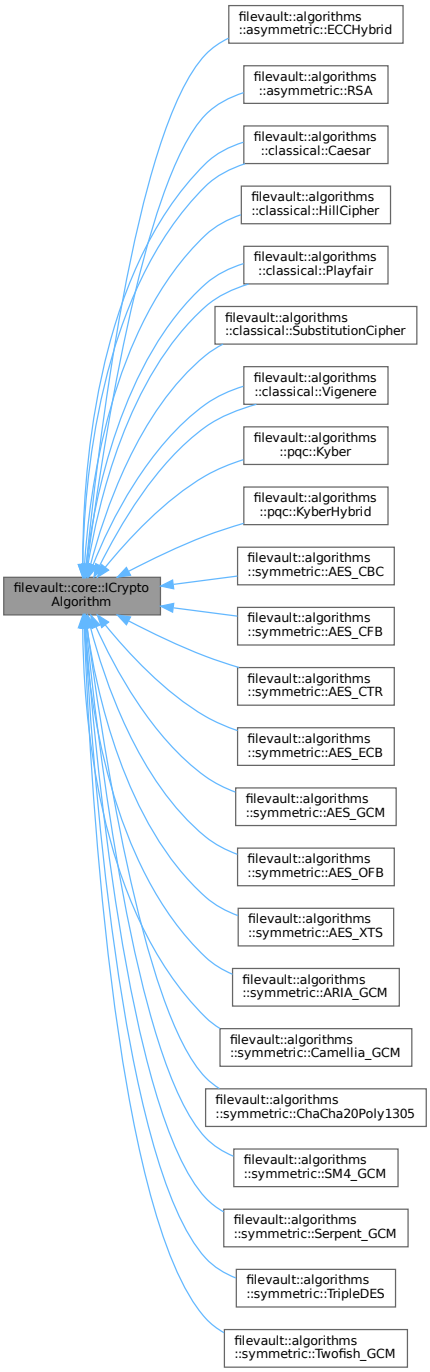
- include/filevault/compression/compressor.hpp

9.54 filevault::core::ICryptoAlgorithm Class Reference

Interface for cryptographic algorithms.

`#include <crypto_algorithm.hpp>`

Inheritance diagram for filevault::core::ICryptoAlgorithm:



Public Member Functions

- virtual ~ICryptoAlgorithm ()=default
- virtual std::string name () const =0

  Get algorithm name.
- virtual AlgorithmType type () const =0

  Get algorithm type.

- virtual CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const EncryptionConfig &config)=0

  Encrypt data.
- virtual CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const EncryptionConfig &config)=0

  Decrypt data.
- virtual size_t key_size () const =0

  Get recommended key size in bytes.
- virtual bool is_suitable_for (SecurityLevel level) const =0

  Check if algorithm is suitable for security level.

9.54.1 Detailed Description

Interface for cryptographic algorithms.

9.54.2 Constructor & Destructor Documentation

9.54.2.1 ~ICryptoAlgorithm()

```
virtual filevault::core::ICryptoAlgorithm::~ICryptoAlgorithm ()  [virtual], [default]
```

9.54.3 Member Function Documentation

9.54.3.1 decrypt()

```
virtual CryptoResult filevault::core::ICryptoAlgorithm::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const EncryptionConfig & config)  [pure virtual]
```

Decrypt data.

Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implemented in filevault::algorithms::asymmetric::ECCHybrid, filevault::algorithms::asymmetric::RSA, filevault::algorithms::classical::C filevault::algorithms::classical::Caesar, filevault::algorithms::classical::HillCipher, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::SubstitutionCipher, filevault::algorithms::classical::Vigenere, filevault::algorithms::classical::Vigenere, filevault::algorithms::pqc::Kyber, filevault::algorithms::pqc::KyberHybrid, filevault::algorithms::symmetric::AES_CBC, filevault::algorithms::symmetric::AES_CFB, filevault::algorithms::symmetric::AES_CTR, filevault::algorithms::symmetric::AES_ECB, filevault::algorithms::symmetric::AES_GCM, filevault::algorithms::symmetric::AES_OFB, filevault::algorithms::symmetric::AES_XTS, filevault::algorithms::symmetric::ARIA_GCM, filevault::algorithms::symmetric::Camellia_G filevault::algorithms::symmetric::ChaCha20Poly1305, filevault::algorithms::symmetric::Serpent_GCM, filevault::algorithms::symmetric filevault::algorithms::symmetric::TripleDES, and filevault::algorithms::symmetric::Twofish_GCM.

9.54.3.2 encrypt()

```
virtual CryptoResult filevault::core::ICryptoAlgorithm::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const EncryptionConfig & config)  [pure virtual]
```

Encrypt data.

Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implemented in filevault::algorithms::asymmetric::ECCHybrid, filevault::algorithms::asymmetric::RSA, filevault::algorithms::classical::C filevault::algorithms::classical::Caesar, filevault::algorithms::classical::HillCipher, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::SubstitutionCipher, filevault::algorithms::classical::Vigenere, filevault::algorithms::classical::Vigenere, filevault::algorithms::pqc::Kyber, filevault::algorithms::pqc::KyberHybrid, filevault::algorithms::symmetric::AES_CBC, filevault::algorithms::symmetric::AES_CFB, filevault::algorithms::symmetric::AES_CTR, filevault::algorithms::symmetric::AES_ECB, filevault::algorithms::symmetric::AES_GCM, filevault::algorithms::symmetric::AES_OFB, filevault::algorithms::symmetric::AES_XTS, filevault::algorithms::symmetric::ARIA_GCM, filevault::algorithms::symmetric::Camellia_G filevault::algorithms::symmetric::ChaCha20Poly1305, filevault::algorithms::symmetric::Serpent_GCM, filevault::algorithms::symmetric filevault::algorithms::symmetric::TripleDES, and filevault::algorithms::symmetric::Twofish_GCM.

9.54.3.3 is_suitable_for()

```
virtual bool filevault::core::ICryptoAlgorithm::is_suitable_for (
            SecurityLevel level) const  [pure virtual]
```

Check if algorithm is suitable for security level.

Implemented in filevault::algorithms::asymmetric::ECCHybrid, filevault::algorithms::asymmetric::RSA, filevault::algorithms::classical::C filevault::algorithms::classical::Caesar, filevault::algorithms::classical::HillCipher, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::SubstitutionCipher, filevault::algorithms::classical::Vigenere, filevault::algorithms::classical::Vigenere, filevault::algorithms::pqc::Kyber, filevault::algorithms::pqc::KyberHybrid, filevault::algorithms::symmetric::AES_CBC, filevault::algorithms::symmetric::AES_CFB, filevault::algorithms::symmetric::AES_CTR, filevault::algorithms::symmetric::AES_ECB, filevault::algorithms::symmetric::AES_GCM, filevault::algorithms::symmetric::AES_OFB, filevault::algorithms::symmetric::AES_XTS, filevault::algorithms::symmetric::ARIA_GCM, filevault::algorithms::symmetric::Camellia_G filevault::algorithms::symmetric::ChaCha20Poly1305, filevault::algorithms::symmetric::Serpent_GCM, filevault::algorithms::symmetric filevault::algorithms::symmetric::TripleDES, and filevault::algorithms::symmetric::Twofish_GCM.

9.54.3.4 key_size()

```
virtual size_t filevault::core::ICryptoAlgorithm::key_size () const  [pure virtual]
```

Get recommended key size in bytes.

Implemented in filevault::algorithms::asymmetric::ECCHybrid, filevault::algorithms::asymmetric::RSA, filevault::algorithms::classical::C filevault::algorithms::classical::Caesar, filevault::algorithms::classical::HillCipher, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::SubstitutionCipher, filevault::algorithms::classical::Vigenere, filevault::algorithms::classical::Vigenere, filevault::algorithms::pqc::Kyber, filevault::algorithms::pqc::KyberHybrid, filevault::algorithms::symmetric::AES_CBC, filevault::algorithms::symmetric::AES_CFB, filevault::algorithms::symmetric::AES_CTR, filevault::algorithms::symmetric::AES_ECB, filevault::algorithms::symmetric::AES_GCM, filevault::algorithms::symmetric::AES_OFB, filevault::algorithms::symmetric::AES_XTS, filevault::algorithms::symmetric::ARIA_GCM, filevault::algorithms::symmetric::Camellia_G filevault::algorithms::symmetric::ChaCha20Poly1305, filevault::algorithms::symmetric::Serpent_GCM, filevault::algorithms::symmetric filevault::algorithms::symmetric::TripleDES, and filevault::algorithms::symmetric::Twofish_GCM.

9.54.3.5 name()

```
virtual std::string filevault::core::ICryptoAlgorithm::name () const  [pure virtual]
```

Get algorithm name.

Implemented in filevault::algorithms::asymmetric::ECCHybrid, filevault::algorithms::asymmetric::RSA, filevault::algorithms::classical::C filevault::algorithms::classical::Caesar, filevault::algorithms::classical::HillCipher, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::SubstitutionCipher, filevault::algorithms::classical::Vigenere, filevault::algorithms::classical::Vigenere, filevault::algorithms::pqc::Kyber, filevault::algorithms::pqc::KyberHybrid, filevault::algorithms::symmetric::AES_CBC, filevault::algorithms::symmetric::AES_CFB, filevault::algorithms::symmetric::AES_CTR,

filevault::algorithms::symmetric::AES_ECB, filevault::algorithms::symmetric::AES_GCM, filevault::algorithms::symmetric::AES_OFB, filevault::algorithms::symmetric::AES_XTS, filevault::algorithms::symmetric::ARIA_GCM, filevault::algorithms::symmetric::Camellia_G filevault::algorithms::symmetric::ChaCha20Poly1305, filevault::algorithms::symmetric::Serpent_GCM, filevault::algorithms::symmetric filevault::algorithms::symmetric::TripleDES, and filevault::algorithms::symmetric::Twofish_GCM.

9.54.3.6 type()

```
virtual AlgorithmType filevault::core::ICryptoAlgorithm::type () const  [pure virtual]
```

Get algorithm type.

Implemented in filevault::algorithms::asymmetric::ECCHybrid, filevault::algorithms::asymmetric::RSA, filevault::algorithms::classical::C filevault::algorithms::classical::Caesar, filevault::algorithms::classical::HillCipher, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::Playfair, filevault::algorithms::classical::SubstitutionCipher, filevault::algorithms::classical::Vigenere, filevault::algorithms::classical::Vigenere, filevault::algorithms::pqc::Kyber, filevault::algorithms::pqc::KyberHybrid, filevault::algorithms::symmetric::AES_CBC, filevault::algorithms::symmetric::AES_CFB, filevault::algorithms::symmetric::AES_CTR, filevault::algorithms::symmetric::AES_ECB, filevault::algorithms::symmetric::AES_GCM, filevault::algorithms::symmetric::AES_OFB, filevault::algorithms::symmetric::AES_XTS, filevault::algorithms::symmetric::ARIA_GCM, filevault::algorithms::symmetric::Camellia_G filevault::algorithms::symmetric::ChaCha20Poly1305, filevault::algorithms::symmetric::Serpent_GCM, filevault::algorithms::symmetric filevault::algorithms::symmetric::TripleDES, and filevault::algorithms::symmetric::Twofish_GCM.

The documentation for this class was generated from the following file:

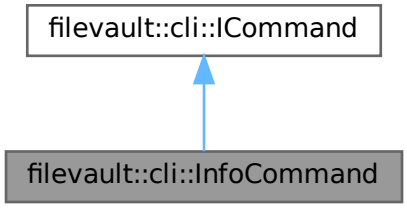
- include/filevault/core/crypto_algorithm.hpp

9.55 filevault::cli::InfoCommand Class Reference

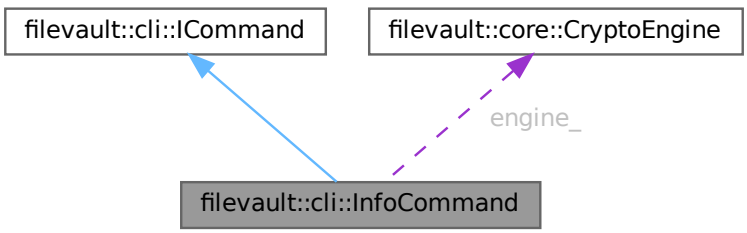
Info command - display encrypted file metadata.

`#include <info_cmd.hpp>`

Inheritance diagram for filevault::cli::InfoCommand:



Collaboration diagram for filevault::cli::InfoCommand:



Classes

- struct FileInfo

  Parse encrypted file header.

Public Member Functions

- InfoCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- FileInfo parse_file (const std::string &path)
- void display_info (const FileInfo &info)

Private Attributes

- core::CryptoEngine & engine_
- std::string input_file_
- bool verbose_ = false

9.55.1 Detailed Description

Info command - display encrypted file metadata.
Shows information about encrypted files without decrypting:

- File size (original, encrypted, compression ratio)
- Encryption algorithm
- KDF type and parameters
- Compression type
- Timestamp (if available)

9.55.2 Constructor & Destructor Documentation

9.55.2.1 InfoCommand()

```
filevault::cli::InfoCommand::InfoCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.55.3 Member Function Documentation

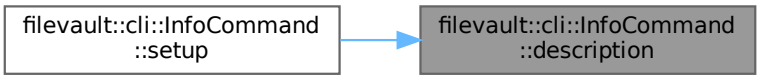
9.55.3.1 description()

```
std::string filevault::cli::InfoCommand::description () const  [inline], [override], [virtual]
```

Get command description.

Implements filevault::cli::ICommand.

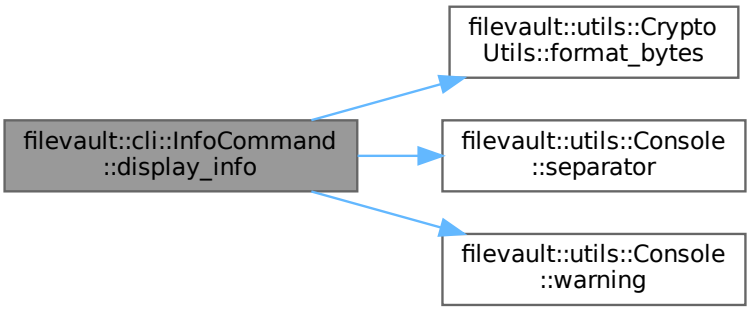
Here is the caller graph for this function:



9.55.3.2 display_info()

```
void filevault::cli::InfoCommand::display_info (
            const FileInfo & info)  [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:

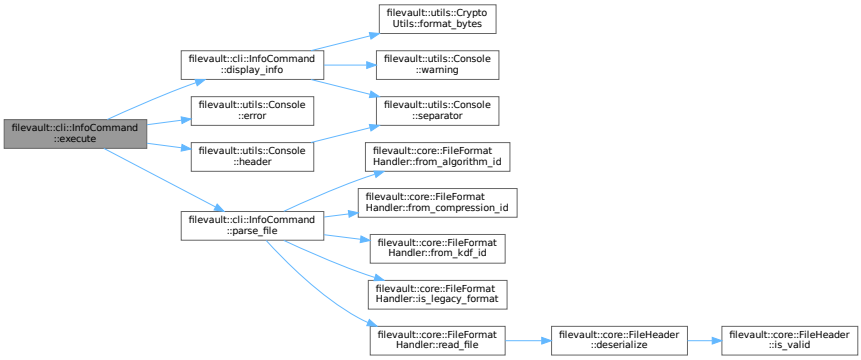
9.55.3.3 execute()

```
int filevault::cli::InfoCommand::execute ()  [override], [virtual]
```

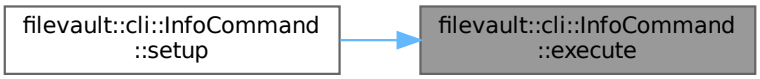
Execute the command.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



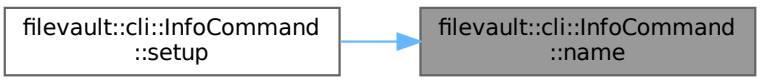
9.55.3.4 name()

```
std::string filevault::cli::InfoCommand::name () const  [inline], [override], [virtual]
```

Get command name.

Implements filevault::cli::ICommand.

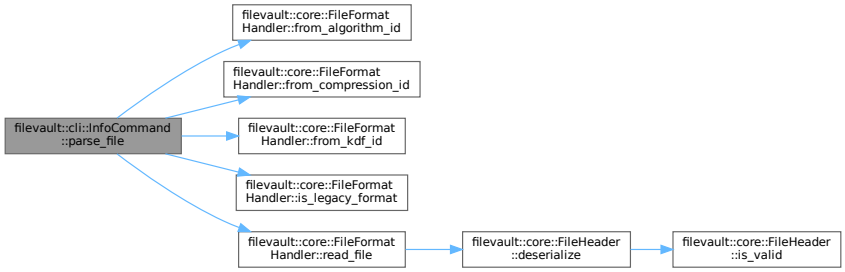
Here is the caller graph for this function:



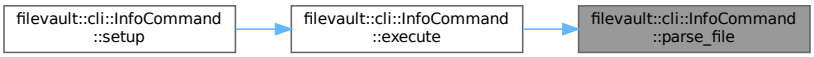
9.55.3.5 parse_file()

```
InfoCommand::FileInfo filevault::cli::InfoCommand::parse_file (
            const std::string & path)  [private]
```

Here is the call graph for this function:



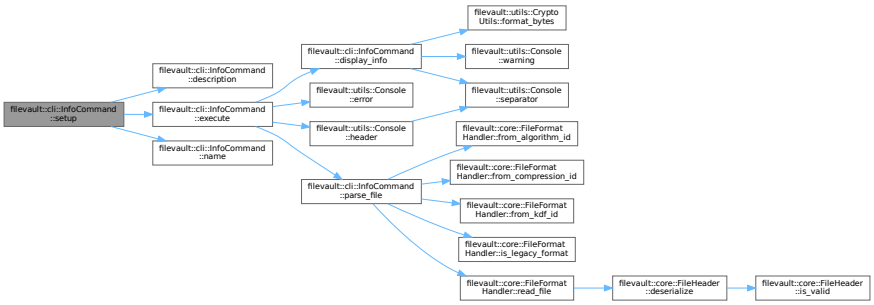
Here is the caller graph for this function:



9.55.3.6 setup()

```
void filevault::cli::InfoCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



9.55.4 Member Data Documentation

9.55.4.1 engine_

```
core::CryptoEngine& filevault::cli::InfoCommand::engine_  [private]
```

9.55.4.2 input_file_

```
std::string filevault::cli::InfoCommand::input_file_  [private]
```

9.55.4.3 verbose_

```
bool filevault::cli::InfoCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

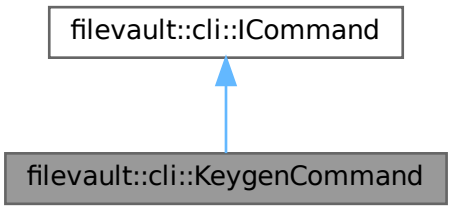
- include/filevault/cli/commands/info_cmd.hpp
- src/cli/commands/info_cmd.cpp

9.56 filevault::cli::KeygenCommand Class Reference

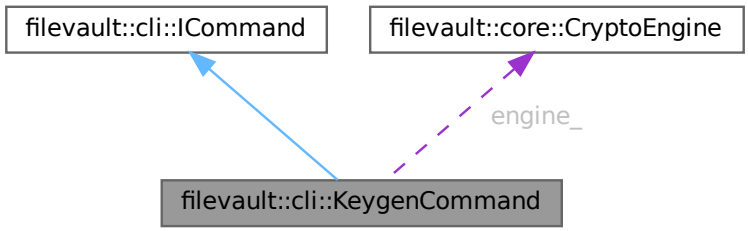
Command to generate key pairs for asymmetric encryption.

```
#include <keygen_cmd.hpp>
```

Inheritance diagram for filevault::cli::KeygenCommand:



Collaboration diagram for filevault::cli::KeygenCommand:



Public Member Functions

- KeygenCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- core::CryptoEngine & engine_
- std::string algorithm_ = "rsa-2048"
- std::string output_prefix_ = "filevault_key"
- bool force_ = false
- bool verbose_ = false

9.56.1 Detailed Description

Command to generate key pairs for asymmetric encryption.

9.56.2 Constructor & Destructor Documentation

9.56.2.1 KeygenCommand()

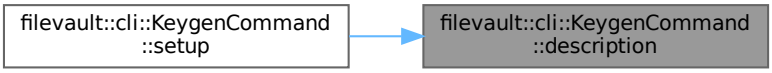
```
filevault::cli::KeygenCommand::KeygenCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.56.3 Member Function Documentation

9.56.3.1 description()

```
std::string filevault::cli::KeygenCommand::description () const  [inline], [override], [virtual]
```

Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:

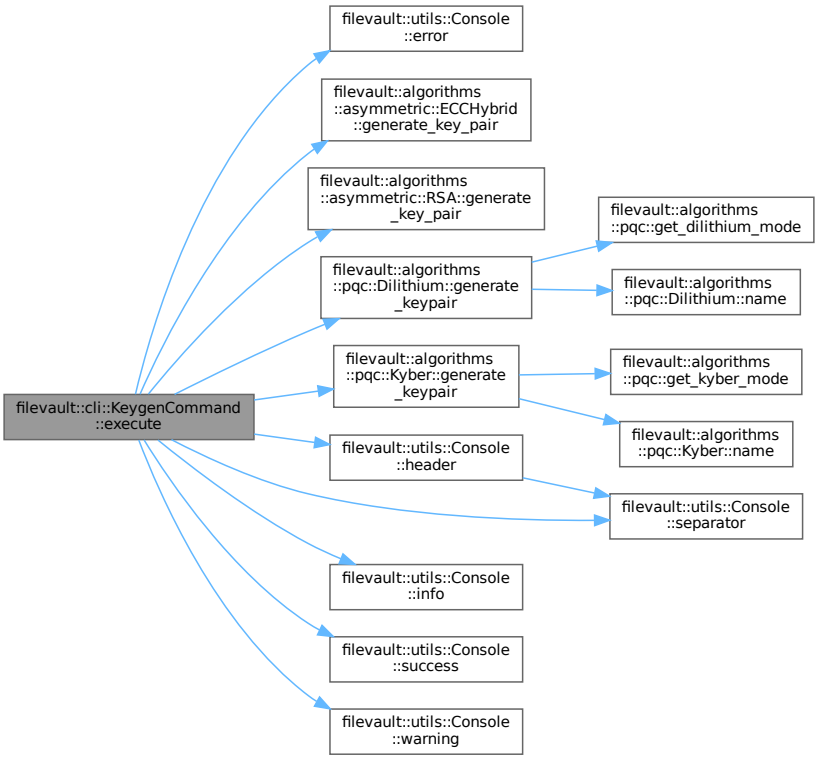


9.56.3.2 execute()

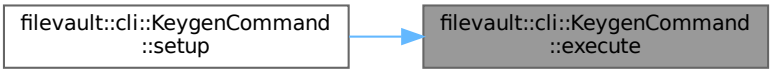
```
int filevault::cli::KeygenCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



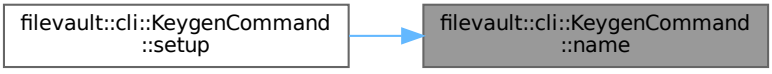
9.56.3.3 name()

```
std::string filevault::cli::KeygenCommand::name () const  [inline], [override], [virtual]
```

Get command name.

Implements filevault::cli::ICommand.

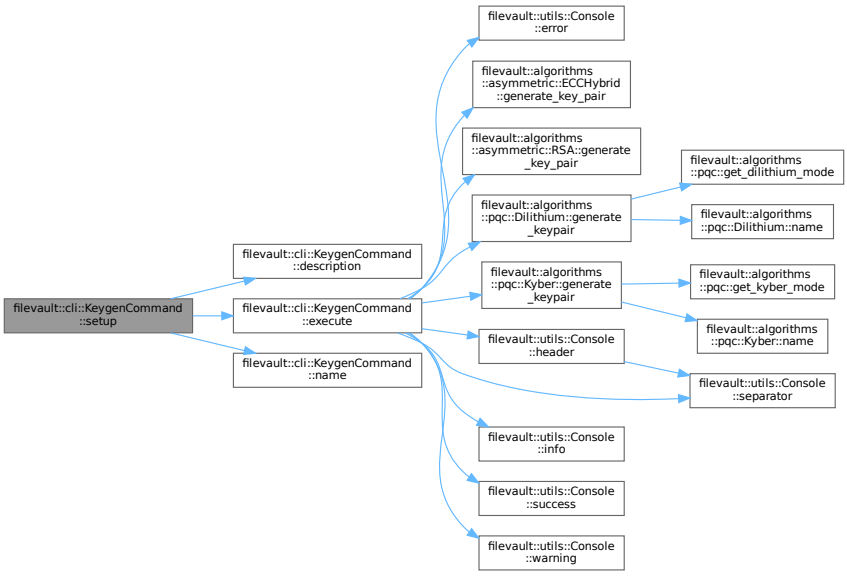
Here is the caller graph for this function:



9.56.3.4 setup()

```
void filevault::cli::KeygenCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



9.56.4 Member Data Documentation

9.56.4.1 algorithm_

```
std::string filevault::cli::KeygenCommand::algorithm_ = "rsa-2048"  [private]
```

9.56.4.2 engine_

```
core::CryptoEngine& filevault::cli::KeygenCommand::engine_  [private]
```

9.56.4.3 force_

```
bool filevault::cli::KeygenCommand::force_ = false  [private]
```

9.56.4.4 output_prefix_

```
std::string filevault::cli::KeygenCommand::output_prefix_ = "filevault_key"  [private]
```

9.56.4.5 verbose_

```
bool filevault::cli::KeygenCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

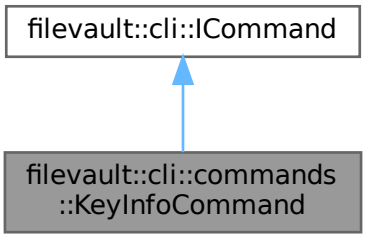
- include/filevault/cli/commands/keygen_cmd.hpp
- src/cli/commands/keygen_cmd.cpp

9.57 filevault::cli::commands::KeyInfoCommand Class Reference

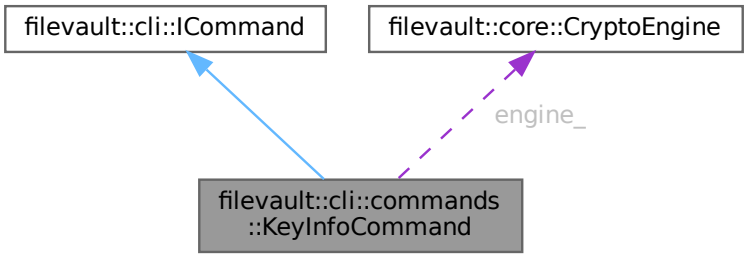
Command to display information about cryptographic keys.

`#include <keyinfo_cmd.hpp>`

Inheritance diagram for filevault::cli::commands::KeyInfoCommand:



Collaboration diagram for filevault::cli::commands::KeyInfoCommand:



Public Member Functions

- KeyInfoCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- std::string name_ = "keyinfo"
- std::string description_ = "Display information about cryptographic keys"
- core::CryptoEngine & engine_
- std::string key_path_
- bool show_public_ = false
- bool check_pair_ = false
- std::string pair_key_path_

9.57.1 Detailed Description

Command to display information about cryptographic keys.

9.57.2 Constructor & Destructor Documentation

9.57.2.1 KeyInfoCommand()

```
filevault::cli::commands::KeyInfoCommand::KeyInfoCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.57.3 Member Function Documentation

9.57.3.1 description()

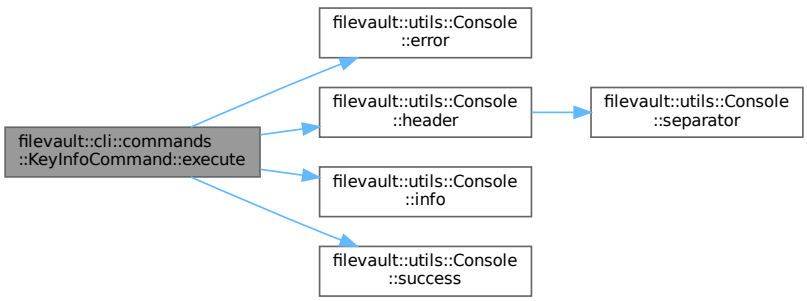
```
std::string filevault::cli::commands::KeyInfoCommand::description () const  [inline], [override],
[virtual]
```

Get command description.
Implements filevault::cli::ICommand.

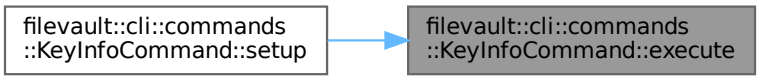
9.57.3.2 execute()

```
int filevault::cli::commands::KeyInfoCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



Here is the caller graph for this function:



9.57.3.3 name()

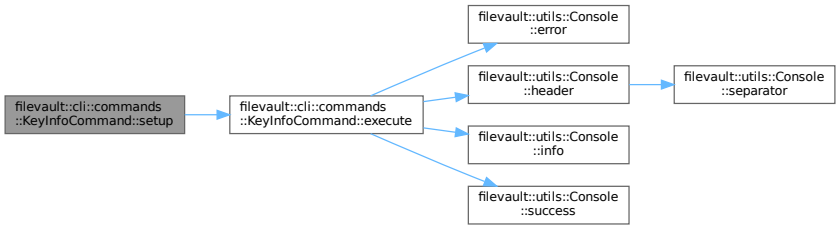
```
std::string filevault::cli::commands::KeyInfoCommand::name () const  [inline], [override],
[virtual]
```

Get command name.
Implements filevault::cli::ICommand.

9.57.3.4 setup()

```
void filevault::cli::commands::KeyInfoCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



9.57.4 Member Data Documentation

9.57.4.1 check_pair_

```
bool filevault::cli::commands::KeyInfoCommand::check_pair_ = false  [private]
```

9.57.4.2 description_

```
std::string filevault::cli::commands::KeyInfoCommand::description_ = "Display information
about cryptographic keys"  [private]
```

9.57.4.3 engine_

```
core::CryptoEngine& filevault::cli::commands::KeyInfoCommand::engine_  [private]
```

9.57.4.4 key_path_

```
std::string filevault::cli::commands::KeyInfoCommand::key_path_  [private]
```

9.57.4.5 name_

```
std::string filevault::cli::commands::KeyInfoCommand::name_ = "keyinfo"  [private]
```

9.57.4.6 pair_key_path_

```
std::string filevault::cli::commands::KeyInfoCommand::pair_key_path_  [private]
```

9.57.4.7 show_public_

```
bool filevault::cli::commands::KeyInfoCommand::show_public_ = false  [private]
```

The documentation for this class was generated from the following files:

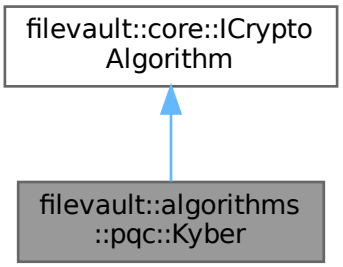
- include/filevault/cli/commands/keyinfo_cmd.hpp
- src/cli/commands/keyinfo_cmd.cpp

9.58 filevault::algorithms::pqc::Kyber Class Reference

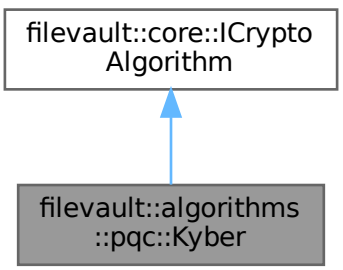
Kyber Key Encapsulation Mechanism (KEM).

`#include <post_quantum.hpp>`

Inheritance diagram for filevault::algorithms::pqc::Kyber:



Collaboration diagram for filevault::algorithms::pqc::Kyber:



Public Types

- enum class Variant { Kyber512 , Kyber768 , Kyber1024 }

Public Member Functions

- Kyber (Variant variant=Variant::Kyber768)
- ~Kyber () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encapsulate: Generate shared secret and ciphertext.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decapsulate: Recover shared secret from ciphertext.
- size_t key_size () const override

  Get recommended key size in bytes.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.
- PQKeyPair generate_keypair ()

  Generate Kyber key pair.
- size_t public_key_size () const

  Get public key size for this variant.
- size_t private_key_size () const

  Get private key size for this variant.
- size_t ciphertext_size () const

  Get ciphertext size for this variant.
- size_t shared_secret_size () const

  Get shared secret size (always 32 bytes).

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- Variant variant_
- std::string botan_name_
- core::AlgorithmType type_

9.58.1 Detailed Description

Kyber Key Encapsulation Mechanism (KEM).
NIST FIPS 203 (ML-KEM) - Post-quantum secure key encapsulation
Kyber is a lattice-based KEM that provides:

- IND-CCA2 security
- Resistance to quantum computer attacks
- Efficient key generation, encapsulation, and decapsulation

Variants:

- Kyber-512: ~128-bit post-quantum security
- Kyber-768: ~192-bit post-quantum security (recommended)
- Kyber-1024: ~256-bit post-quantum security

9.58.2 Member Enumeration Documentation

9.58.2.1 Variant

```
enum class filevault::algorithms::pqc::Kyber::Variant  [strong]
```

Enumerator

Kyber512	
Kyber768	
Kyber1024	

9.58.3 Constructor & Destructor Documentation

9.58.3.1 Kyber()

```
filevault::algorithms::pqc::Kyber::Kyber (
            Variant variant = Variant::Kyber768)  [explicit]
```

9.58.3.2 ~Kyber()

```
filevault::algorithms::pqc::Kyber::~Kyber ()  [override], [default]
```

9.58.4 Member Function Documentation

9.58.4.1 ciphertext_size()

```
size_t filevault::algorithms::pqc::Kyber::ciphertext_size () const
```

Get ciphertext size for this variant.

9.58.4.2 decrypt()

```
core::CryptoResult filevault::algorithms::pqc::Kyber::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decapsulate: Recover shared secret from ciphertext.
Uses own private key to recover the shared secret.

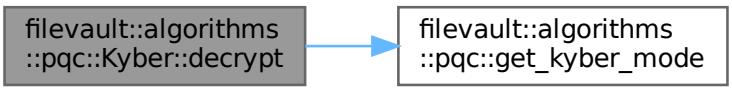
Parameters

ciphertext	Encapsulated ciphertext
key	Own private key
config	Configuration

Returns

CryptoResult with shared secret in 'data'

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:

9.58.4.3 encrypt()

```
core::CryptoResult filevault::algorithms::pqc::Kyber::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encapsulate: Generate shared secret and ciphertext.
Uses recipient's public key to generate a shared secret and encapsulated ciphertext.

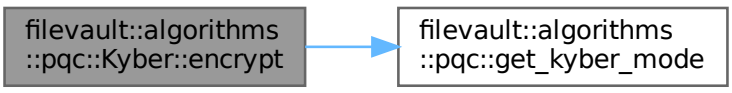
Parameters

plaintext	Not used (KEM doesn't encrypt directly)
key	Recipient's public key
config	Configuration

Returns

CryptoResult with encapsulated ciphertext and shared secret in 'data'

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:

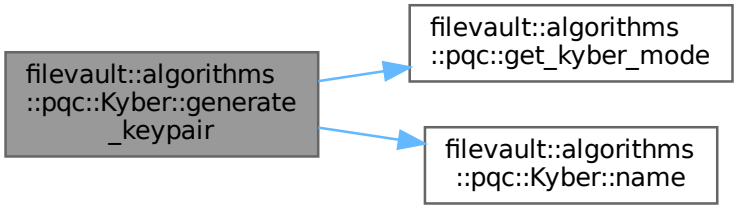


Here is the caller graph for this function:

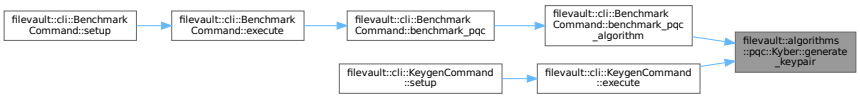
9.58.4.4 generate_keypair()

```
PQKeyPair filevault::algorithms::pqc::Kyber::generate_keypair ()
```

Generate Kyber key pair.
Here is the call graph for this function:



Here is the caller graph for this function:



9.58.4.5 is_suitable_for()

```
bool filevault::algorithms::pqc::Kyber::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.58.4.6 key_size()

```
size_t filevault::algorithms::pqc::Kyber::key_size () const  [override], [virtual]
```

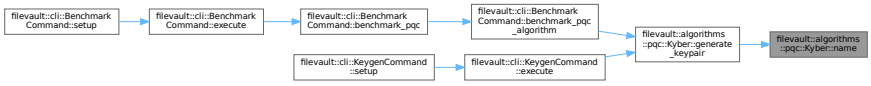
Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.58.4.7 name()

```
std::string filevault::algorithms::pqc::Kyber::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



9.58.4.8 private_key_size()

```
size_t filevault::algorithms::pqc::Kyber::private_key_size () const
```

Get private key size for this variant.

9.58.4.9 public_key_size()

```
size_t filevault::algorithms::pqc::Kyber::public_key_size () const
```

Get public key size for this variant.

9.58.4.10 shared_secret_size()

```
size_t filevault::algorithms::pqc::Kyber::shared_secret_size () const  [inline]
```

Get shared secret size (always 32 bytes).

9.58.4.11 type()

```
core::AlgorithmType filevault::algorithms::pqc::Kyber::type () const  [override], [virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.58.5 Member Data Documentation

9.58.5.1 botan_name_

```
std::string filevault::algorithms::pqc::Kyber::botan_name_  [private]
```

9.58.5.2 type_

```
core::AlgorithmType filevault::algorithms::pqc::Kyber::type_  [private]
```

9.58.5.3 variant_

```
Variant filevault::algorithms::pqc::Kyber::variant_  [private]
```

The documentation for this class was generated from the following files:

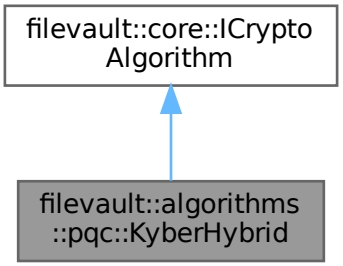
- include/filevault/algorithms/pqc/post_quantum.hpp
- src/algorithms/pqc/post_quantum.cpp

9.59 filevault::algorithms::pqc::KyberHybrid Class Reference

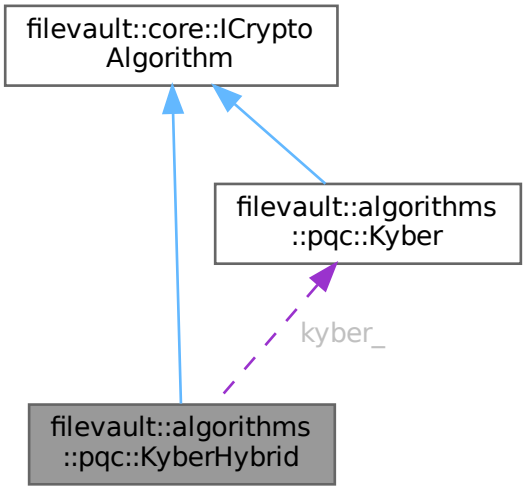
Hybrid encryption combining Kyber with AES-GCM.

```
#include <post_quantum.hpp>
```

Inheritance diagram for filevault::algorithms::pqc::KyberHybrid:



Collaboration diagram for filevault::algorithms::pqc::KyberHybrid:



Public Member Functions

- KyberHybrid (Kyber::Variant variant=Kyber::Variant::Kyber768)
- ~KyberHybrid () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data using Kyber+AES-GCM hybrid.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

Decrypt data using Kyber+AES-GCM hybrid.

- size_t key_size () const override

  Get recommended key size in bytes.

- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

- PQKeyPair generate_keypair ()

  Generate hybrid key pair.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- Kyber kyber_

9.59.1 Detailed Description

Hybrid encryption combining Kyber with AES-GCM.
This provides defense-in-depth by combining:

- Kyber for post-quantum secure key encapsulation
- AES-256-GCM for symmetric encryption

Even if one algorithm is broken, the other provides protection.

9.59.2 Constructor & Destructor Documentation

9.59.2.1 KyberHybrid()

```
filevault::algorithms::pqc::KyberHybrid::KyberHybrid (
            Kyber::Variant variant = Kyber::Variant::Kyber768)  [explicit]
```

9.59.2.2 ~KyberHybrid()

```
filevault::algorithms::pqc::KyberHybrid::~KyberHybrid ()  [override], [default]
```

9.59.3 Member Function Documentation

9.59.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::pqc::KyberHybrid::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data using Kyber+AES-GCM hybrid.

1. Decapsulate Kyber ciphertext to get shared secret
2. Use shared secret as AES key
3. Decrypt data with AES-256-GCM

Parameters

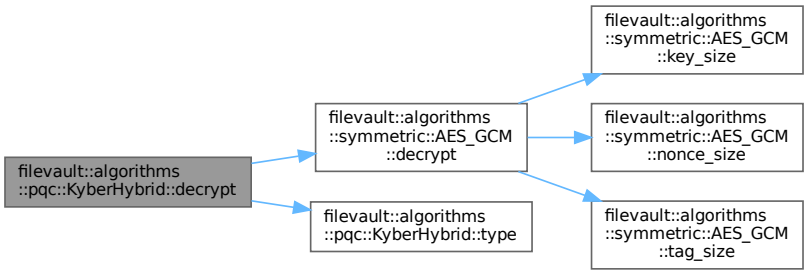
ciphertext	Encrypted data
key	Own Kyber private key

config	Configuration

Returns

Decrypted plaintext

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:

9.59.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::pqc::KyberHybrid::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data using Kyber+AES-GCM hybrid.

1. Encapsulate with Kyber to get shared secret
2. Use shared secret as AES key
3. Encrypt data with AES-256-GCM

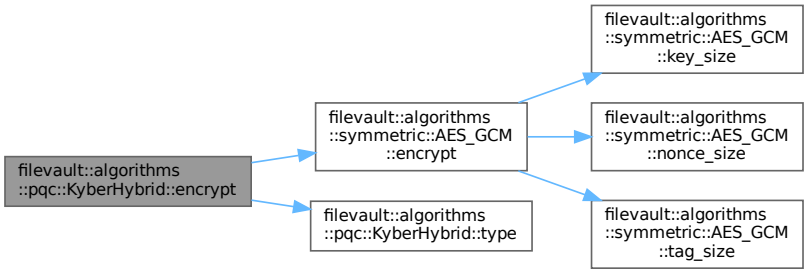
Parameters

plaintext	Data to encrypt
key	Recipient's Kyber public key
config	Configuration

Returns

Ciphertext containing Kyber ciphertext + AES ciphertext

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:

9.59.3.3 generate_keypair()

```
PQKeyPair filevault::algorithms::pqc::KyberHybrid::generate_keypair ()
```

Generate hybrid key pair.
Here is the caller graph for this function:

9.59.3.4 is_suitable_for()

```
bool filevault::algorithms::pqc::KyberHybrid::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.59.3.5 key_size()

```
size_t filevault::algorithms::pqc::KyberHybrid::key_size () const  [override], [virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

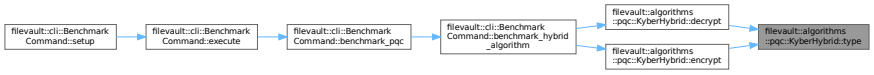
9.59.3.6 name()

```
std::string filevault::algorithms::pqc::KyberHybrid::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.59.3.7 type()

`core::AlgorithmType filevault::algorithms::pqc::KyberHybrid::type () const [override], [virtual]`
Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.59.4 Member Data Documentation

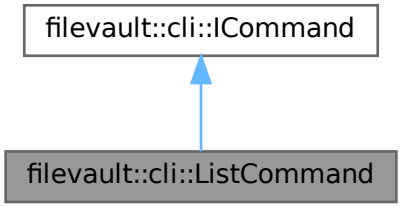
9.59.4.1 kyber_

`Kyber filevault::algorithms::pqc::KyberHybrid::kyber_ [private]`
The documentation for this class was generated from the following files:

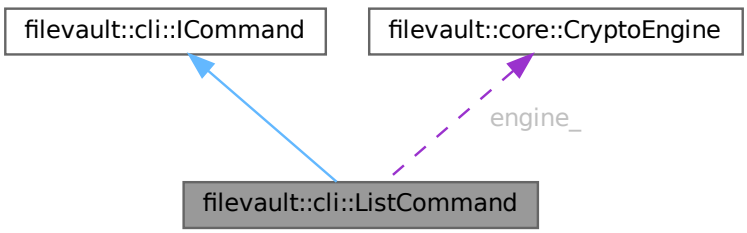
- include/filevault/algorithms/pqc/post_quantum.hpp
- src/algorithms/pqc/post_quantum.cpp

9.60 filevault::cli::ListCommand Class Reference

`#include <list_cmd.hpp>`
Inheritance diagram for filevault::cli::ListCommand:



Collaboration diagram for filevault::cli::ListCommand:



Public Member Functions

- ListCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- core::CryptoEngine & engine_

9.60.1 Constructor & Destructor Documentation

9.60.1.1 ListCommand()

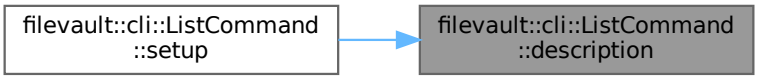
```
filevault::cli::ListCommand::ListCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.60.2 Member Function Documentation

9.60.2.1 description()

```
std::string filevault::cli::ListCommand::description () const  [inline], [override], [virtual]
```

Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:

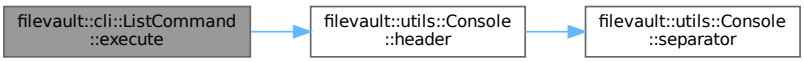


9.60.2.2 execute()

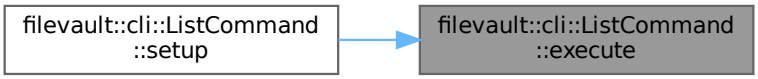
```
int filevault::cli::ListCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



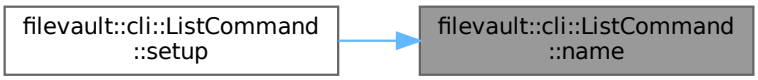
9.60.2.3 name()

```
std::string filevault::cli::ListCommand::name () const  [inline], [override], [virtual]
```

Get command name.

Implements filevault::cli::ICommand.

Here is the caller graph for this function:



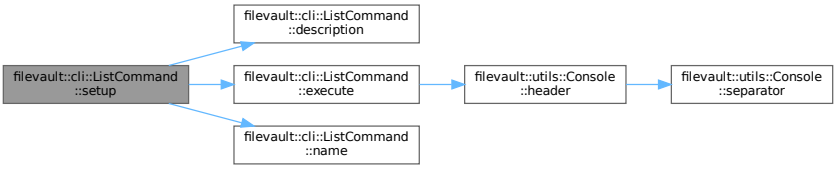
9.60.2.4 setup()

```
void filevault::cli::ListCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



9.60.3 Member Data Documentation

9.60.3.1 engine_

```
core::CryptoEngine& filevault::cli::ListCommand::engine_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/list_cmd.hpp
- src/cli/commands/list_cmd.cpp

9.61 filevault::steganography::LSBSteganography Class Reference

LSB (Least Significant Bit) Steganography.

```
#include <lsb.hpp>
```

Static Public Member Functions

- static bool embed (const std::string &cover_image_path, std::span< const uint8_t > secret_data, const std::string &output_path, int bits_per_channel=1)

  Embed secret data into image.
- static std::vector< uint8_t > extract (const std::string &stego_image_path, int bits_per_channel=1)

  Extract hidden data from stego image.
- static size_t calculate_capacity (const std::string &image_path, int bits_per_channel=1)

  Calculate maximum capacity for given image.

Static Private Member Functions

- static void embed_byte (uint8_t *pixel_data, size_t pixel_count, size_t &bit_index, uint8_t byte, int bits_per_channel)
- static uint8_t extract_byte (const uint8_t *pixel_data, size_t pixel_count, size_t &bit_index, int bits_per_channel)

Static Private Attributes

- static constexpr size_t LENGTH_HEADER_SIZE = 4

9.61.1 Detailed Description

LSB (Least Significant Bit) Steganography.
Embeds secret data into image pixels by modifying the least significant bits. Supports PNG and BMP images.
Capacity: For RGB images, ~3 bits per pixel (if using 1 bit per channel) For an 800x600 image: (800 * 600 * 3) / 8 = 180,000 bytes max

9.61.2 Member Function Documentation

9.61.2.1 calculate_capacity()

```
size_t filevault::steganography::LSBSteganography::calculate_capacity (
            const std::string & image_path,
            int bits_per_channel = 1)  [static]
```

Calculate maximum capacity for given image.

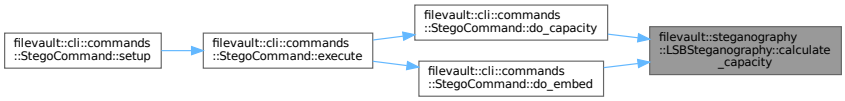
Parameters

image_path	Path to image
bits_per_channel	Number of LSBs per channel

Returns

Maximum bytes that can be hidden, or 0 on error

Here is the caller graph for this function:



9.61.2.2 embed()

```
bool filevault::steganography::LSBSteganography::embed (
            const std::string & cover_image_path,
            std::span< const uint8_t > secret_data,
            const std::string & output_path,
            int bits_per_channel = 1)  [static]
```

Embed secret data into image.

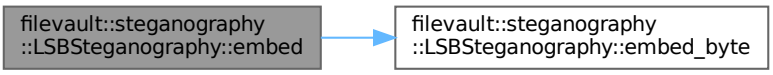
Parameters

cover_image_path	Path to cover image (PNG/BMP)
secret_data	Data to hide
output_path	Path for output stego image
bits_per_channel	Number of LSBs to use per color channel (1-4, default 1)

Returns

true if successful, false otherwise

Here is the call graph for this function:



Here is the caller graph for this function:

9.61.2.3 embed_byte()

```
void filevault::steganography::LSBSteganography::embed_byte (
            uint8_t * pixel_data,
```

```
            size_t pixel_count,
            size_t & bit_index,
            uint8_t byte,
            int bits_per_channel)  [static], [private]
```

Here is the caller graph for this function:

9.61.2.4 extract()

```
std::vector< uint8_t > filevault::steganography::LSBSteganography::extract (
            const std::string & stego_image_path,
            int bits_per_channel = 1)  [static]
```

Extract hidden data from stego image.

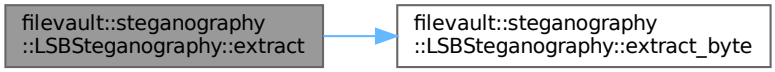
Parameters

stego_image_path	Path to stego image
bits_per_channel	Number of LSBs used per color channel (must match embed)

Returns

Extracted secret data, or empty vector on failure

Here is the call graph for this function:



Here is the caller graph for this function:

9.61.2.5 extract_byte()

```
uint8_t filevault::steganography::LSBSteganography::extract_byte (
            const uint8_t * pixel_data,
            size_t pixel_count,
            size_t & bit_index,
            int bits_per_channel)  [static], [private]
```

Here is the caller graph for this function:

9.61.3 Member Data Documentation

9.61.3.1 LENGTH_HEADER_SIZE

```
size_t filevault::steganography::LSBSteganography::LENGTH_HEADER_SIZE = 4  [static], [constexpr], [private]
```

The documentation for this class was generated from the following files:

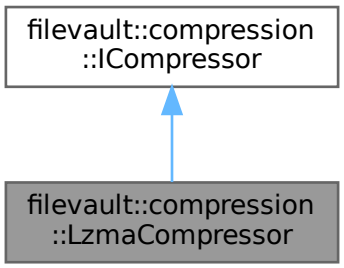
- include/filevault/steganography/lsb.hpp
- src/steganography/lsb.cpp

9.62 filevault::compression::LzmaCompressor Class Reference

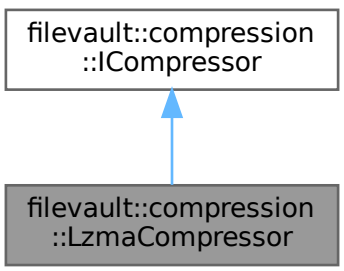
LZMA compressor (maximum compression, slowest).

```
#include <compressor.hpp>
```

Inheritance diagram for filevault::compression::LzmaCompressor:



Collaboration diagram for filevault::compression::LzmaCompressor:



Public Member Functions

- std::string name () const override

  Get compressor name.
- CompressionResult compress (std::span< const uint8_t > input, int level=6) override

  Compress data.
- CompressionResult decompress (std::span< const uint8_t > input) override

  Decompress data.

Public Member Functions inherited from filevault::compression::ICompressor

- virtual ~ICompressor ()=default

9.62.1 Detailed Description

LZMA compressor (maximum compression, slowest).

9.62.2 Member Function Documentation

9.62.2.1 compress()

```
CompressionResult filevault::compression::LzmaCompressor::compress (
            std::span< const uint8_t > input,
            int level = 6)  [override], [virtual]
```

Compress data.

Parameters

input	Data to compress
level	Compression level (1-9, algorithm-specific)

Returns

Compressed data

Implements filevault::compression::ICompressor.

9.62.2.2 decompress()

```
CompressionResult filevault::compression::LzmaCompressor::decompress (
            std::span< const uint8_t > input)  [override], [virtual]
```

Decompress data.

Parameters

input	Compressed data

Returns

Decompressed data

Implements filevault::compression::ICompressor.

9.62.2.3 name()

```
std::string filevault::compression::LzmaCompressor::name () const  [inline], [override], [virtual]
```

Get compressor name.
Implements filevault::compression::ICompressor.
The documentation for this class was generated from the following files:

- include/filevault/compression/compressor.hpp
- src/compression/compressor.cpp

9.63 filevault::core::ModePreset Struct Reference

Mode presets for different user levels.
`#include <modes.hpp>`

Public Member Functions

- std::string name () const
- std::string description () const
- void apply_to (EncryptionConfig &config) const

  Apply preset to config.

Static Public Member Functions

- static ModePreset get_preset (UserMode mode)

  Get preset by mode.
- static UserMode parse_mode (const std::string &name)

  Parse mode from string.
- static std::vector< ModePreset > get_all_presets ()

  Get all available presets.

Public Attributes

- UserMode mode
- AlgorithmType algorithm
- KDFType kdf
- SecurityLevel security_level
- CompressionType compression
- int compression_level
- uint32_t kdf_iterations
- uint32_t kdf_memory_kb
- uint32_t kdf_parallelism

9.63.1 Detailed Description

Mode presets for different user levels.
BASIC: Fast, good security (student/casual use) STANDARD: Balanced security/performance (professional use) ADVANCED: Maximum security (paranoid/high-value data)

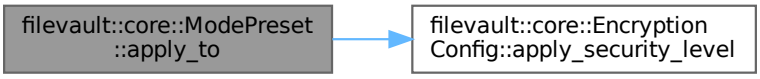
9.63.2 Member Function Documentation

9.63.2.1 apply_to()

```
void filevault::core::ModePreset::apply_to (
            EncryptionConfig & config) const
```

Apply preset to config.

Here is the call graph for this function:



9.63.2.2 description()

```
std::string filevault::core::ModePreset::description () const
```

9.63.2.3 get_all_presets()

```
std::vector< ModePreset > filevault::core::ModePreset::get_all_presets ()  [static]
```

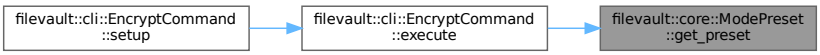
Get all available presets.

9.63.2.4 get_preset()

```
ModePreset filevault::core::ModePreset::get_preset (
            UserMode mode)  [static]
```

Get preset by mode.

Here is the caller graph for this function:



9.63.2.5 name()

```
std::string filevault::core::ModePreset::name () const
```

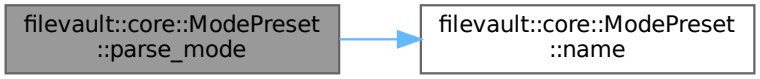
Here is the caller graph for this function:

9.63.2.6 parse_mode()

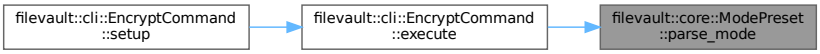
```
UserMode filevault::core::ModePreset::parse_mode (
            const std::string & name)  [static]
```

Parse mode from string.

Here is the call graph for this function:



Here is the caller graph for this function:



9.63.3 Member Data Documentation

9.63.3.1 algorithm

```
AlgorithmType filevault::core::ModePreset::algorithm
```

9.63.3.2 compression

```
CompressionType filevault::core::ModePreset::compression
```

9.63.3.3 compression_level

```
int filevault::core::ModePreset::compression_level
```

9.63.3.4 kdf

```
KDFType filevault::core::ModePreset::kdf
```

9.63.3.5 kdf_iterations

```
uint32_t filevault::core::ModePreset::kdf_iterations
```

9.63.3.6 kdf_memory_kb

```
uint32_t filevault::core::ModePreset::kdf_memory_kb
```

9.63.3.7 kdf_parallelism

```
uint32_t filevault::core::ModePreset::kdf_parallelism
```

9.63.3.8 mode

```
UserMode filevault::core::ModePreset::mode
```

9.63.3.9 security_level

```
SecurityLevel filevault::core::ModePreset::security_level
```

The documentation for this struct was generated from the following files:

- include/filevault/core/modes.hpp
- src/core/modes.cpp

9.64 filevault::utils::Password Class Reference

Password utilities for secure input and strength analysis.

```
#include <password.hpp>
```

Static Public Member Functions

- static std::string read_secure (const std::string &prompt="Password: ", bool confirm=false)

  Read password from terminal with masking.
- static core::PasswordAnalysis analyze_strength (const std::string &password)

  Analyze password strength.
- static std::string get_strength_color (core::PasswordStrength strength)

  Get strength color for display.
- static std::string get_strength_label (core::PasswordStrength strength)

  Get strength label.
- static void display_strength_meter (const core::PasswordAnalysis &analysis)

  Display password strength meter.

Static Private Member Functions

- static void disable_echo ()

  Platform-specific: Disable terminal echo.
- static void enable_echo ()

  Platform-specific: Enable terminal echo.
- static bool is_common_password (const std::string &password)

  Check if password is in common password list.
- static double calculate_entropy (const std::string &password)

  Calculate password entropy.
- static std::pair< std::string, std::string > estimate_crack_time (double entropy, const core::PasswordAnalysis &analysis)

  Estimate crack time.

Static Private Attributes

- static const std::vector< std::string > common_passwords_

9.64.1 Detailed Description

Password utilities for secure input and strength analysis.

9.64.2 Member Function Documentation

9.64.2.1 analyze_strength()

```
core::PasswordAnalysis filevault::utils::Password::analyze_strength (
            const std::string & password)  [static]
```

Analyze password strength.

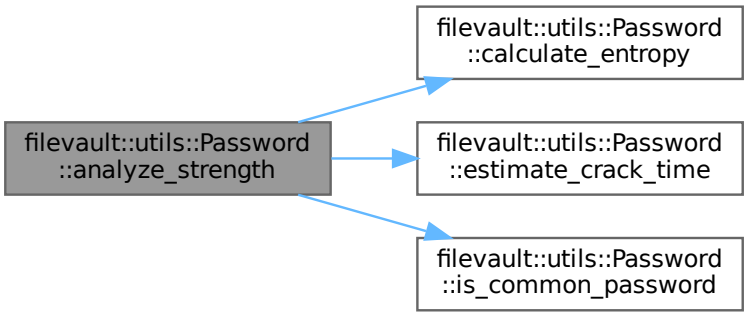
Parameters

password	Password to analyze

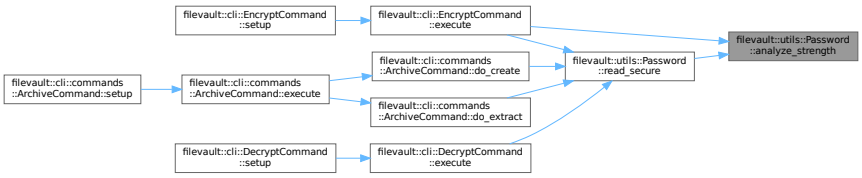
Returns

Detailed analysis with score and recommendations

Here is the call graph for this function:



Here is the caller graph for this function:

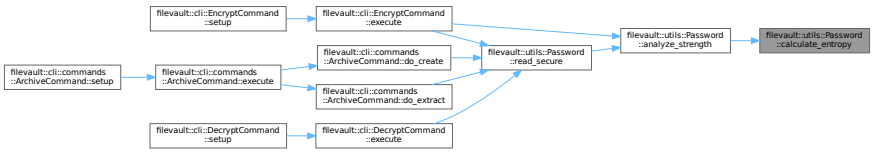


9.64.2.2 calculate_entropy()

```
double filevault::utils::Password::calculate_entropy (
            const std::string & password)  [static], [private]
```

Calculate password entropy.

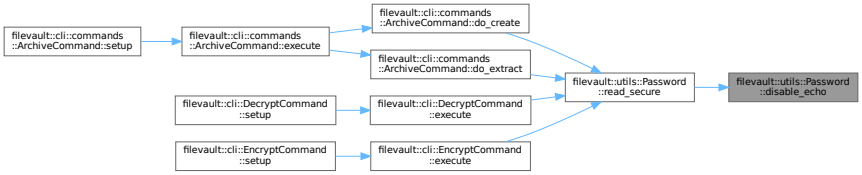
Here is the caller graph for this function:



9.64.2.3 disable_echo()

```
void filevault::utils::Password::disable_echo ()  [static], [private]
```

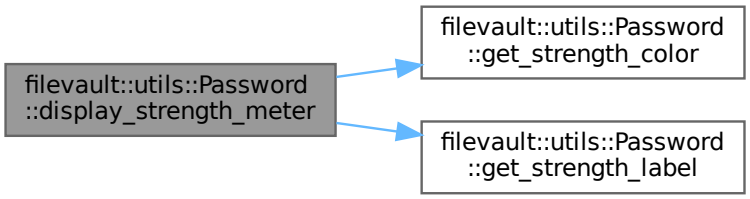
Platform-specific: Disable terminal echo.
Here is the caller graph for this function:



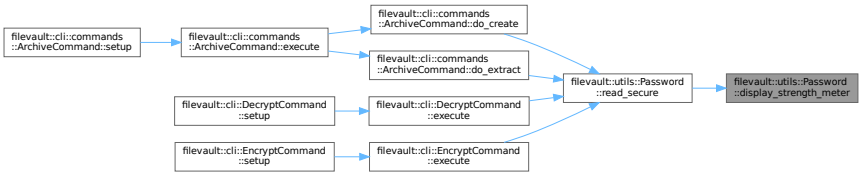
9.64.2.4 display_strength_meter()

```
void filevault::utils::Password::display_strength_meter (
            const core::PasswordAnalysis & analysis)  [static]
```

Display password strength meter.
Here is the call graph for this function:



Here is the caller graph for this function:

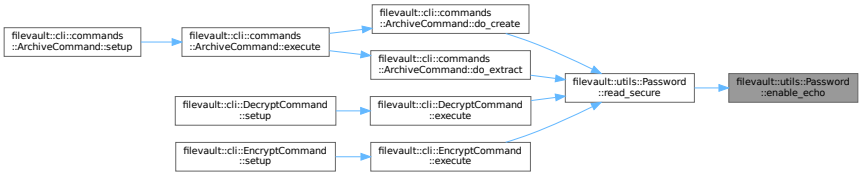


9.64.2.5 enable_echo()

```
void filevault::utils::Password::enable_echo ()  [static], [private]
```

Platform-specific: Enable terminal echo.

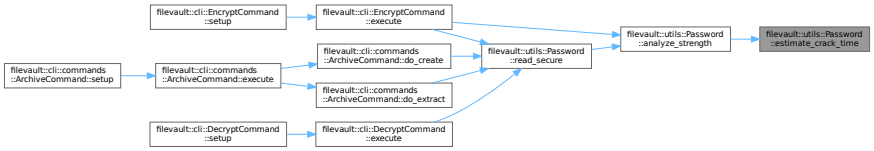
Here is the caller graph for this function:



9.64.2.6 estimate_crack_time()

```
std::pair< std::string, std::string > filevault::utils::Password::estimate_crack_time (
            double entropy,
            const core::PasswordAnalysis & analysis)  [static], [private]
```

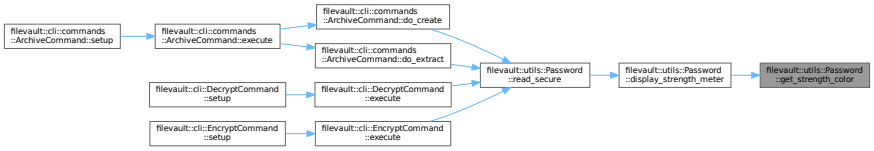
Estimate crack time.
Here is the caller graph for this function:



9.64.2.7 get_strength_color()

```
std::string filevault::utils::Password::get_strength_color (
            core::PasswordStrength strength)  [static]
```

Get strength color for display.
Here is the caller graph for this function:

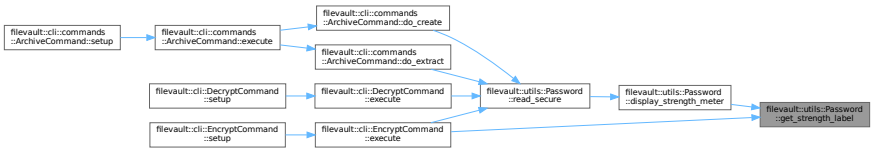


9.64.2.8 get_strength_label()

```
std::string filevault::utils::Password::get_strength_label (
            core::PasswordStrength strength)  [static]
```

Get strength label.

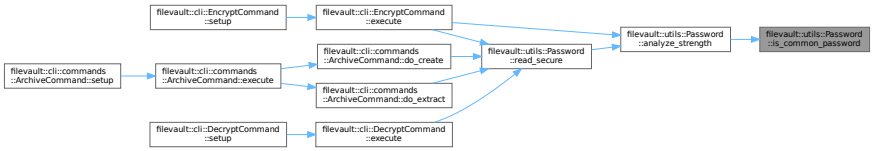
Here is the caller graph for this function:



9.64.2.9 is_common_password()

```
bool filevault::utils::Password::is_common_password (
            const std::string & password)  [static], [private]
```

Check if password is in common password list.
Here is the caller graph for this function:



9.64.2.10 read_secure()

```
std::string filevault::utils::Password::read_secure (
            const std::string & prompt = "Password: ",
            bool confirm = false)  [static]
```

Read password from terminal with masking.

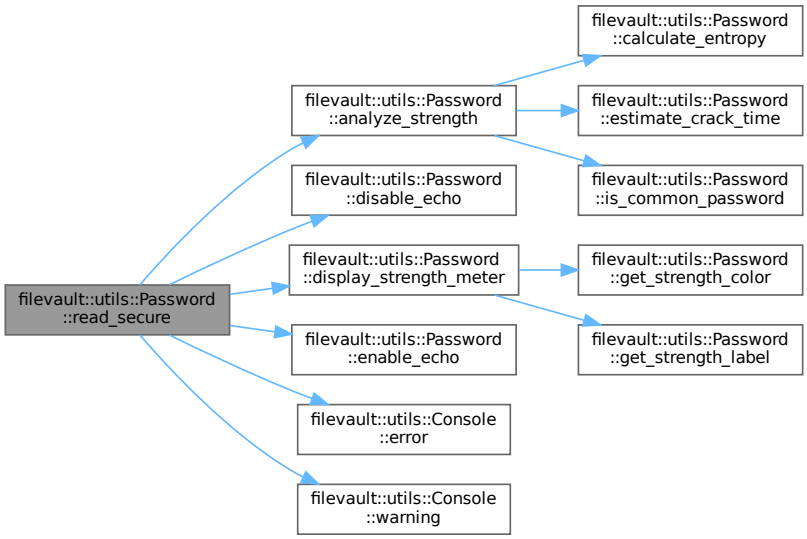
Parameters

prompt	Prompt message to display
confirm	If true, ask for confirmation

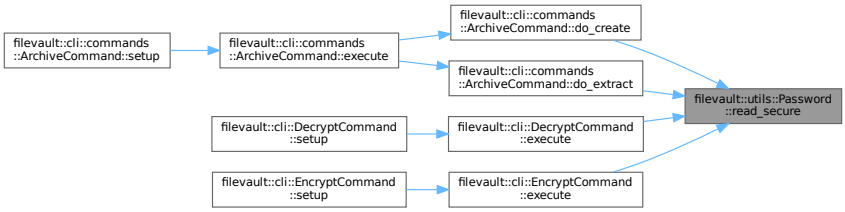
Returns

Password string

Here is the call graph for this function:



Here is the caller graph for this function:



9.64.3 Member Data Documentation

9.64.3.1 common_passwords_

```
const std::vector< std::string > filevault::utils::Password::common_passwords_  [static],
[private]
```

Initial value:

```
= {
    "password", "123456", "12345678", "qwerty", "abc123", "monkey", "1234567",
    "letmein", "trustno1", "dragon", "baseball", "111111", "iloveyou", "master",
    "sunshine", "ashley", "bailey", "passw0rd", "shadow", "123123", "654321",
    "superman", "qazwsx", "michael", "football", "welcome", "jesus", "ninja",
    "mustang", "password1", "123456789", "adobe123", "admin", "1234567890",
    "photoshop", "1234", "12345", "000000", "computer", "test", "qwerty123"
}
```

The documentation for this class was generated from the following files:

- include/filevault/utils/password.hpp
- src/utils/password.cpp

9.65 filevault::core::PasswordAnalysis Struct Reference

Password strength analysis result.
`#include <types.hpp>`

Public Attributes

- PasswordStrength strength
- int score
- std::vector< std::string > warnings
- std::vector< std::string > suggestions
- size_t length
- bool has_lowercase
- bool has_uppercase
- bool has_digits
- bool has_special
- bool has_repeated_chars
- bool is_common_password
- std::string crack_time_online
- std::string crack_time_offline

9.65.1 Detailed Description

Password strength analysis result.

9.65.2 Member Data Documentation

9.65.2.1 crack_time_offline

```
std::string filevault::core::PasswordAnalysis::crack_time_offline
```

9.65.2.2 crack_time_online

```
std::string filevault::core::PasswordAnalysis::crack_time_online
```

9.65.2.3 has_digits

```
bool filevault::core::PasswordAnalysis::has_digits
```

9.65.2.4 has_lowercase

```
bool filevault::core::PasswordAnalysis::has_lowercase
```

9.65.2.5 has_repeated_chars

```
bool filevault::core::PasswordAnalysis::has_repeated_chars
```

9.65.2.6 has_special

```
bool filevault::core::PasswordAnalysis::has_special
```

9.65.2.7 has_uppercase

```
bool filevault::core::PasswordAnalysis::has_uppercase
```

9.65.2.8 is_common_password

```
bool filevault::core::PasswordAnalysis::is_common_password
```

9.65.2.9 length

```
size_t filevault::core::PasswordAnalysis::length
```

9.65.2.10 score

```
int filevault::core::PasswordAnalysis::score
```

9.65.2.11 strength

```
PasswordStrength filevault::core::PasswordAnalysis::strength
```

9.65.2.12 suggestions

```
std::vector<std::string> filevault::core::PasswordAnalysis::suggestions
```

9.65.2.13 warnings

```
std::vector<std::string> filevault::core::PasswordAnalysis::warnings
```

The documentation for this struct was generated from the following file:

- include/filevault/core/types.hpp

9.66 filevault::core::PBKDF2Params Struct Reference

KDF parameters for PBKDF2.

`#include <file_format.hpp>`

Public Member Functions

- std::vector< uint8_t > serialize () const

Static Public Member Functions

- static PBKDF2Params deserialize (std::span< const uint8_t > data)

Public Attributes

- uint32_t iterations = 100000

9.66.1 Detailed Description

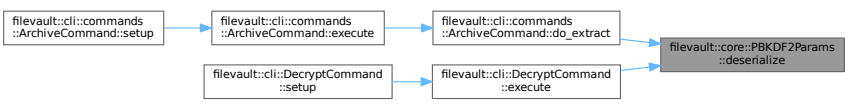
KDF parameters for PBKDF2.

9.66.2 Member Function Documentation

9.66.2.1 deserialize()

```
PBKDF2Params filevault::core::PBKDF2Params::deserialize (
            std::span< const uint8_t > data)  [static]
```

Here is the caller graph for this function:



9.66.2.2 serialize()

`std::vector< uint8_t > filevault::core::PBKDF2Params::serialize () const`
Here is the caller graph for this function:

9.66.3 Member Data Documentation

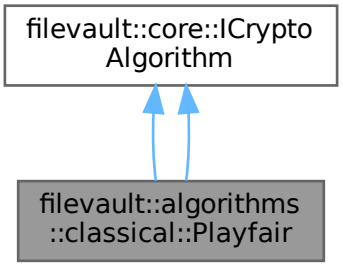
9.66.3.1 iterations

`uint32_t filevault::core::PBKDF2Params::iterations = 100000`
The documentation for this struct was generated from the following files:

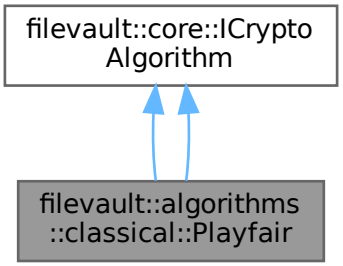
- include/filevault/core/file_format.hpp
- src/format/file_format.cpp

9.67 filevault::algorithms::classical::Playfair Class Reference

Playfair Cipher - Digraph substitution.
`#include <classical_ciphers.hpp>`
Inheritance diagram for filevault::algorithms::classical::Playfair:



Collaboration diagram for filevault::algorithms::classical::Playfair:



Public Member Functions

- Playfair (const std::string &keyword="KEYWORD")
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- size_t key_size () const override

  Get recommended key size in bytes.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.
- Playfair (const std::string &keyword="KEYWORD")
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- size_t key_size () const override

  Get recommended key size in bytes.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Member Functions

- void build_matrix (const std::string &keyword)
- std::pair< int, int > find_position (char ch) const
- void build_matrix (const std::string &keyword)
- std::pair< int, int > find_position (char ch) const

Private Attributes

- char matrix_ [5][5]

9.67.1 Detailed Description

Playfair Cipher - Digraph substitution.
Uses a 5x5 matrix based on a keyword. First practical digraph substitution cipher.
Security: BROKEN - 600 possible digraphs still analyzable Purpose: Educational - shows digraph cryptanalysis
Uses a 5x5 matrix based on a keyword. First practical digraph substitution cipher.
Security: BROKEN - 600 possible digraphs still analyzable Purpose: Educational - shows digraph cryptanalysis

See also

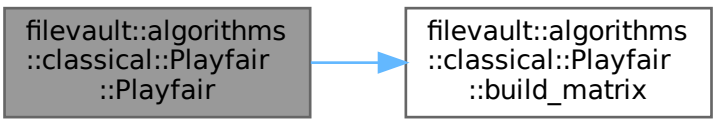
https://en.wikipedia.org/wiki/Playfair_cipher

9.67.2 Constructor & Destructor Documentation

9.67.2.1 Playfair() [1/2]

```
filevault::algorithms::classical::Playfair::Playfair (
            const std::string & keyword = "KEYWORD")  [explicit]
```

Here is the call graph for this function:



9.67.2.2 Playfair() [2/2]

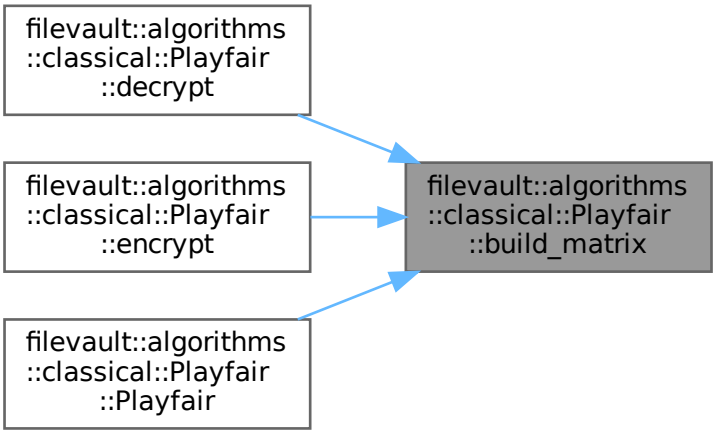
```
filevault::algorithms::classical::Playfair::Playfair (
            const std::string & keyword = "KEYWORD")  [explicit]
```

9.67.3 Member Function Documentation

9.67.3.1 build_matrix() [1/2]

```
void filevault::algorithms::classical::Playfair::build_matrix (
            const std::string & keyword)  [private]
```

Here is the caller graph for this function:



9.67.3.2 build_matrix() [2/2]

```
void filevault::algorithms::classical::Playfair::build_matrix (
            const std::string & keyword)  [private]
```

9.67.3.3 decrypt() [1/2]

```
core::CryptoResult filevault::algorithms::classical::Playfair::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

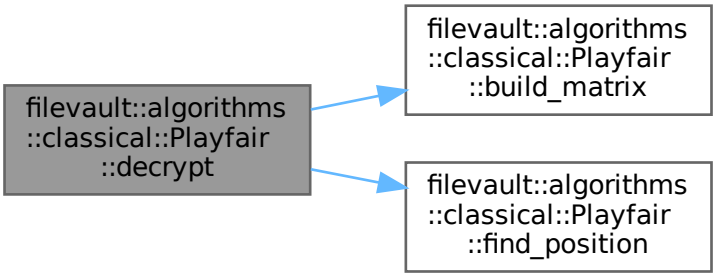
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.67.3.4 decrypt() [2/2]

```
core::CryptoResult filevault::algorithms::classical::Playfair::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

9.67.3.5 encrypt() [1/2]

```
core::CryptoResult filevault::algorithms::classical::Playfair::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

Parameters

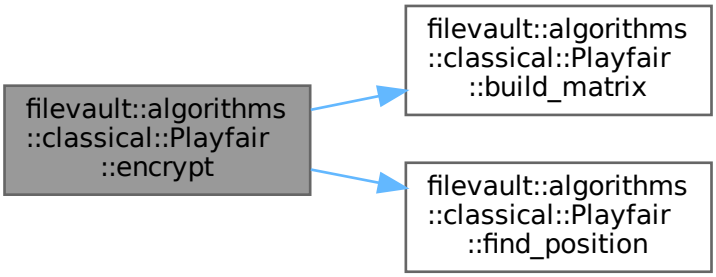
plaintext	Input data to encrypt

key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.67.3.6 encrypt() [2/2]

```
core::CryptoResult filevault::algorithms::classical::Playfair::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

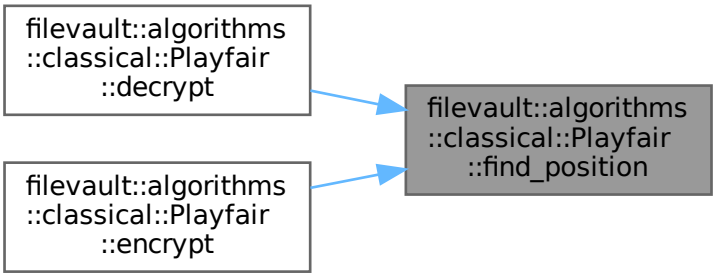
Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.

9.67.3.7 find_position() [1/2]

```
std::pair< int, int > filevault::algorithms::classical::Playfair::find_position (
            char ch) const  [private]
```

Here is the caller graph for this function:



9.67.3.8 find_position() [2/2]

```
std::pair< int, int > filevault::algorithms::classical::Playfair::find_position (
            char ch) const  [private]
```

9.67.3.9 is_suitable_for() [1/2]

```
bool filevault::algorithms::classical::Playfair::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.67.3.10 is_suitable_for() [2/2]

```
bool filevault::algorithms::classical::Playfair::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.67.3.11 key_size() [1/2]

```
size_t filevault::algorithms::classical::Playfair::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.67.3.12 key_size() [2/2]

```
size_t filevault::algorithms::classical::Playfair::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.67.3.13 name() [1/2]

```
std::string filevault::algorithms::classical::Playfair::name () const  [inline], [override],
[virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.67.3.14 name() [2/2]

```
std::string filevault::algorithms::classical::Playfair::name () const  [inline], [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.67.3.15 type() [1/2]

```
core::AlgorithmType filevault::algorithms::classical::Playfair::type () const  [inline], [override], [virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.67.3.16 type() [2/2]

```
core::AlgorithmType filevault::algorithms::classical::Playfair::type () const  [inline], [override], [virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.67.4 Member Data Documentation

9.67.4.1 matrix_

```
char filevault::algorithms::classical::Playfair::matrix_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/classical/classical_ciphers.hpp
- include/filevault/algorithms/classical/playfair.hpp
- src/algorithms/classical/classical_ciphers.cpp
- src/algorithms/classical/playfair.cpp

9.68 filevault::cli::PQCBenchmarkResult Struct Reference

```
#include <benchmark_cmd.hpp>
```

Public Attributes

- std::string algorithm
- double keygen_ms = 0
- double encaps_ms = 0
- double decaps_ms = 0
- bool success = false

9.68.1 Member Data Documentation

9.68.1.1 algorithm

```
std::string filevault::cli::PQCBenchmarkResult::algorithm
```

9.68.1.2 decaps_ms

```
double filevault::cli::PQCBenchmarkResult::decaps_ms = 0
```

9.68.1.3 encaps_ms

```
double filevault::cli::PQCBenchmarkResult::encaps_ms = 0
```

9.68.1.4 keygen_ms

```
double filevault::cli::PQCBenchmarkResult::keygen_ms = 0
```

9.68.1.5 success

```
bool filevault::cli::PQCBenchmarkResult::success = false
```

The documentation for this struct was generated from the following file:

- include/filevault/cli/commands/benchmark_cmd.hpp

9.69 filevault::algorithms::pqc::PQKeyPair Struct Reference

Key pair for post-quantum algorithms.

`#include <post_quantum.hpp>`

Public Attributes

- std::vector< uint8_t > public_key
- std::vector< uint8_t > private_key
- std::string algorithm

9.69.1 Detailed Description

Key pair for post-quantum algorithms.

9.69.2 Member Data Documentation

9.69.2.1 algorithm

```
std::string filevault::algorithms::pqc::PQKeyPair::algorithm
```

9.69.2.2 private_key

```
std::vector<uint8_t> filevault::algorithms::pqc::PQKeyPair::private_key
```

9.69.2.3 public_key

```
std::vector<uint8_t> filevault::algorithms::pqc::PQKeyPair::public_key
```

The documentation for this struct was generated from the following file:

- include/filevault/algorithms/pqc/post_quantum.hpp

9.70 filevault::utils::ProgressBar Class Reference

Modern progress bar wrapper.

`#include <progress.hpp>`

Public Member Functions

- ProgressBar (const std::string &prefix, size_t max_progress=100)
- ~ProgressBar ()
- void set_progress (size_t progress)
- void tick ()
- void set_postfix (const std::string &text)
- void mark_as_completed ()
- void hide ()
- void show ()

Private Attributes

- std::unique_ptr< indicators::ProgressBar > bar_
- size_t current_progress_
- size_t max_progress_

9.70.1 Detailed Description

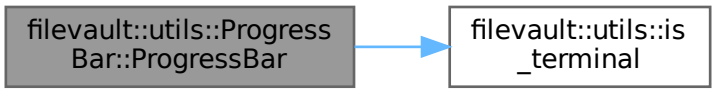
Modern progress bar wrapper.

9.70.2 Constructor & Destructor Documentation

9.70.2.1 ProgressBar()

```
filevault::utils::ProgressBar::ProgressBar (
            const std::string & prefix,
            size_t max_progress = 100)
```

Here is the call graph for this function:



9.70.2.2 ~ProgressBar()

```
filevault::utils::ProgressBar::~ProgressBar ()
```

9.70.3 Member Function Documentation

9.70.3.1 hide()

```
void filevault::utils::ProgressBar::hide ()
```

9.70.3.2 mark_as_completed()

```
void filevault::utils::ProgressBar::mark_as_completed ()
```

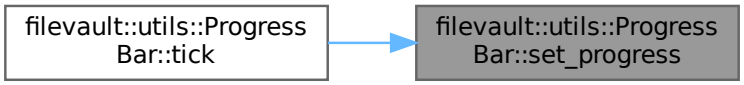
9.70.3.3 set_postfix()

```
void filevault::utils::ProgressBar::set_postfix (
            const std::string & text)
```

9.70.3.4 set_progress()

```
void filevault::utils::ProgressBar::set_progress (
            size_t progress)
```

Here is the caller graph for this function:



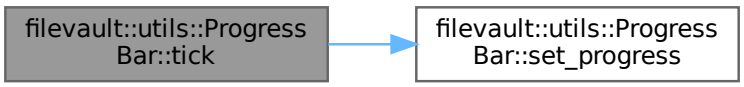
9.70.3.5 show()

```
void filevault::utils::ProgressBar::show ()
```

9.70.3.6 tick()

```
void filevault::utils::ProgressBar::tick ()
```

Here is the call graph for this function:



9.70.4 Member Data Documentation

9.70.4.1 bar_

```
std::unique_ptr<indicators::ProgressBar> filevault::utils::ProgressBar::bar_  [private]
```

9.70.4.2 current_progress_

```
size_t filevault::utils::ProgressBar::current_progress_  [private]
```

9.70.4.3 max_progress_

```
size_t filevault::utils::ProgressBar::max_progress_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/utils/progress.hpp
- src/utils/progress.cpp

9.71 filevault::core::Result< T > Struct Template Reference

Generic result type.

```
#include <result.hpp>
```

Public Member Functions

- operator bool () const
- T & unwrap ()

Static Public Member Functions

- static Result< T > ok (T val)
- static Result< T > error (const std::string &msg)

Public Attributes

- bool success = false
- std::string error_message
- T value

9.71.1 Detailed Description

template<typename T>
struct filevault::core::Result< T >

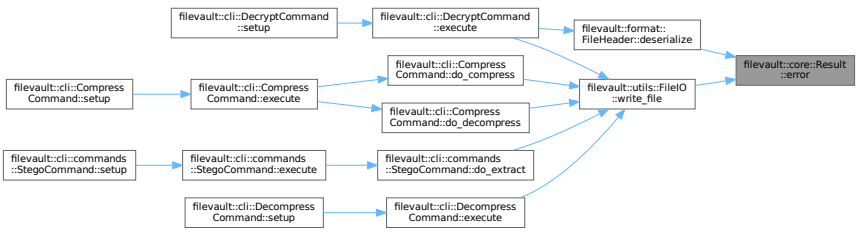
Generic result type.

9.71.2 Member Function Documentation

9.71.2.1 error()

```
template<typename T>
Result< T > filevault::core::Result< T >::error (
            const std::string & msg)  [inline], [static]
```

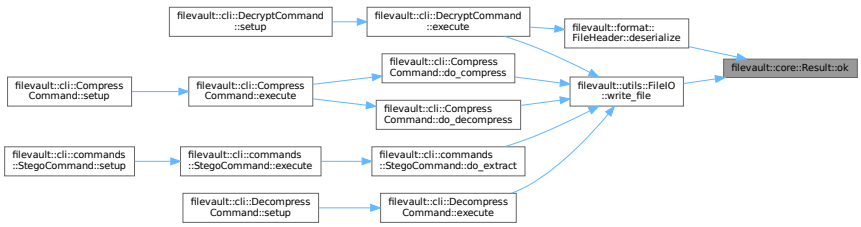
Here is the caller graph for this function:



9.71.2.2 ok()

```
template<typename T>
Result< T > filevault::core::Result< T >::ok (
            T val)  [inline], [static]
```

Here is the caller graph for this function:



9.71.2.3 operator bool()

```
template<typename T>
filevault::core::Result< T >::operator bool () const  [inline], [explicit]
```

9.71.2.4 unwrap()

```
template<typename T>
T & filevault::core::Result< T >::unwrap ()  [inline]
```

9.71.3 Member Data Documentation

9.71.3.1 error_message

```
template<typename T>
std::string filevault::core::Result< T >::error_message
```

9.71.3.2 success

```
template<typename T>
bool filevault::core::Result< T >::success = false
```

9.71.3.3 value

```
template<typename T>
T filevault::core::Result< T >::value
```

The documentation for this struct was generated from the following file:

- include/filevault/core/result.hpp

9.72 filevault::core::Result< void > Struct Reference

Specialization for void.

`#include <result.hpp>`

Public Member Functions

- operator bool () const
- void & unwrap ()

Static Public Member Functions

- static Result< void > ok ()
- static Result< void > error (const std::string &msg)

Public Attributes

- bool success = false
- std::string error_message
- void value

9.72.1 Detailed Description

Specialization for void.

9.72.2 Member Function Documentation

9.72.2.1 error()

```
Result< void > filevault::core::Result< void >::error (
            const std::string & msg)  [inline], [static]
```

9.72.2.2 ok()

```
Result< void > filevault::core::Result< void >::ok ()  [inline], [static]
```

9.72.2.3 operator bool()

```
filevault::core::Result< void >::operator bool () const  [inline], [explicit]
```

9.72.2.4 unwrap()

```
void & filevault::core::Result< void >::unwrap ()  [inline]
```

9.72.3 Member Data Documentation

9.72.3.1 error_message

```
std::string filevault::core::Result< void >::error_message
```

9.72.3.2 success

```
bool filevault::core::Result< void >::success = false
```

9.72.3.3 value

```
void filevault::core::Result< void >::value
```

The documentation for this struct was generated from the following file:

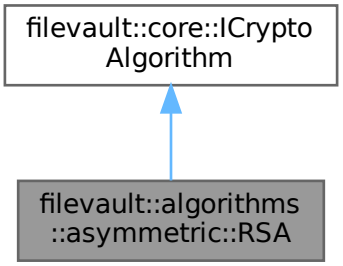
- include/filevault/core/result.hpp

9.73 filevault::algorithms::asymmetric::RSA Class Reference

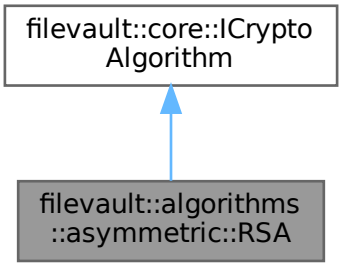
RSA asymmetric encryption algorithm.

```
#include <rsa.hpp>
```

Inheritance diagram for filevault::algorithms::asymmetric::RSA:



Collaboration diagram for filevault::algorithms::asymmetric::RSA:



Public Member Functions

- RSA (size_t key_bits=2048)

  Construct RSA with specified key size.
- ~RSA () override=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt with public key.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt with private key.
- RSAKeyPair generate_key_pair ()

  Generate a new RSA key pair.
- std::vector< uint8_t > sign (std::span< const uint8_t > data, std::span< const uint8_t > private_key)

  Sign data with private key.

- bool verify (std::span< const uint8_t > data, std::span< const uint8_t > signature, std::span< const uint8_t > public_key)

  Verify signature with public key.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t max_plaintext_size () const

  Maximum plaintext size that can be encrypted For OAEP with SHA-256: key_bytes - 2∗hash_len - 2 = key_bytes - 66.
- bool requires_padding () const
- bool is_authenticated () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_

9.73.1 Detailed Description

RSA asymmetric encryption algorithm.
Features:

- Public key encryption, private key decryption
- Key sizes: 2048, 3072, 4096 bits
- OAEP padding (PKCS#1 v2.1) for encryption
- PSS padding for signatures

Note: RSA encrypts limited data (key_size - padding overhead). For large data, use hybrid encryption (RSA + symmetric cipher).

9.73.2 Constructor & Destructor Documentation

9.73.2.1 RSA()

```
filevault::algorithms::asymmetric::RSA::RSA (
            size_t key_bits = 2048)  [explicit]
```

Construct RSA with specified key size.

Parameters

key_bits	Key size in bits (2048, 3072, or 4096)

9.73.2.2 ~RSA()

```
filevault::algorithms::asymmetric::RSA::~RSA ()  [override], [default]
```

9.73.3 Member Function Documentation

9.73.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::asymmetric::RSA::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt with private key.

Parameters

ciphertext	Data to decrypt
key	Private key (PEM or DER encoded)
config	Encryption configuration

Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:

9.73.3.2 encrypt()

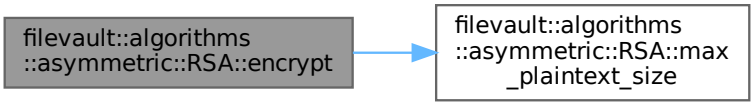
```
core::CryptoResult filevault::algorithms::asymmetric::RSA::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt with public key.

Parameters

plaintext	Data to encrypt (limited by key size)
key	Public key (PEM or DER encoded)
config	Encryption configuration

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:

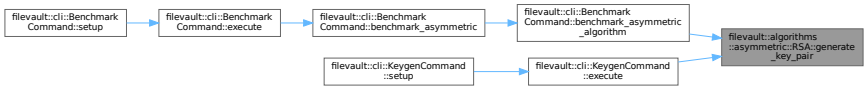
9.73.3.3 generate_key_pair()

`RSAKeyPair filevault::algorithms::asymmetric::RSA::generate_key_pair ()`
Generate a new RSA key pair.

Returns

RSAKeyPair containing public and private keys

Here is the caller graph for this function:



9.73.3.4 is_authenticated()

`bool filevault::algorithms::asymmetric::RSA::is_authenticated () const [inline]`

9.73.3.5 is_suitable_for()

```
bool filevault::algorithms::asymmetric::RSA::is_suitable_for (
            core::SecurityLevel level) const [override], [virtual]
```
Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.73.3.6 key_size()

`size_t filevault::algorithms::asymmetric::RSA::key_size () const [inline], [override], [virtual]`
Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.73.3.7 max_plaintext_size()

`size_t filevault::algorithms::asymmetric::RSA::max_plaintext_size () const [inline]`
Maximum plaintext size that can be encrypted For OAEP with SHA-256: key_bytes - 2∗hash_len - 2 = key_bytes - 66.
Here is the caller graph for this function:

9.73.3.8 name()

```
std::string filevault::algorithms::asymmetric::RSA::name () const  [override], [virtual]
```

Get algorithm name.

Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:

9.73.3.9 requires_padding()

```
bool filevault::algorithms::asymmetric::RSA::requires_padding () const  [inline]
```

9.73.3.10 sign()

```
std::vector< uint8_t > filevault::algorithms::asymmetric::RSA::sign (
            std::span< const uint8_t > data,
            std::span< const uint8_t > private_key)
```

Sign data with private key.

Parameters

data	Data to sign
private_key	Private key (PEM encoded)

Returns

Signature bytes

Here is the caller graph for this function:

9.73.3.11 type()

```
core::AlgorithmType filevault::algorithms::asymmetric::RSA::type () const  [override], [virtual]
```

Get algorithm type.

Implements filevault::core::ICryptoAlgorithm.

9.73.3.12 verify()

```
bool filevault::algorithms::asymmetric::RSA::verify (
            std::span< const uint8_t > data,
            std::span< const uint8_t > signature,
            std::span< const uint8_t > public_key)
```

Verify signature with public key.

Parameters

data	Original data
signature	Signature to verify
public_key	Public key (PEM encoded)

Returns

true if signature is valid

Here is the caller graph for this function:

9.73.4 Member Data Documentation

9.73.4.1 key_bits_

```
size_t filevault::algorithms::asymmetric::RSA::key_bits_  [private]
```

9.73.4.2 type_

```
core::AlgorithmType filevault::algorithms::asymmetric::RSA::type_  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/algorithms/asymmetric/rsa.hpp
- src/algorithms/asymmetric/rsa.cpp

9.74 filevault::algorithms::asymmetric::RSAKeyPair Struct Reference

RSA key pair.

```
#include <rsa.hpp>
```

Public Attributes

- std::vector< uint8_t > public_key
- std::vector< uint8_t > private_key
- size_t bits

9.74.1 Detailed Description

RSA key pair.

9.74.2 Member Data Documentation

9.74.2.1 bits

```
size_t filevault::algorithms::asymmetric::RSAKeyPair::bits
```

9.74.2.2 private_key

```
std::vector<uint8_t> filevault::algorithms::asymmetric::RSAKeyPair::private_key
```

9.74.2.3 public_key

```
std::vector<uint8_t> filevault::algorithms::asymmetric::RSAKeyPair::public_key
```

The documentation for this struct was generated from the following file:

- include/filevault/algorithms/asymmetric/rsa.hpp

9.75 filevault::core::ScryptParams Struct Reference

KDF parameters for Scrypt.

```
#include <file_format.hpp>
```

Public Member Functions

- std::vector< uint8_t > serialize () const

Static Public Member Functions

- static ScryptParams deserialize (std::span< const uint8_t > data)

Public Attributes

- uint32_t n = 32768
- uint32_t r = 8
- uint32_t p = 1

9.75.1 Detailed Description

KDF parameters for Scrypt.

9.75.2 Member Function Documentation

9.75.2.1 deserialize()

```
ScryptParams filevault::core::ScryptParams::deserialize (
            std::span< const uint8_t > data)  [static]
```

9.75.2.2 serialize()

```
std::vector< uint8_t > filevault::core::ScryptParams::serialize () const
```

Here is the caller graph for this function:

9.75.3 Member Data Documentation

9.75.3.1 n

```
uint32_t filevault::core::ScryptParams::n = 32768
```

9.75.3.2 p

```
uint32_t filevault::core::ScryptParams::p = 1
```

9.75.3.3 r

```
uint32_t filevault::core::ScryptParams::r = 8
```

The documentation for this struct was generated from the following files:

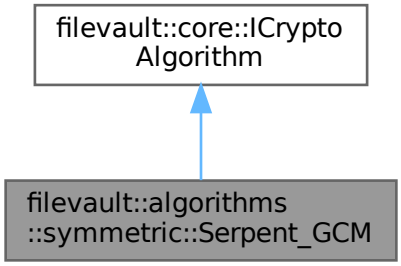
- include/filevault/core/file_format.hpp
- src/format/file_format.cpp

9.76 filevault::algorithms::symmetric::Serpent_GCM Class Reference

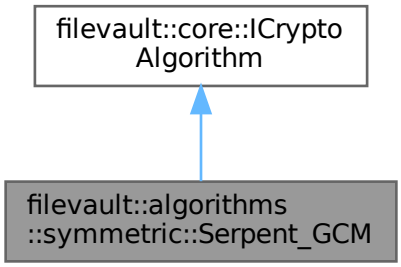
Serpent-256-GCM implementation.

```
#include <serpent_gcm.hpp>
```

Inheritance diagram for filevault::algorithms::symmetric::Serpent_GCM:



Collaboration diagram for filevault::algorithms::symmetric::Serpent_GCM:



Public Member Functions

- Serpent_GCM ()

  Construct Serpent-256-GCM cipher.
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.

- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt plaintext with Serpent-256-GCM.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt ciphertext with Serpent-256-GCM.
- size_t key_size () const override

  Get required key size in bytes.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Member Functions

- std::vector< uint8_t > process_gcm (std::span< const uint8_t > input, std::span< const uint8_t > key, std::span< const uint8_t > nonce, std::span< const uint8_t > tag, bool encrypt)

  Perform encryption/decryption operation.

9.76.1 Detailed Description

Serpent-256-GCM implementation.
Serpent is a 128-bit block cipher designed by Ross Anderson, Eli Biham, and Lars Knudsen. It was an AES finalist and is known for its high security margin.
References:

- Serpent: A Candidate Block Cipher for the AES https://www.cl.cam.ac.uk/~rja14/serpent.html
- NIST SP 800-38D: GCM Mode https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublicat pdf

Security:

- Key size: 256 bits (32 bytes)
- Block size: 128 bits (16 bytes)
- Nonce size: 96 bits (12 bytes) for GCM
- Tag size: 128 bits (16 bytes) for authentication

Note

This implementation uses Botan's Serpent/GCM mode

Serpent-256 provides very high security margin

Slower than AES on most platforms but more secure

9.76.2 Constructor & Destructor Documentation

9.76.2.1 Serpent_GCM()

`filevault::algorithms::symmetric::Serpent_GCM::Serpent_GCM ()`
Construct Serpent-256-GCM cipher.

9.76.3 Member Function Documentation

9.76.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::Serpent_GCM::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt ciphertext with Serpent-256-GCM.

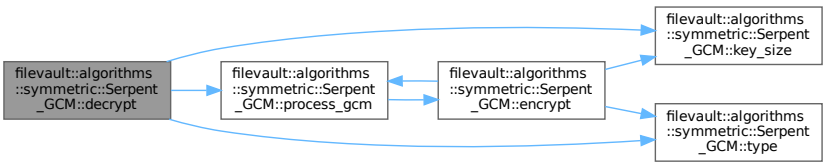
Parameters

ciphertext	Encrypted data
key	256-bit decryption key
config	Decryption configuration with nonce and tag

Returns

Decrypted plaintext

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.76.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::Serpent_GCM::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt plaintext with Serpent-256-GCM.

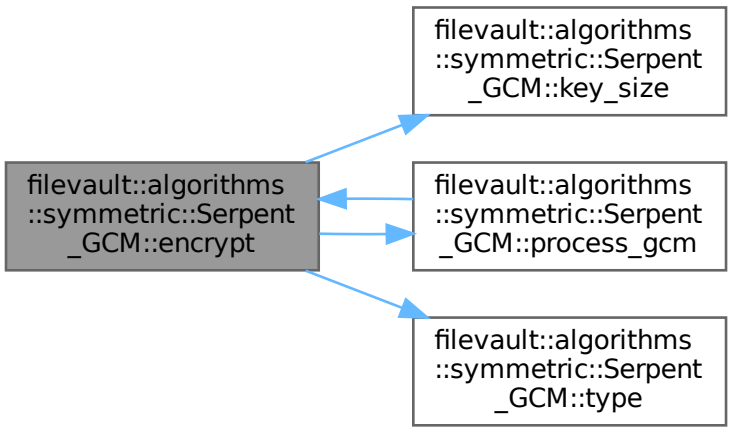
Parameters

plaintext	Input data to encrypt
key	256-bit encryption key
config	Encryption configuration with nonce

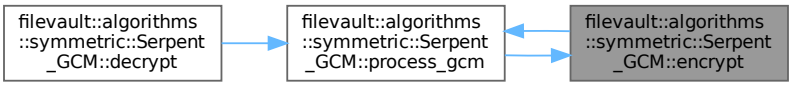
Returns

Encrypted data with authentication tag

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:



9.76.3.3 is_suitable_for()

```
bool filevault::algorithms::symmetric::Serpent_GCM::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

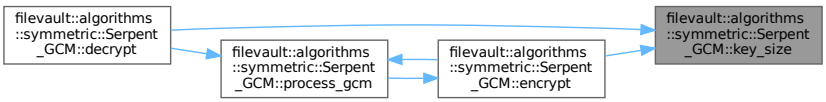
Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.76.3.4 key_size()

```
size_t filevault::algorithms::symmetric::Serpent_GCM::key_size () const  [override], [virtual]
```

Get required key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



9.76.3.5 name()

```
std::string filevault::algorithms::symmetric::Serpent_GCM::name () const  [override], [virtual]
```

Get algorithm name.

Implements filevault::core::ICryptoAlgorithm.

9.76.3.6 process_gcm()

```
std::vector< uint8_t > filevault::algorithms::symmetric::Serpent_GCM::process_gcm (
            std::span< const uint8_t > input,
            std::span< const uint8_t > key,
            std::span< const uint8_t > nonce,
            std::span< const uint8_t > tag,
            bool encrypt)  [private]
```

Perform encryption/decryption operation.

Parameters

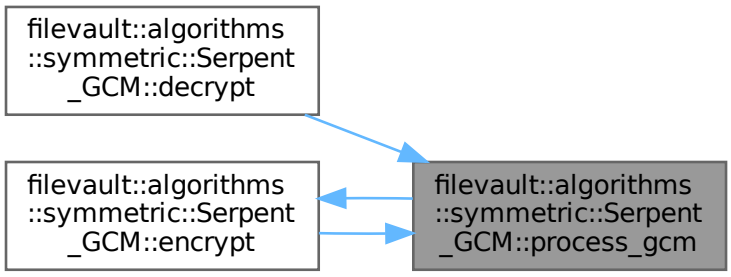
input	Input data
key	Encryption/decryption key
nonce	Unique nonce (96 bits)
tag	Authentication tag (for decryption, empty for encryption)
encrypt	True for encryption, false for decryption

Returns

Encrypted/decrypted data (with tag appended for encryption)

Here is the call graph for this function:

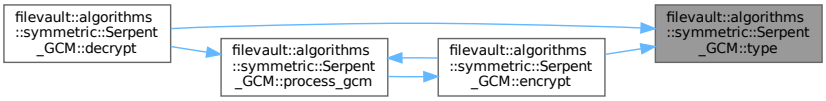
Here is the caller graph for this function:



9.76.3.7 type()

```
core::AlgorithmType filevault::algorithms::symmetric::Serpent_GCM::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/serpent_gcm.hpp
- src/algorithms/symmetric/serpent_gcm.cpp

9.77 filevault::cli::SignatureBenchmarkResult Struct Reference

```
#include <benchmark_cmd.hpp>
```

Public Attributes

- std::string algorithm
- double keygen_ms = 0
- double sign_ms = 0
- double verify_ms = 0
- bool success = false

9.77.1 Member Data Documentation

9.77.1.1 algorithm

```
std::string filevault::cli::SignatureBenchmarkResult::algorithm
```

9.77.1.2 keygen_ms

```
double filevault::cli::SignatureBenchmarkResult::keygen_ms = 0
```

9.77.1.3 sign_ms

```
double filevault::cli::SignatureBenchmarkResult::sign_ms = 0
```

9.77.1.4 success

```
bool filevault::cli::SignatureBenchmarkResult::success = false
```

9.77.1.5 verify_ms

```
double filevault::cli::SignatureBenchmarkResult::verify_ms = 0
```

The documentation for this struct was generated from the following file:

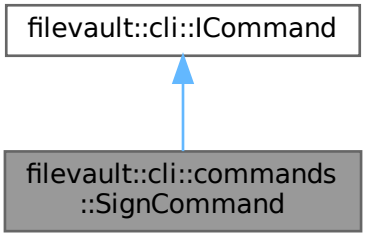
- include/filevault/cli/commands/benchmark_cmd.hpp

9.78 filevault::cli::commands::SignCommand Class Reference

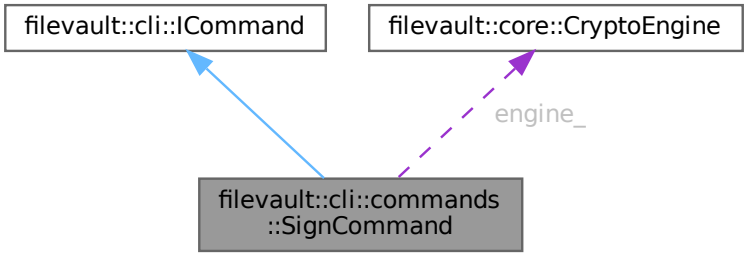
Command to create digital signature for a file.

`#include <sign_cmd.hpp>`

Inheritance diagram for filevault::cli::commands::SignCommand:



Collaboration diagram for filevault::cli::commands::SignCommand:



Public Member Functions

- SignCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- std::string name_ = "sign"
- std::string description_ = "Create digital signature for a file"
- core::CryptoEngine & engine_
- std::string file_path_
- std::string private_key_path_
- std::string output_path_
- std::string algorithm_ = "rsa"

9.78.1 Detailed Description

Command to create digital signature for a file.

9.78.2 Constructor & Destructor Documentation

9.78.2.1 SignCommand()

```
filevault::cli::commands::SignCommand::SignCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.78.3 Member Function Documentation

9.78.3.1 description()

```
std::string filevault::cli::commands::SignCommand::description () const  [inline], [override],
[virtual]
```

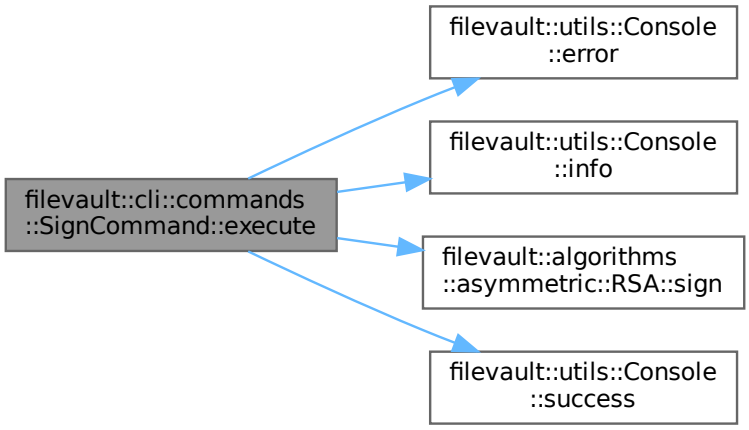
Get command description.
Implements filevault::cli::ICommand.

9.78.3.2 execute()

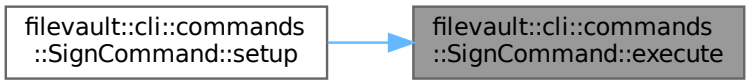
```
int filevault::cli::commands::SignCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



9.78.3.3 name()

```
std::string filevault::cli::commands::SignCommand::name () const  [inline], [override], [virtual]
```

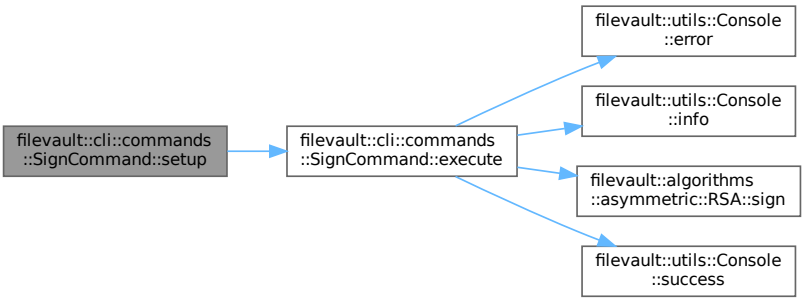
Get command name.
Implements filevault::cli::ICommand.

9.78.3.4 setup()

```
void filevault::cli::commands::SignCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.

Here is the call graph for this function:



9.78.4 Member Data Documentation

9.78.4.1 algorithm_

```
std::string filevault::cli::commands::SignCommand::algorithm_ = "rsa"  [private]
```

9.78.4.2 description_

```
std::string filevault::cli::commands::SignCommand::description_ = "Create digital signature
for a file"  [private]
```

9.78.4.3 engine_

```
core::CryptoEngine& filevault::cli::commands::SignCommand::engine_  [private]
```

9.78.4.4 file_path_

```
std::string filevault::cli::commands::SignCommand::file_path_  [private]
```

9.78.4.5 name_

```
std::string filevault::cli::commands::SignCommand::name_ = "sign"  [private]
```

9.78.4.6 output_path_

```
std::string filevault::cli::commands::SignCommand::output_path_  [private]
```

9.78.4.7 private_key_path_

```
std::string filevault::cli::commands::SignCommand::private_key_path_  [private]
```

The documentation for this class was generated from the following files:

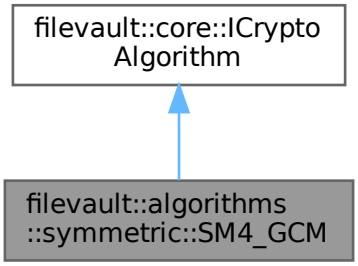
- include/filevault/cli/commands/sign_cmd.hpp
- src/cli/commands/sign_cmd.cpp

9.79 filevault::algorithms::symmetric::SM4_GCM Class Reference

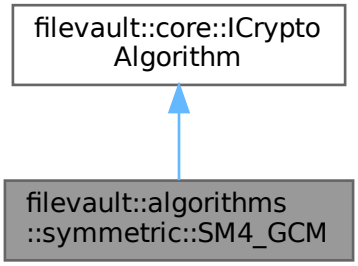
SM4-GCM AEAD encryption.

```
#include <sm4_gcm.hpp>
```

Inheritance diagram for filevault::algorithms::symmetric::SM4_GCM:



Collaboration diagram for filevault::algorithms::symmetric::SM4_GCM:



Public Member Functions

- SM4_GCM ()
- virtual ~SM4_GCM ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- size_t nonce_size () const
- size_t tag_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

9.79.1 Detailed Description

SM4-GCM AEAD encryption.
SM4 only supports 128-bit keys. Uses GCM mode for authenticated encryption.

9.79.2 Constructor & Destructor Documentation

9.79.2.1 SM4_GCM()

```
filevault::algorithms::symmetric::SM4_GCM::SM4_GCM ()
```

9.79.2.2 ~SM4_GCM()

```
virtual filevault::algorithms::symmetric::SM4_GCM::~SM4_GCM ()  [virtual], [default]
```

9.79.3 Member Function Documentation

9.79.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::SM4_GCM::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

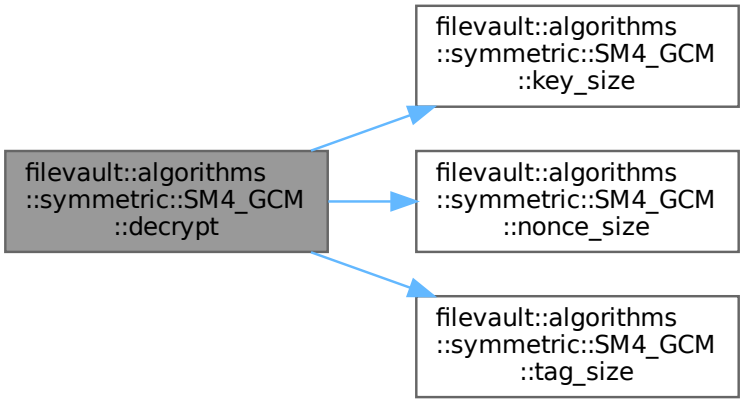
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.79.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::SM4_GCM::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

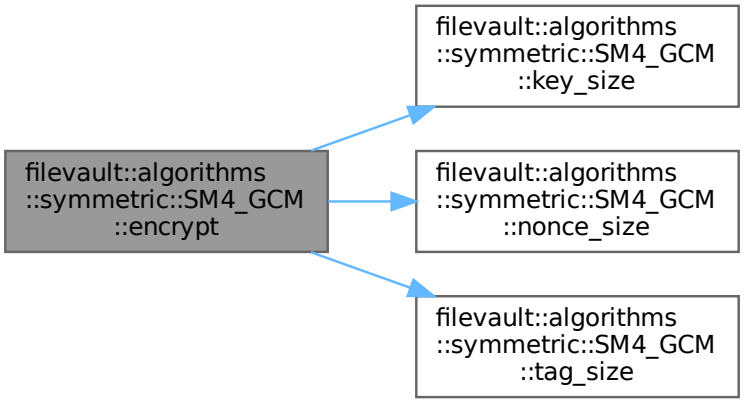
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.79.3.3 is_suitable_for()

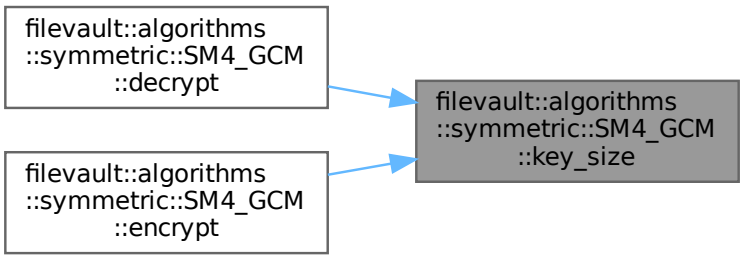
```
bool filevault::algorithms::symmetric::SM4_GCM::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.79.3.4 key_size()

```
size_t filevault::algorithms::symmetric::SM4_GCM::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.79.3.5 name()

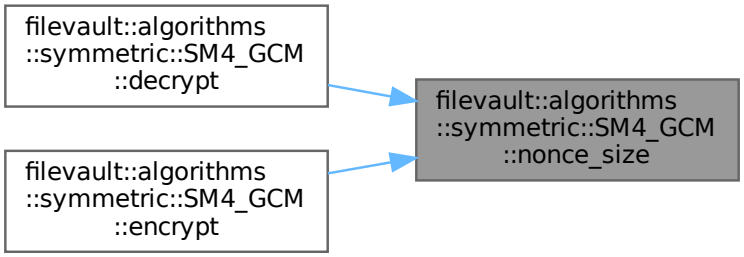
```
std::string filevault::algorithms::symmetric::SM4_GCM::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.79.3.6 nonce_size()

```
size_t filevault::algorithms::symmetric::SM4_GCM::nonce_size () const  [inline]
```

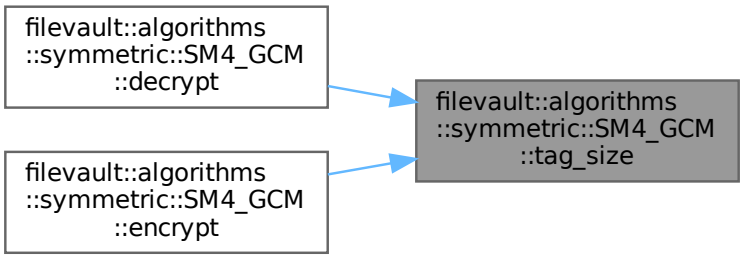
Here is the caller graph for this function:



9.79.3.7 tag_size()

```
size_t filevault::algorithms::symmetric::SM4_GCM::tag_size () const  [inline]
```

Here is the caller graph for this function:



9.79.3.8 type()

```
core::AlgorithmType filevault::algorithms::symmetric::SM4_GCM::type () const  [override],
[virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.
The documentation for this class was generated from the following files:

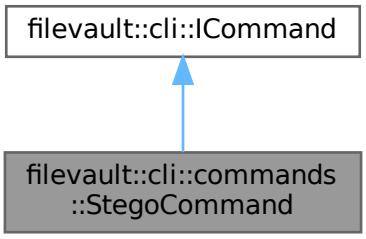
- include/filevault/algorithms/symmetric/sm4_gcm.hpp
- src/algorithms/symmetric/sm4_gcm.cpp

9.80 filevault::cli::commands::StegoCommand Class Reference

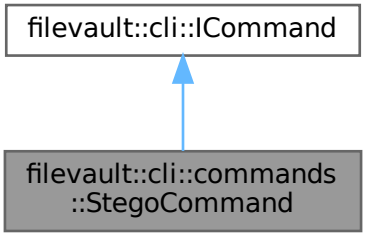
Steganography command for embedding/extracting data in images.

`#include <stego_cmd.hpp>`

Inheritance diagram for filevault::cli::commands::StegoCommand:



Collaboration diagram for filevault::cli::commands::StegoCommand:



Public Member Functions

- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Member Functions

- int do_embed ()

- int do_extract ()
- int do_capacity ()

Private Attributes

- std::string operation_
- std::string input_file_
- std::string cover_image_
- std::string output_file_
- int bits_per_channel_ = 1
- bool verbose_ = false

9.80.1 Detailed Description

Steganography command for embedding/extracting data in images.
Supports LSB (Least Significant Bit) steganography for hiding data within image files without visible changes.
Features:

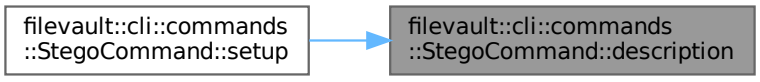
- Embed secret data into PNG/BMP images
- Extract hidden data from stego images
- Calculate embedding capacity
- Configurable bits per channel (1-4)

9.80.2 Member Function Documentation

9.80.2.1 description()

```
std::string filevault::cli::commands::StegoCommand::description () const  [inline], [override], [virtual]
```

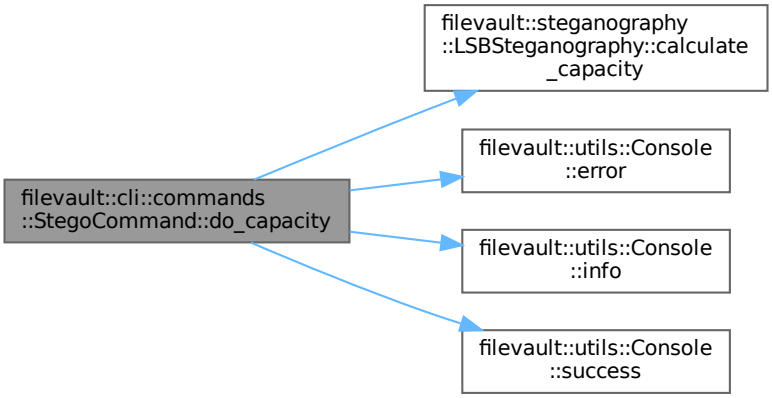
Get command description.
Implements filevault::cli::ICommand.
Here is the caller graph for this function:



9.80.2.2 do_capacity()

```
int filevault::cli::commands::StegoCommand::do_capacity ()  [private]
```

Here is the call graph for this function:

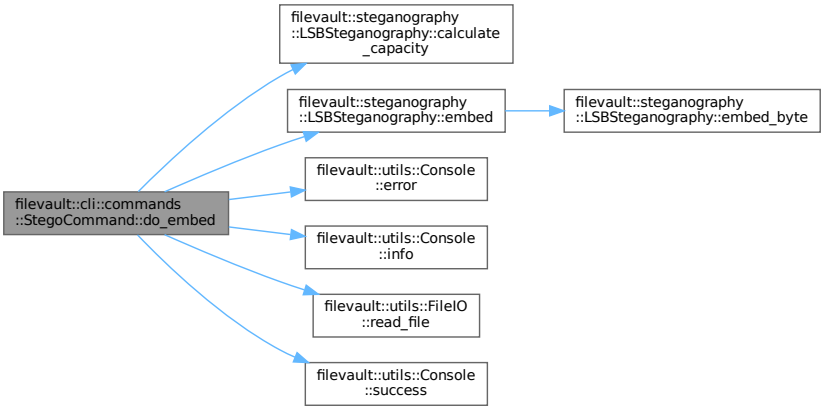


Here is the caller graph for this function:

9.80.2.3 do_embed()

```
int filevault::cli::commands::StegoCommand::do_embed ()  [private]
```

Here is the call graph for this function:

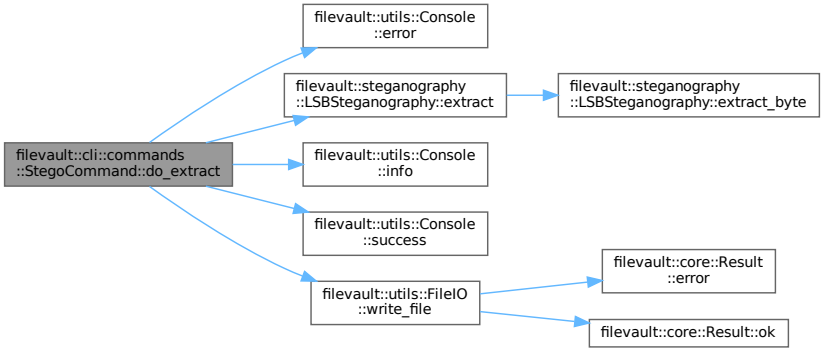


Here is the caller graph for this function:

9.80.2.4 do_extract()

`int filevault::cli::commands::StegoCommand::do_extract () [private]`

Here is the call graph for this function:



Here is the caller graph for this function:

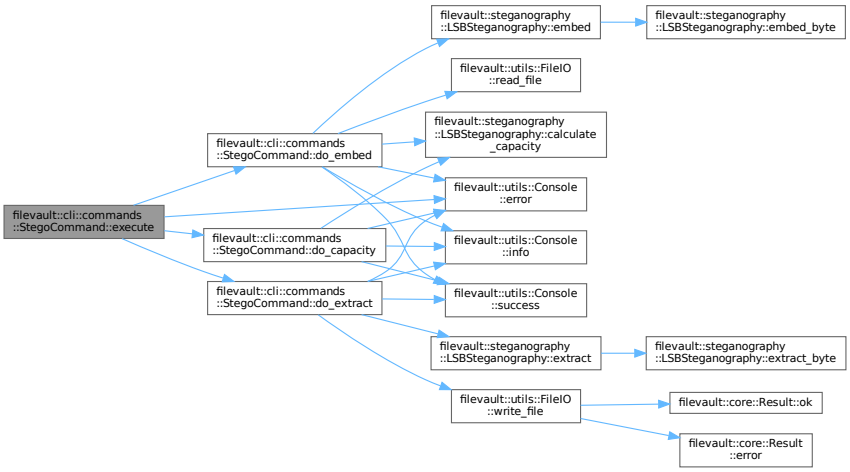
9.80.2.5 execute()

`int filevault::cli::commands::StegoCommand::execute () [override], [virtual]`

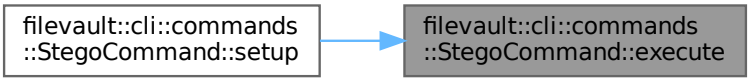
Execute the command.

Implements filevault::cli::ICommand.

Here is the call graph for this function:



Here is the caller graph for this function:



9.80.2.6 name()

```
std::string filevault::cli::commands::StegoCommand::name () const  [inline], [override], [virtual]
```

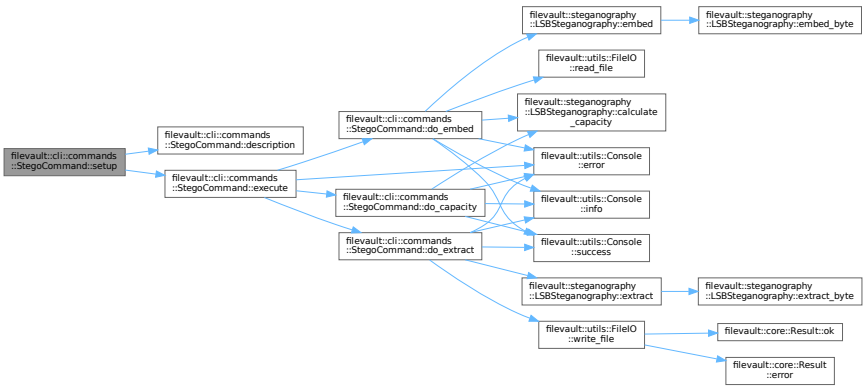
Get command name.
Implements filevault::cli::ICommand.

9.80.2.7 setup()

```
void filevault::cli::commands::StegoCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.

Here is the call graph for this function:



9.80.3 Member Data Documentation

9.80.3.1 bits_per_channel_

```
int filevault::cli::commands::StegoCommand::bits_per_channel_ = 1  [private]
```

9.80.3.2 cover_image_

```
std::string filevault::cli::commands::StegoCommand::cover_image_  [private]
```

9.80.3.3 input_file_

```
std::string filevault::cli::commands::StegoCommand::input_file_  [private]
```

9.80.3.4 operation_

```
std::string filevault::cli::commands::StegoCommand::operation_  [private]
```

9.80.3.5 output_file_

```
std::string filevault::cli::commands::StegoCommand::output_file_  [private]
```

9.80.3.6 verbose_

```
bool filevault::cli::commands::StegoCommand::verbose_ = false  [private]
```

The documentation for this class was generated from the following files:

- include/filevault/cli/commands/stego_cmd.hpp
- src/cli/commands/stego_cmd.cpp

9.81 filevault::core::StreamingConfig Struct Reference

Configuration for streaming encryption.

```
#include <streaming.hpp>
```

Public Attributes

- size_t chunk_size = 64 ∗ 1024 ∗ 1024
- AlgorithmType algorithm = AlgorithmType::AES_256_GCM
- KDFType kdf = KDFType::ARGON2ID

- SecurityLevel level = SecurityLevel::STRONG
- CompressionType compression = CompressionType::NONE
- int compression_level = 6
- StreamProgressCallback progress_callback = nullptr

9.81.1 Detailed Description

Configuration for streaming encryption.

9.81.2 Member Data Documentation

9.81.2.1 algorithm

```
AlgorithmType filevault::core::StreamingConfig::algorithm = AlgorithmType::AES_256_GCM
```

9.81.2.2 chunk_size

```
size_t filevault::core::StreamingConfig::chunk_size = 64 * 1024 * 1024
```

9.81.2.3 compression

```
CompressionType filevault::core::StreamingConfig::compression = CompressionType::NONE
```

9.81.2.4 compression_level

```
int filevault::core::StreamingConfig::compression_level = 6
```

9.81.2.5 kdf

```
KDFType filevault::core::StreamingConfig::kdf = KDFType::ARGON2ID
```

9.81.2.6 level

```
SecurityLevel filevault::core::StreamingConfig::level = SecurityLevel::STRONG
```

9.81.2.7 progress_callback

```
StreamProgressCallback filevault::core::StreamingConfig::progress_callback = nullptr
```

The documentation for this struct was generated from the following file:

- include/filevault/core/streaming.hpp

9.82 filevault::core::StreamingCrypto Class Reference

Streaming encryption/decryption for large files.

```
#include <streaming.hpp>
```

Static Public Member Functions

- static StreamingResult encrypt_file (const std::string &input_path, const std::string &output_path, const std::string &password, const StreamingConfig &config={})

  Encrypt a large file using streaming.
- static StreamingResult decrypt_file (const std::string &input_path, const std::string &output_path, const std::string &password, StreamProgressCallback progress_callback=nullptr)

  Decrypt a large file using streaming.
- static bool should_use_streaming (const std::string &file_path, size_t threshold=100 ∗1024 ∗1024)

  Check if a file should use streaming (based on size).
- static size_t get_recommended_chunk_size ()

  Get recommended chunk size based on available memory.

Static Private Member Functions

- static std::vector< uint8_t > derive_chunk_nonce (const std::vector< uint8_t > &base_nonce, size_t chunk_index)

  Derive chunk-specific nonce from base nonce and chunk index.
- static bool write_stream_header (std::ofstream &file, const StreamingConfig &config, const std::vector< uint8_t > &salt, const std::vector< uint8_t > &base_nonce, size_t total_size, size_t chunk_count)

  Write streaming file header.
- static bool read_stream_header (std::ifstream &file, StreamingConfig &config, std::vector< uint8_t > &salt, std::vector< uint8_t > &base_nonce, size_t &original_size, size_t &chunk_count)

  Read streaming file header.

9.82.1 Detailed Description

Streaming encryption/decryption for large files.
Uses chunk-based processing to handle files larger than available RAM. Each chunk is encrypted independently with a unique nonce derived from the base nonce and chunk index.
File format for streaming: [Header][Chunk1][Chunk2]...[ChunkN][Footer]
Each chunk: [4 bytes: chunk_size][12 bytes: nonce][encrypted_data][16 bytes: tag]

9.82.2 Member Function Documentation

9.82.2.1 decrypt_file()

```
StreamingResult filevault::core::StreamingCrypto::decrypt_file (
            const std::string & input_path,
            const std::string & output_path,
            const std::string & password,
            StreamProgressCallback progress_callback = nullptr)  [static]
```

Decrypt a large file using streaming.

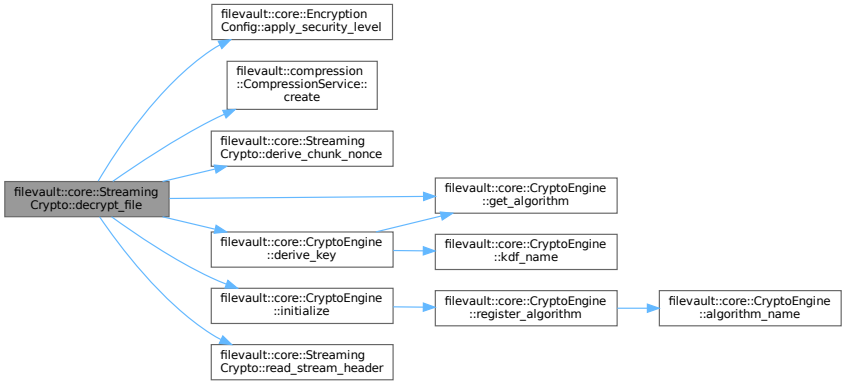
Parameters

input_path	Path to encrypted file
output_path	Path to output file
password	Decryption password
progress_callback	Optional progress callback

Returns

Result of the operation

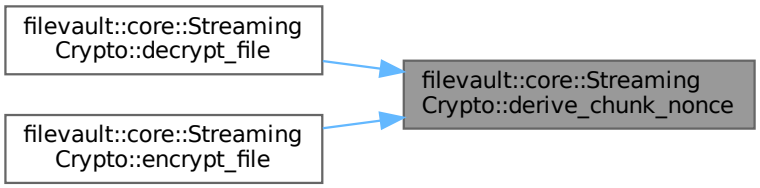
Here is the call graph for this function:



9.82.2.2 derive_chunk_nonce()

```
std::vector< uint8_t > filevault::core::StreamingCrypto::derive_chunk_nonce (
            const std::vector< uint8_t > & base_nonce,
            size_t chunk_index)  [static], [private]
```

Derive chunk-specific nonce from base nonce and chunk index.
Here is the caller graph for this function:



9.82.2.3 encrypt_file()

```
StreamingResult filevault::core::StreamingCrypto::encrypt_file (
            const std::string & input_path,
            const std::string & output_path,
            const std::string & password,
            const StreamingConfig & config = {})  [static]
```

Encrypt a large file using streaming.

Parameters

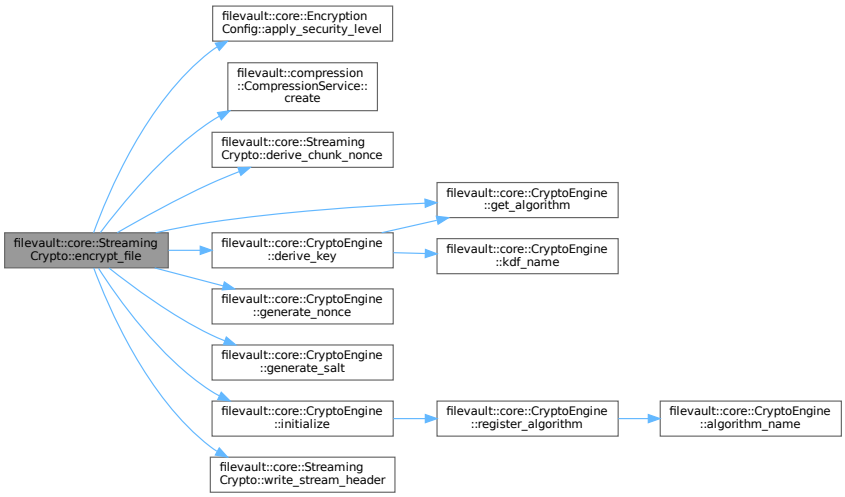
input_path	Path to input file

output_path	Path to output file
password	Encryption password
config	Streaming configuration

Returns

Result of the operation

Here is the call graph for this function:



9.82.2.4 get_recommended_chunk_size()

```
size_t filevault::core::StreamingCrypto::get_recommended_chunk_size ()  [static]
```

Get recommended chunk size based on available memory.

Returns

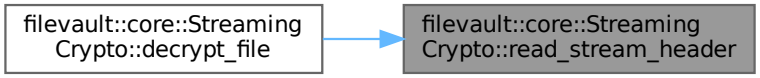
Recommended chunk size in bytes

9.82.2.5 read_stream_header()

```
bool filevault::core::StreamingCrypto::read_stream_header (
            std::ifstream & file,
            StreamingConfig & config,
            std::vector< uint8_t > & salt,
            std::vector< uint8_t > & base_nonce,
            size_t & original_size,
            size_t & chunk_count)  [static], [private]
```

Read streaming file header.

Here is the caller graph for this function:



9.82.2.6 should_use_streaming()

```
bool filevault::core::StreamingCrypto::should_use_streaming (
            const std::string & file_path,
            size_t threshold = 100 * 1024 * 1024)  [static]
```

Check if a file should use streaming (based on size).

Parameters

file_path	Path to file
threshold	Size threshold (default 100MB)

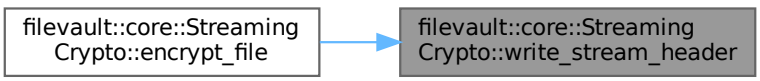
Returns

true if file is larger than threshold

9.82.2.7 write_stream_header()

```
bool filevault::core::StreamingCrypto::write_stream_header (
            std::ofstream & file,
            const StreamingConfig & config,
            const std::vector< uint8_t > & salt,
            const std::vector< uint8_t > & base_nonce,
            size_t total_size,
            size_t chunk_count)  [static], [private]
```

Write streaming file header.
Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/core/streaming.hpp
- src/core/streaming.cpp

9.83 filevault::core::StreamingResult Struct Reference

Result of streaming operation.
`#include <streaming.hpp>`

Public Attributes

- bool success = false
- std::string error_message
- size_t bytes_processed = 0
- size_t chunks_processed = 0
- double processing_time_ms = 0.0
- double throughput_mbps = 0.0

9.83.1 Detailed Description

Result of streaming operation.

9.83.2 Member Data Documentation

9.83.2.1 bytes_processed

```
size_t filevault::core::StreamingResult::bytes_processed = 0
```

9.83.2.2 chunks_processed

```
size_t filevault::core::StreamingResult::chunks_processed = 0
```

9.83.2.3 error_message

```
std::string filevault::core::StreamingResult::error_message
```

9.83.2.4 processing_time_ms

```
double filevault::core::StreamingResult::processing_time_ms = 0.0
```

9.83.2.5 success

```
bool filevault::core::StreamingResult::success = false
```

9.83.2.6 throughput_mbps

```
double filevault::core::StreamingResult::throughput_mbps = 0.0
```

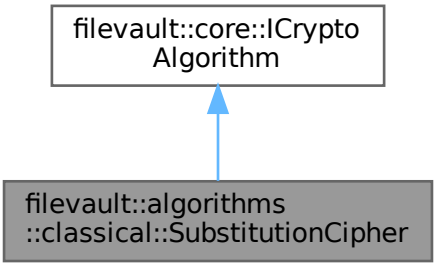
The documentation for this struct was generated from the following file:

- include/filevault/core/streaming.hpp

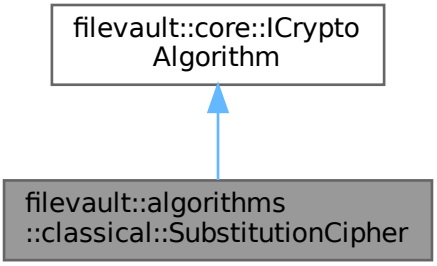
9.84 filevault::algorithms::classical::SubstitutionCipher Class Reference

Simple substitution cipher.
`#include <substitution.hpp>`

Inheritance diagram for filevault::algorithms::classical::SubstitutionCipher:



Collaboration diagram for filevault::algorithms::classical::SubstitutionCipher:



Public Member Functions

- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Types

- using SubstitutionMap = std::array<char, 26>

Private Member Functions

- SubstitutionMap parse_key (std::span< const uint8_t > key)
- SubstitutionMap create_reverse_map (const SubstitutionMap &forward_map)
- bool is_valid_key (const SubstitutionMap &map)
- std::string apply_substitution (const std::string &text, const SubstitutionMap &map)

9.84.1 Detailed Description

Simple substitution cipher.
Educational cipher using a 26-letter substitution alphabet Key: 26-character permutation of the alphabet

9.84.2 Member Typedef Documentation

9.84.2.1 SubstitutionMap

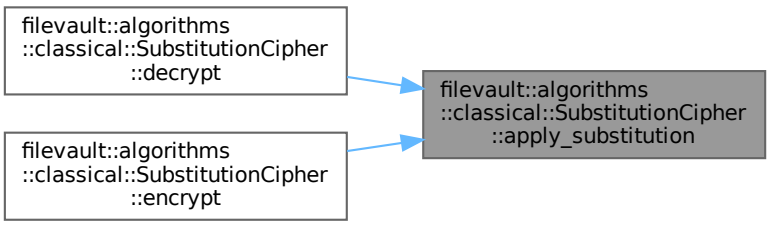
```
using filevault::algorithms::classical::SubstitutionCipher::SubstitutionMap = std::array<char,
26>  [private]
```

9.84.3 Member Function Documentation

9.84.3.1 apply_substitution()

```
std::string filevault::algorithms::classical::SubstitutionCipher::apply_substitution (
            const std::string & text,
            const SubstitutionMap & map)  [private]
```

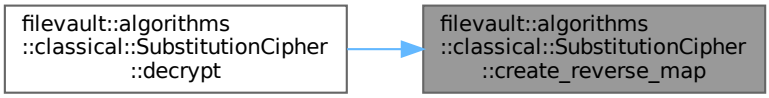
Here is the caller graph for this function:



9.84.3.2 create_reverse_map()

```
SubstitutionCipher::SubstitutionMap filevault::algorithms::classical::SubstitutionCipher↩
::create_reverse_map (
            const SubstitutionMap & forward_map)  [private]
```

Here is the caller graph for this function:



9.84.3.3 decrypt()

```
core::CryptoResult filevault::algorithms::classical::SubstitutionCipher::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

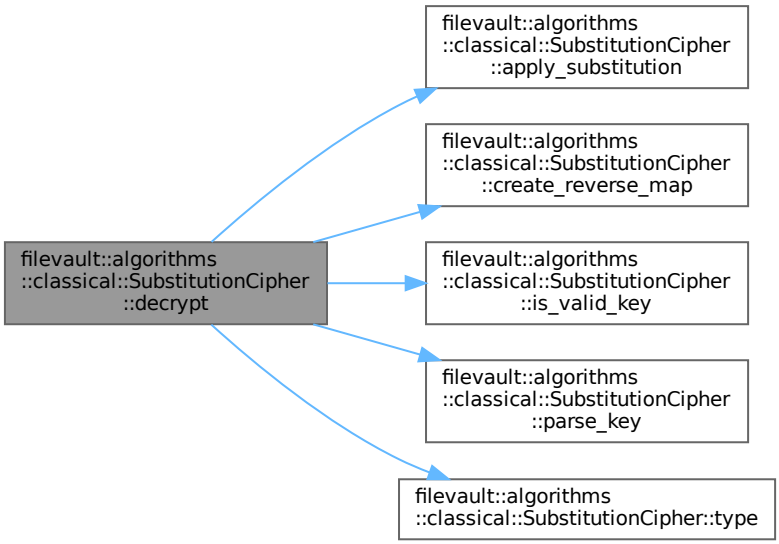
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.84.3.4 encrypt()

```
core::CryptoResult filevault::algorithms::classical::SubstitutionCipher::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

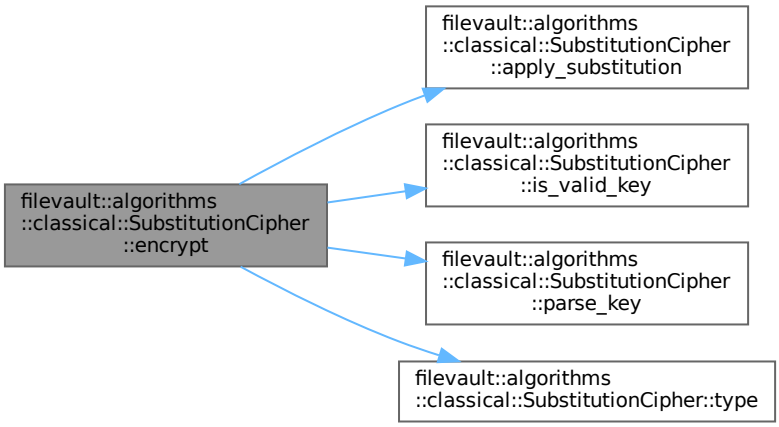
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.84.3.5 is_suitable_for()

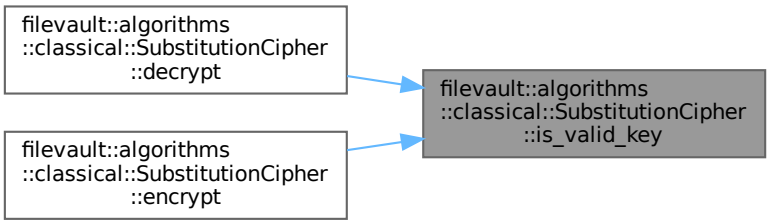
```
bool filevault::algorithms::classical::SubstitutionCipher::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.84.3.6 is_valid_key()

```
bool filevault::algorithms::classical::SubstitutionCipher::is_valid_key (
            const SubstitutionMap & map)  [private]
```

Here is the caller graph for this function:



9.84.3.7 key_size()

```
size_t filevault::algorithms::classical::SubstitutionCipher::key_size () const  [inline],
[override], [virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.84.3.8 name()

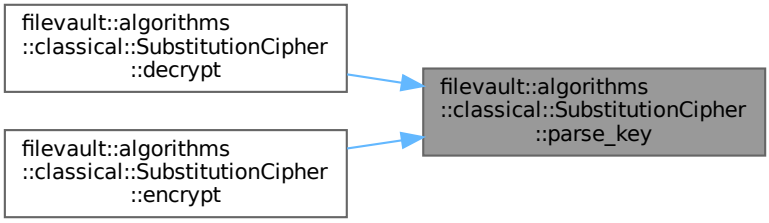
```
std::string filevault::algorithms::classical::SubstitutionCipher::name () const  [inline],
[override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.84.3.9 parse_key()

```
SubstitutionCipher::SubstitutionMap filevault::algorithms::classical::SubstitutionCipher↩
::parse_key (
            std::span< const uint8_t > key)  [private]
```

Here is the caller graph for this function:

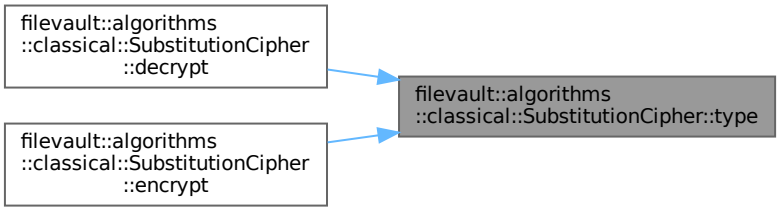


9.84.3.10 type()

```
core::AlgorithmType filevault::algorithms::classical::SubstitutionCipher::type () const  [inline],
[override], [virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/algorithms/classical/substitution.hpp
- src/algorithms/classical/substitution.cpp

9.85 filevault::utils::TableBuilder Class Reference

Quick table builders.

`#include <table_formatter.hpp>`

Static Public Member Functions

- static void print_algorithm_list (const std::vector< std::tuple< std::string, std::string, std::string > > &algorithms)
- static void print_benchmark_results (const std::vector< std::tuple< std::string, double, double > > &results)
- static void print_file_summary (const std::string &input_file, const std::string &output_file, size_t input_size, size_t output_size, double processing_time_ms)
- static void print_encryption_config (const std::string &algorithm, const std::string &kdf, const std::string &security_level, size_t kdf_iterations, size_t kdf_memory_kb)

9.85.1 Detailed Description

Quick table builders.

9.85.2 Member Function Documentation

9.85.2.1 print_algorithm_list()

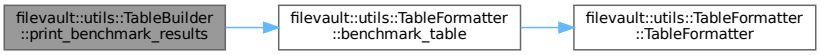
```
void filevault::utils::TableBuilder::print_algorithm_list (
            const std::vector< std::tuple< std::string, std::string, std::string > > & algorithms)
[static]
```

Here is the call graph for this function:

9.85.2.2 print_benchmark_results()

```
void filevault::utils::TableBuilder::print_benchmark_results (
            const std::vector< std::tuple< std::string, double, double > > & results)  [static]
```

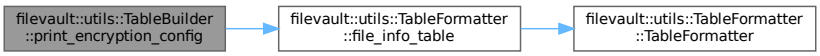
Here is the call graph for this function:



9.85.2.3 print_encryption_config()

```
void filevault::utils::TableBuilder::print_encryption_config (
            const std::string & algorithm,
            const std::string & kdf,
            const std::string & security_level,
            size_t kdf_iterations,
            size_t kdf_memory_kb)  [static]
```

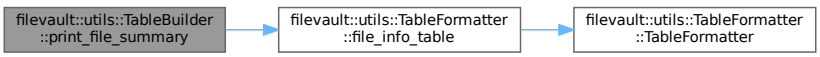
Here is the call graph for this function:



9.85.2.4 print_file_summary()

```
void filevault::utils::TableBuilder::print_file_summary (
            const std::string & input_file,
            const std::string & output_file,
            size_t input_size,
            size_t output_size,
            double processing_time_ms)  [static]
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- include/filevault/utils/table_formatter.hpp
- src/utils/table_formatter.cpp

9.86 filevault::utils::TableFormatter Class Reference

Beautiful table formatter.

`#include <table_formatter.hpp>`

Public Member Functions

- TableFormatter (const std::vector< std::string > &headers)
- void add_row (const std::vector< std::string > &row)
- void print ()
- std::string to_string ()
- void set_border_style (const std::string &style)
- void set_column_alignment (size_t col_index, tabulate::FontAlign align)
- void set_column_format (size_t col_index, tabulate::Color color)

Static Public Member Functions

- static TableFormatter algorithm_list_table ()
- static TableFormatter benchmark_table ()
- static TableFormatter file_info_table ()

Private Attributes

- tabulate::Table table_

9.86.1 Detailed Description

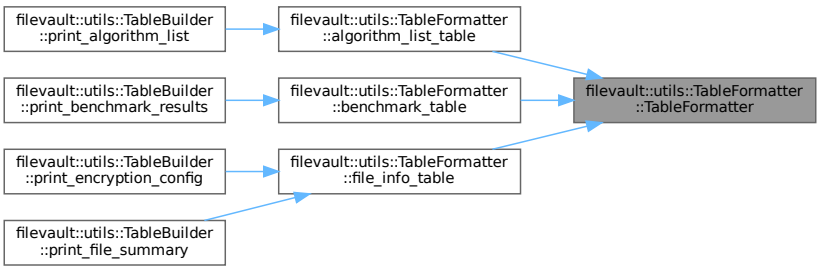
Beautiful table formatter.

9.86.2 Constructor & Destructor Documentation

9.86.2.1 TableFormatter()

```
filevault::utils::TableFormatter::TableFormatter (
            const std::vector< std::string > & headers)
```

Here is the caller graph for this function:



9.86.3 Member Function Documentation

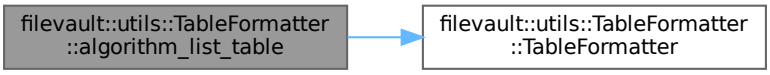
9.86.3.1 add_row()

```
void filevault::utils::TableFormatter::add_row (
            const std::vector< std::string > & row)
```

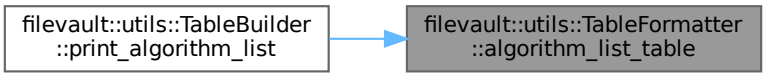
9.86.3.2 algorithm_list_table()

```
TableFormatter filevault::utils::TableFormatter::algorithm_list_table ()  [static]
```

Here is the call graph for this function:



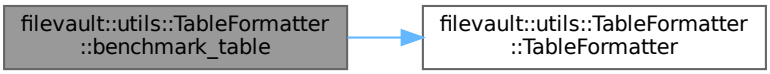
Here is the caller graph for this function:



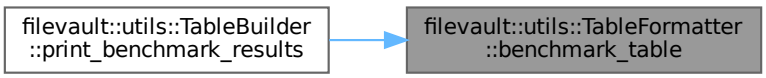
9.86.3.3 benchmark_table()

`TableFormatter filevault::utils::TableFormatter::benchmark_table () [static]`

Here is the call graph for this function:



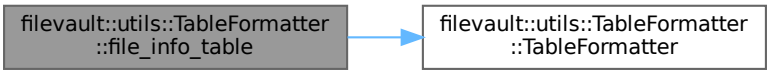
Here is the caller graph for this function:



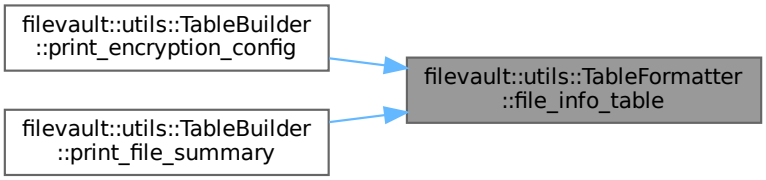
9.86.3.4 file_info_table()

`TableFormatter filevault::utils::TableFormatter::file_info_table () [static]`

Here is the call graph for this function:



Here is the caller graph for this function:



9.86.3.5 print()

```
void filevault::utils::TableFormatter::print ()
```

9.86.3.6 set_border_style()

```
void filevault::utils::TableFormatter::set_border_style (
            const std::string & style)
```

9.86.3.7 set_column_alignment()

```
void filevault::utils::TableFormatter::set_column_alignment (
            size_t col_index,
            tabulate::FontAlign align)
```

9.86.3.8 set_column_format()

```
void filevault::utils::TableFormatter::set_column_format (
            size_t col_index,
            tabulate::Color color)
```

9.86.3.9 to_string()

```
std::string filevault::utils::TableFormatter::to_string ()
```

9.86.4 Member Data Documentation

9.86.4.1 table_

```
tabulate::Table filevault::utils::TableFormatter::table_  [private]
```

The documentation for this class was generated from the following files:

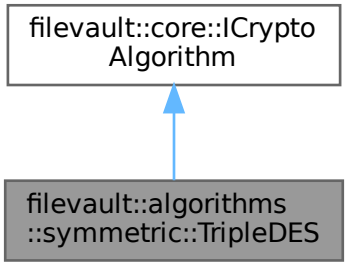
- include/filevault/utils/table_formatter.hpp
- src/utils/table_formatter.cpp

9.87 filevault::algorithms::symmetric::TripleDES Class Reference

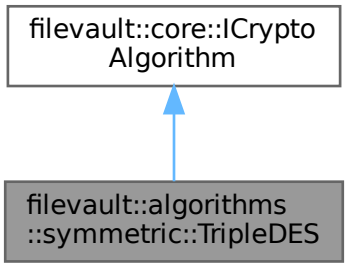
Triple-DES (3DES/TDEA) encryption.

`#include <triple_des.hpp>`

Inheritance diagram for filevault::algorithms::symmetric::TripleDES:



Collaboration diagram for filevault::algorithms::symmetric::TripleDES:



Public Member Functions

- TripleDES ()
- virtual ~TripleDES ()=default
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- size_t key_size () const override

  Get recommended key size in bytes.

- size_t iv_size () const
- size_t block_size () const
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

9.87.1 Detailed Description

Triple-DES (3DES/TDEA) encryption.
Uses three DES operations: Encrypt-Decrypt-Encrypt (EDE) 168-bit key (112 bits effective security) 64-bit block size

Warning

Legacy algorithm! Prefer AES for new applications. Block size of 64 bits is vulnerable to birthday attacks on large datasets (Sweet32 attack).

9.87.2 Constructor & Destructor Documentation

9.87.2.1 TripleDES()

```
filevault::algorithms::symmetric::TripleDES::TripleDES ()  [explicit]
```

9.87.2.2 ~TripleDES()

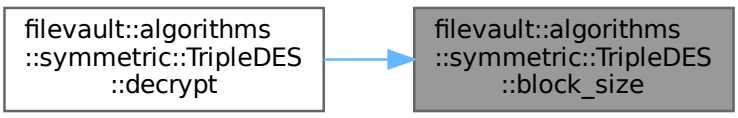
```
virtual filevault::algorithms::symmetric::TripleDES::~TripleDES ()  [virtual], [default]
```

9.87.3 Member Function Documentation

9.87.3.1 block_size()

```
size_t filevault::algorithms::symmetric::TripleDES::block_size () const  [inline]
```

Here is the caller graph for this function:



9.87.3.2 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::TripleDES::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

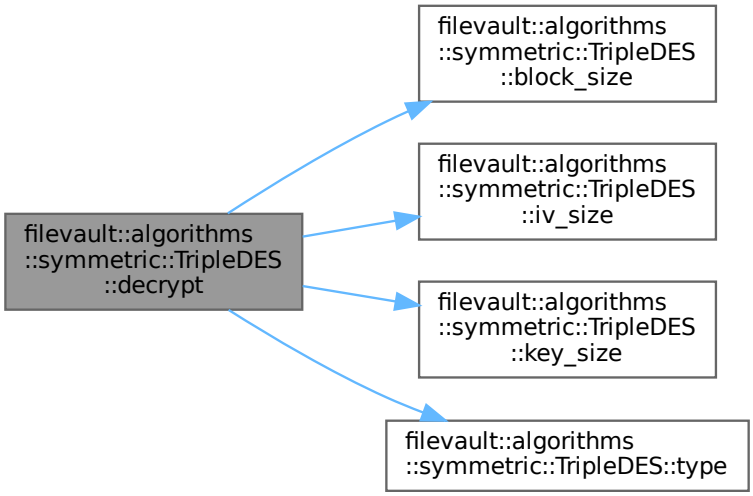
Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.87.3.3 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::TripleDES::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

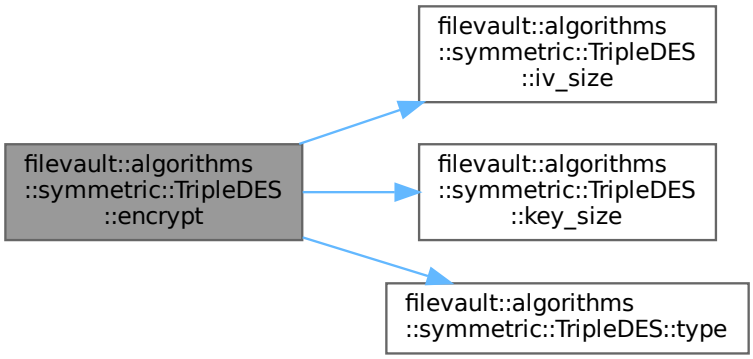
Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.87.3.4 is_suitable_for()

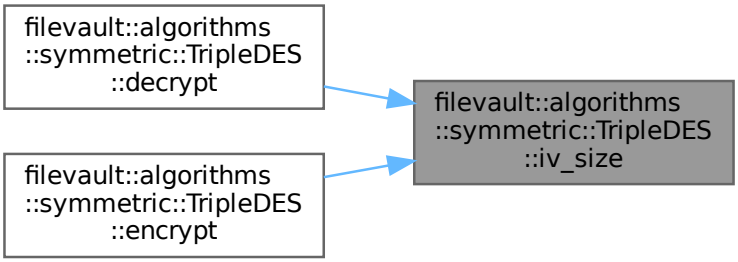
```
bool filevault::algorithms::symmetric::TripleDES::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.87.3.5 iv_size()

```
size_t filevault::algorithms::symmetric::TripleDES::iv_size () const  [inline]
```

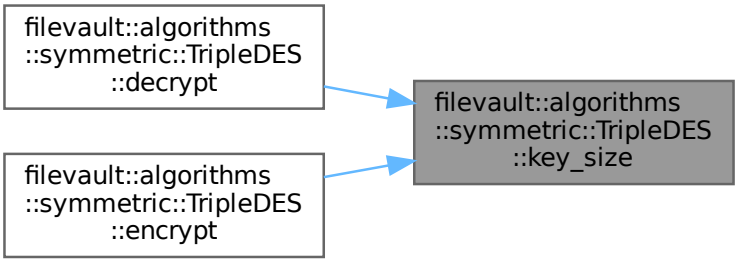
Here is the caller graph for this function:



9.87.3.6 key_size()

```
size_t filevault::algorithms::symmetric::TripleDES::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.87.3.7 name()

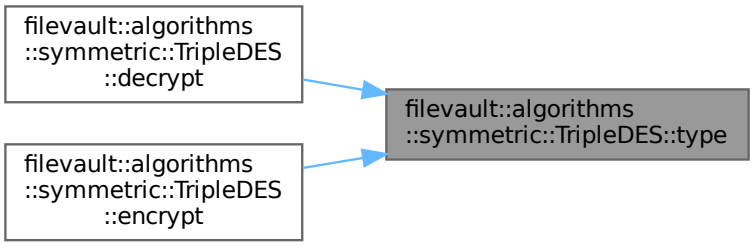
```
std::string filevault::algorithms::symmetric::TripleDES::name () const  [inline], [override],
[virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.87.3.8 type()

```
core::AlgorithmType filevault::algorithms::symmetric::TripleDES::type () const  [inline],
[override], [virtual]
```

Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



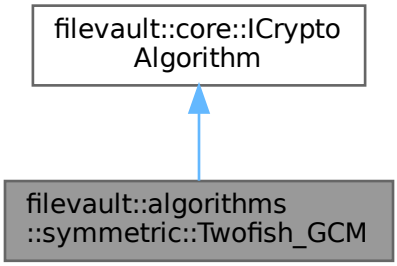
The documentation for this class was generated from the following files:

- include/filevault/algorithms/symmetric/triple_des.hpp
- src/algorithms/symmetric/triple_des.cpp

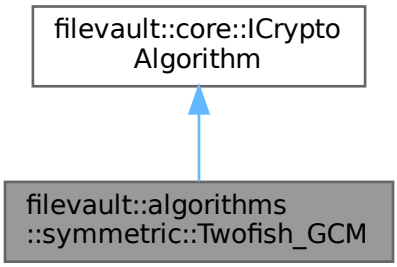
9.88 filevault::algorithms::symmetric::Twofish_GCM Class Reference

Twofish-256-GCM implementation.

`#include <twofish_gcm.hpp>`
Inheritance diagram for filevault::algorithms::symmetric::Twofish_GCM:



Collaboration diagram for filevault::algorithms::symmetric::Twofish_GCM:



Public Member Functions

- Twofish_GCM (size_t key_bits=256)

  Construct Twofish-256-GCM cipher.
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt plaintext with Twofish-GCM.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt ciphertext with Twofish-GCM.
- size_t key_size () const override

  Get required key size in bytes.
- size_t nonce_size () const

  Get nonce size in bytes (12 for GCM).

- size_t tag_size () const

  Get tag size in bytes (16 for GCM).
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Private Member Functions

- std::vector< uint8_t > process_gcm (std::span< const uint8_t > input, std::span< const uint8_t > key, std::span< const uint8_t > nonce, std::span< const uint8_t > tag, bool encrypt)

  Perform encryption/decryption operation.

Private Attributes

- size_t key_bits_
- core::AlgorithmType type_
- std::string botan_name_

9.88.1 Detailed Description

Twofish-256-GCM implementation.
Twofish is a 128-bit block cipher designed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. It was an AES finalist and is known for its conservative design.
References:

- Twofish: A 128-Bit Block Cipher https://www.schneier.com/academic/twofish/
- NIST SP 800-38D: GCM Mode https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublicat pdf

Security:

- Key size: 256 bits (32 bytes)
- Block size: 128 bits (16 bytes)
- Nonce size: 96 bits (12 bytes) for GCM
- Tag size: 128 bits (16 bytes) for authentication

Features:

- Pre-computed S-boxes for performance
- Key-dependent S-boxes for security
- Supports 128, 192, and 256-bit keys
- Patent-free and royalty-free

Note

This implementation uses Botan's Twofish/GCM mode

Twofish-256 provides equivalent security to AES-256

9.88.2 Constructor & Destructor Documentation

9.88.2.1 Twofish_GCM()

filevault::algorithms::symmetric::Twofish_GCM::Twofish_GCM (
size_t *key_bits* = 256) [explicit]

Construct Twofish-256-GCM cipher.

Parameters

key_bits	Key size in bits (128, 192, or 256, default 256)

9.88.3 Member Function Documentation

9.88.3.1 decrypt()

```
core::CryptoResult filevault::algorithms::symmetric::Twofish_GCM::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt ciphertext with Twofish-GCM.

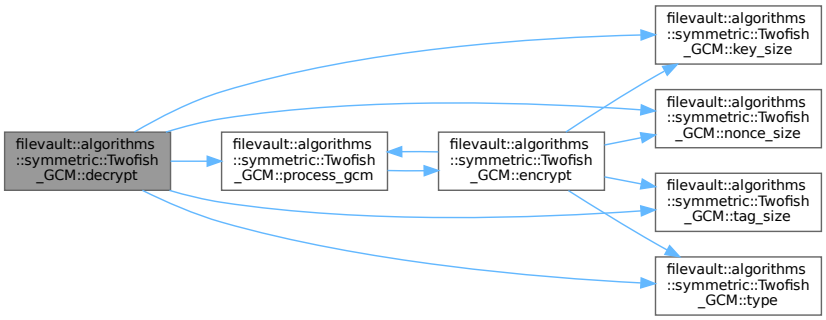
Parameters

ciphertext	Encrypted data
key	Decryption key (size depends on key_bits)
config	Decryption configuration with nonce and tag

Returns

Decrypted plaintext

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



9.88.3.2 encrypt()

```
core::CryptoResult filevault::algorithms::symmetric::Twofish_GCM::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt plaintext with Twofish-GCM.

Parameters

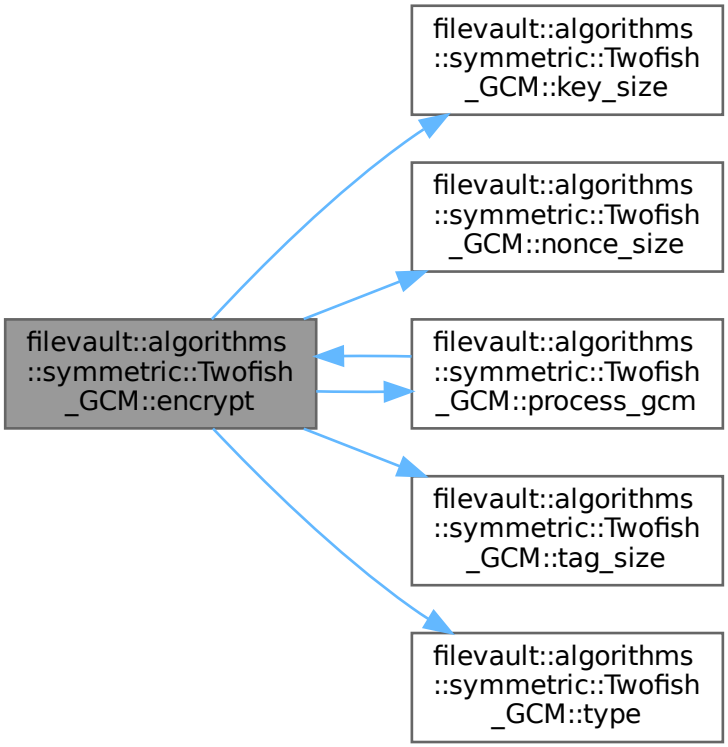
plaintext	Input data to encrypt

key	Encryption key (size depends on key_bits)
config	Encryption configuration with nonce

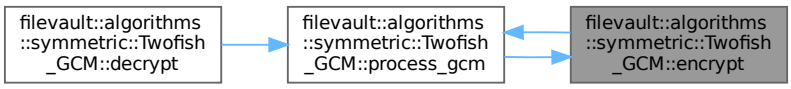
Returns

Encrypted data with authentication tag

Implements filevault::core::ICryptoAlgorithm.
Here is the call graph for this function:



Here is the caller graph for this function:



9.88.3.3 is_suitable_for()

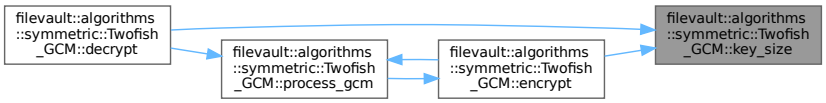
```
bool filevault::algorithms::symmetric::Twofish_GCM::is_suitable_for (
            core::SecurityLevel level) const  [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.88.3.4 key_size()

```
size_t filevault::algorithms::symmetric::Twofish_GCM::key_size () const  [override], [virtual]
```

Get required key size in bytes.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.88.3.5 name()

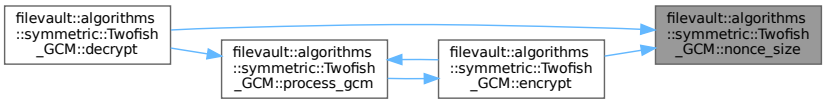
```
std::string filevault::algorithms::symmetric::Twofish_GCM::name () const  [override], [virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.88.3.6 nonce_size()

```
size_t filevault::algorithms::symmetric::Twofish_GCM::nonce_size () const  [inline]
```

Get nonce size in bytes (12 for GCM).
Here is the caller graph for this function:

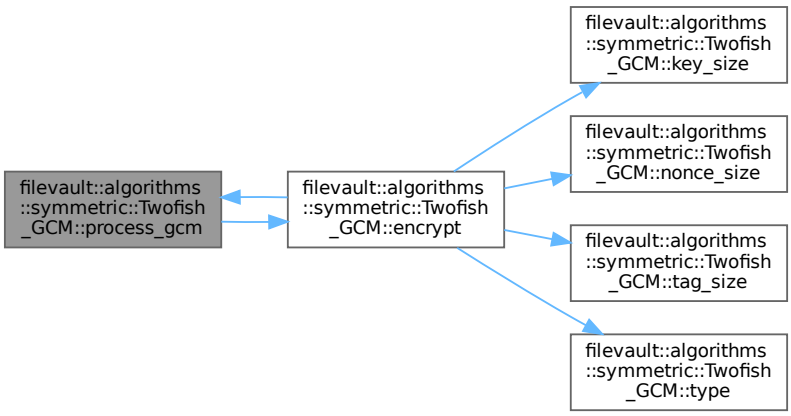


9.88.3.7 process_gcm()

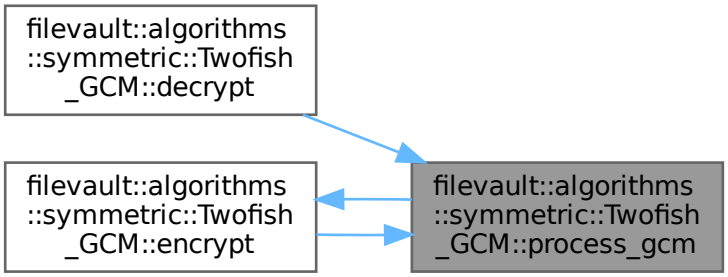
```
std::vector< uint8_t > filevault::algorithms::symmetric::Twofish_GCM::process_gcm (
            std::span< const uint8_t > input,
            std::span< const uint8_t > key,
            std::span< const uint8_t > nonce,
            std::span< const uint8_t > tag,
            bool encrypt)  [private]
```

Perform encryption/decryption operation.

Here is the call graph for this function:



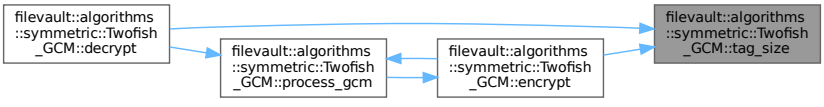
Here is the caller graph for this function:



9.88.3.8 tag_size()

```
size_t filevault::algorithms::symmetric::Twofish_GCM::tag_size () const  [inline]
```

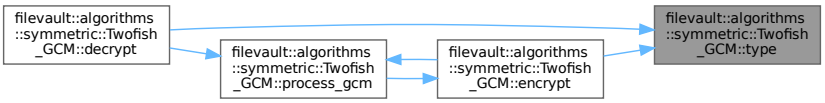
Get tag size in bytes (16 for GCM).
Here is the caller graph for this function:



9.88.3.9 type()

```
core::AlgorithmType filevault::algorithms::symmetric::Twofish_GCM::type () const  [override],
```

[virtual]
Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.
Here is the caller graph for this function:



9.88.4 Member Data Documentation

9.88.4.1 botan_name_

```
std::string filevault::algorithms::symmetric::Twofish_GCM::botan_name_  [private]
```

9.88.4.2 key_bits_

```
size_t filevault::algorithms::symmetric::Twofish_GCM::key_bits_  [private]
```

9.88.4.3 type_

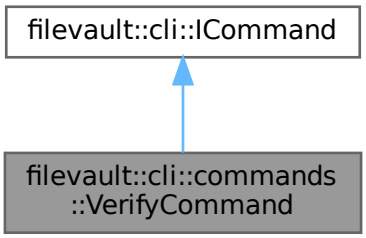
```
core::AlgorithmType filevault::algorithms::symmetric::Twofish_GCM::type_  [private]
```

The documentation for this class was generated from the following files:

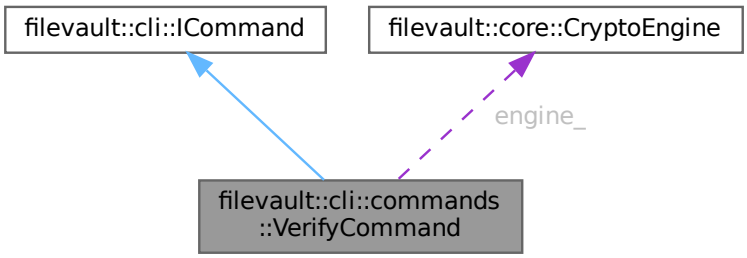
- include/filevault/algorithms/symmetric/twofish_gcm.hpp
- src/algorithms/symmetric/twofish_gcm.cpp

9.89 filevault::cli::commands::VerifyCommand Class Reference

Command to verify digital signature of a file.
`#include <verify_cmd.hpp>`
Inheritance diagram for filevault::cli::commands::VerifyCommand:



Collaboration diagram for filevault::cli::commands::VerifyCommand:



Public Member Functions

- VerifyCommand (core::CryptoEngine &engine)
- std::string name () const override

  Get command name.
- std::string description () const override

  Get command description.
- void setup (CLI::App &app) override

  Setup CLI11 subcommand.
- int execute () override

  Execute the command.

Public Member Functions inherited from filevault::cli::ICommand

- virtual ~ICommand ()=default

Private Attributes

- std::string name_ = "verify"
- std::string description_ = "Verify digital signature of a file"
- core::CryptoEngine & engine_
- std::string file_path_
- std::string signature_path_
- std::string public_key_path_
- std::string algorithm_ = "rsa"

9.89.1 Detailed Description

Command to verify digital signature of a file.

9.89.2 Constructor & Destructor Documentation

9.89.2.1 VerifyCommand()

```
filevault::cli::commands::VerifyCommand::VerifyCommand (
            core::CryptoEngine & engine)  [explicit]
```

9.89.3 Member Function Documentation

9.89.3.1 description()

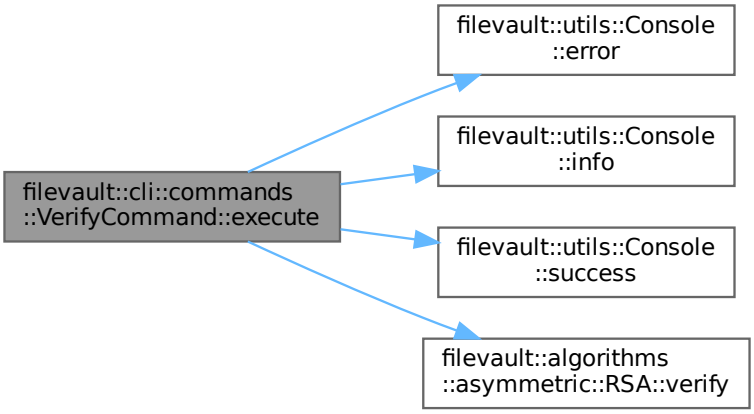
```
std::string filevault::cli::commands::VerifyCommand::description () const  [inline], [override],
[virtual]
```

Get command description.
Implements filevault::cli::ICommand.

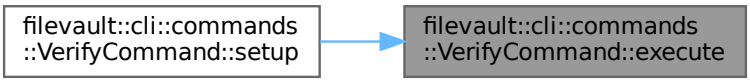
9.89.3.2 execute()

```
int filevault::cli::commands::VerifyCommand::execute ()  [override], [virtual]
```

Execute the command.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



Here is the caller graph for this function:



9.89.3.3 name()

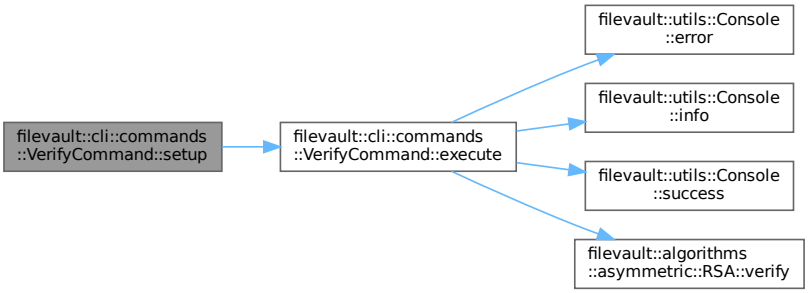
```
std::string filevault::cli::commands::VerifyCommand::name () const  [inline], [override],
[virtual]
```

Get command name.
Implements filevault::cli::ICommand.

9.89.3.4 setup()

```
void filevault::cli::commands::VerifyCommand::setup (
            CLI::App & app)  [override], [virtual]
```

Setup CLI11 subcommand.
Implements filevault::cli::ICommand.
Here is the call graph for this function:



9.89.4 Member Data Documentation

9.89.4.1 algorithm_

```
std::string filevault::cli::commands::VerifyCommand::algorithm_ = "rsa"  [private]
```

9.89.4.2 description_

```
std::string filevault::cli::commands::VerifyCommand::description_ = "Verify digital signature
of a file"  [private]
```

9.89.4.3 engine_

```
core::CryptoEngine& filevault::cli::commands::VerifyCommand::engine_  [private]
```

9.89.4.4 file_path_

```
std::string filevault::cli::commands::VerifyCommand::file_path_  [private]
```

9.89.4.5 name_

```
std::string filevault::cli::commands::VerifyCommand::name_ = "verify"  [private]
```

9.89.4.6 public_key_path_

```
std::string filevault::cli::commands::VerifyCommand::public_key_path_  [private]
```

9.89.4.7 signature_path_

```
std::string filevault::cli::commands::VerifyCommand::signature_path_  [private]
```

The documentation for this class was generated from the following files:

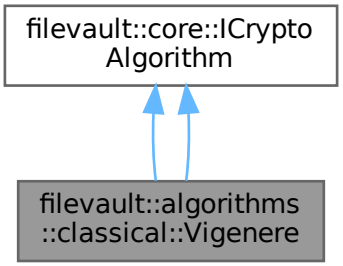
- include/filevault/cli/commands/verify_cmd.hpp
- src/cli/commands/verify_cmd.cpp

9.90 filevault::algorithms::classical::Vigenere Class Reference

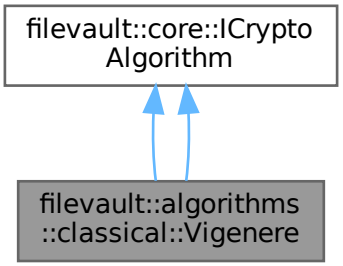
Vigenre Cipher - Polyalphabetic substitution.

```
#include <classical_ciphers.hpp>
```

Inheritance diagram for filevault::algorithms::classical::Vigenere:



Collaboration diagram for filevault::algorithms::classical::Vigenere:



Public Member Functions

- Vigenere (const std::string &keyword="KEY")
- std::string name () const override

  Get algorithm name.
- core::AlgorithmType type () const override

  Get algorithm type.
- size_t key_size () const override

  Get recommended key size in bytes.
- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.
- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.
- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.
- Vigenere (const std::string &keyword="KEY")
- std::string name () const override

Get algorithm name.

- core::AlgorithmType type () const override

  Get algorithm type.

- size_t key_size () const override

  Get recommended key size in bytes.

- core::CryptoResult encrypt (std::span< const uint8_t > plaintext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Encrypt data.

- core::CryptoResult decrypt (std::span< const uint8_t > ciphertext, std::span< const uint8_t > key, const core::EncryptionConfig &config) override

  Decrypt data.

- bool is_suitable_for (core::SecurityLevel level) const override

  Check if algorithm is suitable for security level.

Public Member Functions inherited from filevault::core::ICryptoAlgorithm

- virtual ~ICryptoAlgorithm ()=default

Static Public Member Functions

- static std::vector< size_t > kasiski_examination (const std::string &ciphertext)
- static size_t estimate_key_length (const std::string &ciphertext)

Private Attributes

- std::string keyword_

9.90.1 Detailed Description

Vigenre Cipher - Polyalphabetic substitution.
Uses a keyword to create multiple Caesar shifts. Considered "le chiffre indchiffrable" (unbreakable cipher) until 1863.
Security: BROKEN - Vulnerable to Kasiski examination Purpose: Educational - shows improvement over monoalphabetic ciphers
Uses a keyword to create multiple Caesar shifts. Considered "le chiffre indchiffrable" (unbreakable cipher) until 1863.
Security: BROKEN - Vulnerable to Kasiski examination Purpose: Educational - shows improvement over monoalphabetic ciphers

See also

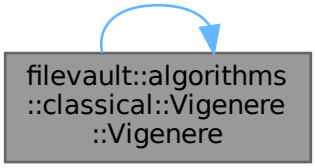
https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

9.90.2 Constructor & Destructor Documentation

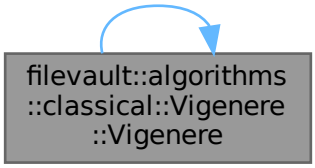
9.90.2.1 Vigenere() [1/2]

```
filevault::algorithms::classical::Vigenere::Vigenere (
            const std::string & keyword = "KEY")  [explicit]
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.90.2.2 Vigenere() [2/2]

```
filevault::algorithms::classical::Vigenere::Vigenere (
            const std::string & keyword = "KEY")  [explicit]
```

9.90.3 Member Function Documentation

9.90.3.1 decrypt() [1/2]

```
core::CryptoResult filevault::algorithms::classical::Vigenere::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

9.90.3.2 decrypt() [2/2]

```
core::CryptoResult filevault::algorithms::classical::Vigenere::decrypt (
            std::span< const uint8_t > ciphertext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Decrypt data.

Parameters

ciphertext	Encrypted data
key	Decryption key (derived from password)
config	Decryption configuration

Returns

Decrypted data

Implements filevault::core::ICryptoAlgorithm.

9.90.3.3 encrypt() [1/2]

```
core::CryptoResult filevault::algorithms::classical::Vigenere::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.

9.90.3.4 encrypt() [2/2]

```
core::CryptoResult filevault::algorithms::classical::Vigenere::encrypt (
            std::span< const uint8_t > plaintext,
            std::span< const uint8_t > key,
            const core::EncryptionConfig & config)  [override], [virtual]
```

Encrypt data.

Parameters

plaintext	Input data to encrypt
key	Encryption key (derived from password)
config	Encryption configuration

Returns

Encrypted data with metadata

Implements filevault::core::ICryptoAlgorithm.

9.90.3.5 estimate_key_length()

```
size_t filevault::algorithms::classical::Vigenere::estimate_key_length (
            const std::string & ciphertext)  [static]
```

9.90.3.6 is_suitable_for() [1/2]

```
bool filevault::algorithms::classical::Vigenere::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.90.3.7 is_suitable_for() [2/2]

```
bool filevault::algorithms::classical::Vigenere::is_suitable_for (
            core::SecurityLevel level) const  [inline], [override], [virtual]
```

Check if algorithm is suitable for security level.
Implements filevault::core::ICryptoAlgorithm.

9.90.3.8 kasiski_examination()

```
std::vector< size_t > filevault::algorithms::classical::Vigenere::kasiski_examination (
            const std::string & ciphertext)  [static]
```

9.90.3.9 key_size() [1/2]

```
size_t filevault::algorithms::classical::Vigenere::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.90.3.10 key_size() [2/2]

```
size_t filevault::algorithms::classical::Vigenere::key_size () const  [inline], [override],
[virtual]
```

Get recommended key size in bytes.
Implements filevault::core::ICryptoAlgorithm.

9.90.3.11 name() [1/2]

```
std::string filevault::algorithms::classical::Vigenere::name () const  [inline], [override],
[virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.90.3.12 name() [2/2]

```
std::string filevault::algorithms::classical::Vigenere::name () const  [inline], [override],
[virtual]
```

Get algorithm name.
Implements filevault::core::ICryptoAlgorithm.

9.90.3.13 type() [1/2]

`core::AlgorithmType filevault::algorithms::classical::Vigenere::type () const [inline], [override], [virtual]`
Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.90.3.14 type() [2/2]

`core::AlgorithmType filevault::algorithms::classical::Vigenere::type () const [inline], [override], [virtual]`
Get algorithm type.
Implements filevault::core::ICryptoAlgorithm.

9.90.4 Member Data Documentation

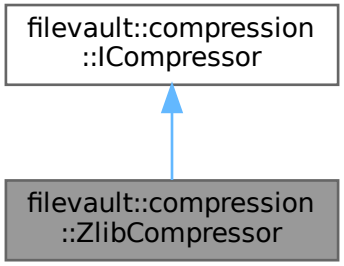
9.90.4.1 keyword_

`std::string filevault::algorithms::classical::Vigenere::keyword_ [private]`
The documentation for this class was generated from the following files:

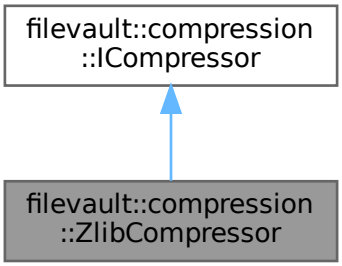
- include/filevault/algorithms/classical/classical_ciphers.hpp
- include/filevault/algorithms/classical/vigenere.hpp
- src/algorithms/classical/classical_ciphers.cpp
- src/algorithms/classical/vigenere.cpp

9.91 filevault::compression::ZlibCompressor Class Reference

ZLIB compressor (fast, good compression).
`#include <compressor.hpp>`
Inheritance diagram for filevault::compression::ZlibCompressor:



Collaboration diagram for filevault::compression::ZlibCompressor:



Public Member Functions

- std::string name () const override

  Get compressor name.
- CompressionResult compress (std::span< const uint8_t > input, int level=6) override

  Compress data.
- CompressionResult decompress (std::span< const uint8_t > input) override

  Decompress data.

Public Member Functions inherited from filevault::compression::ICompressor

- virtual ~ICompressor ()=default

9.91.1 Detailed Description

ZLIB compressor (fast, good compression).

9.91.2 Member Function Documentation

9.91.2.1 compress()

```
CompressionResult filevault::compression::ZlibCompressor::compress (
            std::span< const uint8_t > input,
            int level = 6)  [override], [virtual]
```

Compress data.

Parameters

input	Data to compress
level	Compression level (1-9, algorithm-specific)

Returns

Compressed data

Implements filevault::compression::ICompressor.

9.91.2.2 decompress()

```
CompressionResult filevault::compression::ZlibCompressor::decompress (
            std::span< const uint8_t > input)  [override], [virtual]
```

Decompress data.

Parameters

input	Compressed data

Returns

Decompressed data

Implements filevault::compression::ICompressor.

9.91.2.3 name()

```
std::string filevault::compression::ZlibCompressor::name () const  [inline], [override], [virtual]
```

Get compressor name.

Implements filevault::compression::ICompressor.

The documentation for this class was generated from the following files:

- include/filevault/compression/compressor.hpp
- src/compression/compressor.cpp

Chapter 10

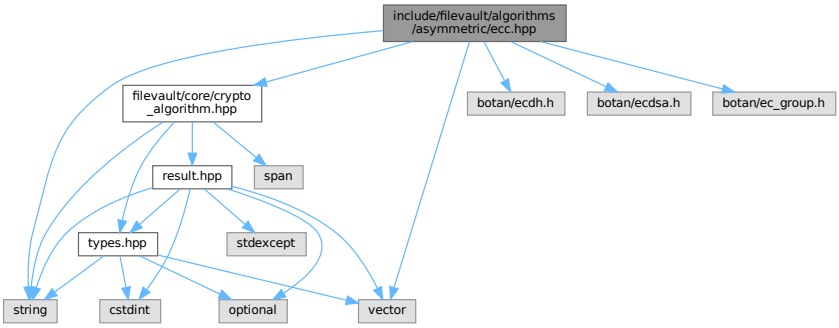
File Documentation

10.1 include/filevault/algorithms/asymmetric/ecc.hpp File Reference

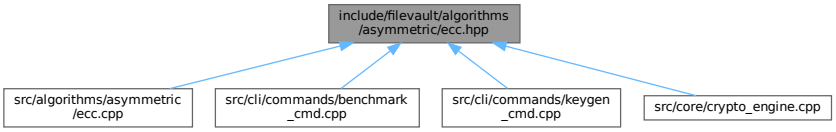
Elliptic Curve Cryptography (ECC) implementation.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/ecdh.h>
#include <botan/ecdsa.h>
#include <botan/ec_group.h>
#include <string>
#include <vector>
```

Include dependency graph for ecc.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::algorithms::asymmetric::ECCKeyPair

  ECC key pair (ECDH or ECDSA).
- struct filevault::algorithms::asymmetric::ECDHResult

  ECDH Key Exchange result.

- struct filevault::algorithms::asymmetric::ECDSASignResult

  ECDSA signature result.
- class filevault::algorithms::asymmetric::ECDH

  Elliptic Curve Diffie-Hellman (ECDH) key exchange.
- class filevault::algorithms::asymmetric::ECDSA

  Elliptic Curve Digital Signature Algorithm (ECDSA).
- class filevault::algorithms::asymmetric::ECCHybrid

  Hybrid encryption using ECDH + AES-GCM.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::asymmetric

Enumerations

- enum class filevault::algorithms::asymmetric::ECCurve { filevault::algorithms::asymmetric::SECP256R1 , filevault::algorithms::asymmetric::SECP384R1 , filevault::algorithms::asymmetric::SECP521R1 , filevault::algorithms::asymmetr }

  Supported elliptic curves.

10.1.1 Detailed Description

Elliptic Curve Cryptography (ECC) implementation.
Provides ECDH key exchange and ECDSA digital signatures using standard NIST curves.

10.2 ecc.hpp

Go to the documentation of this file.

```
00001
00008
00009 #ifndef FILEVAULT_ALGORITHMS_ASYMMETRIC_ECC_HPP
00010 #define FILEVAULT_ALGORITHMS_ASYMMETRIC_ECC_HPP
00011
00012 #include "filevault/core/crypto_algorithm.hpp"
00013 #include <botan/ecdh.h>
00014 #include <botan/ecdsa.h>
00015 #include <botan/ec_group.h>
00016 #include <string>
00017 #include <vector>
00018
00019 namespace filevault {
00020 namespace algorithms {
00021 namespace asymmetric {
00022
00026 enum class ECCurve {
00027     SECP256R1,  // P-256, 128-bit security
00028     SECP384R1,  // P-384, 192-bit security
00029     SECP521R1,  // P-521, 256-bit security
00030     X25519      // Curve25519, 128-bit security (for ECDH only)
00031 };
00032
00036 struct ECCKeyPair {
00037     std::vector<uint8_t> public_key;   // Raw or DER encoded
00038     std::vector<uint8_t> private_key;  // Raw or DER encoded
00039     ECCurve curve;
00040     std::string curve_name;
00041 };
00042
00046 struct ECDHResult {
00047     bool success;
00048     std::vector<uint8_t> shared_secret;  // Derived shared secret
00049     std::string error_message;
00050 };
00051
00055 struct ECDSASignResult {
00056     bool success;
00057     std::vector<uint8_t> signature;
```

```
00058     std::string error_message;
00059 };
00060
00068 class ECDH {
00069 public:
00074     explicit ECDH(ECCurve curve = ECCurve::SECP256R1);
00075
00076     ~ECDH() = default;
00077
00081     ECCKeyPair generate_key_pair();
00082
00089     ECDHResult derive_shared_secret(
00090         std::span<const uint8_t> own_private_key,
00091         std::span<const uint8_t> peer_public_key
00092     );
00093
00094     std::string name() const;
00095     std::string curve_name() const;
00096     size_t key_size() const;  // In bytes
00097
00098 private:
00099     ECCurve curve_;
00100     std::string botan_curve_name_;
00101 };
00102
00108 class ECDSA {
00109 public:
00114     explicit ECDSA(ECCurve curve = ECCurve::SECP256R1);
00115
00116     ~ECDSA() = default;
00117
00121     ECCKeyPair generate_key_pair();
00122
00129     ECDSASignResult sign(
00130         std::span<const uint8_t> data,
00131         std::span<const uint8_t> private_key
00132     );
00133
00141     bool verify(
00142         std::span<const uint8_t> data,
00143         std::span<const uint8_t> signature,
00144         std::span<const uint8_t> public_key
00145     );
00146
00147     std::string name() const;
00148     std::string curve_name() const;
00149     size_t key_size() const;
00150     size_t signature_size() const;
00151
00152 private:
00153     ECCurve curve_;
00154     std::string botan_curve_name_;
00155 };
00156
00170 class ECCHybrid : public core::ICryptoAlgorithm {
00171 public:
00176     explicit ECCHybrid(ECCurve curve = ECCurve::SECP256R1);
00177
00178     ~ECCHybrid() override = default;
00179
00180     std::string name() const override;
00181     core::AlgorithmType type() const override;
00182
00189     core::CryptoResult encrypt(
00190         std::span<const uint8_t> plaintext,
00191         std::span<const uint8_t> key,
00192         const core::EncryptionConfig& config
00193     ) override;
00194
00201     core::CryptoResult decrypt(
00202         std::span<const uint8_t> ciphertext,
00203         std::span<const uint8_t> key,
00204         const core::EncryptionConfig& config
00205     ) override;
00206
00210     ECCKeyPair generate_key_pair();
00211
00212     size_t key_size() const override;
00213     bool is_suitable_for(core::SecurityLevel level) const override;
00214
00215 private:
00216     ECCurve curve_;
00217     std::string botan_curve_name_;
00218     core::AlgorithmType type_;
00219     ECDH ecdh_;
00220 };
00221
```

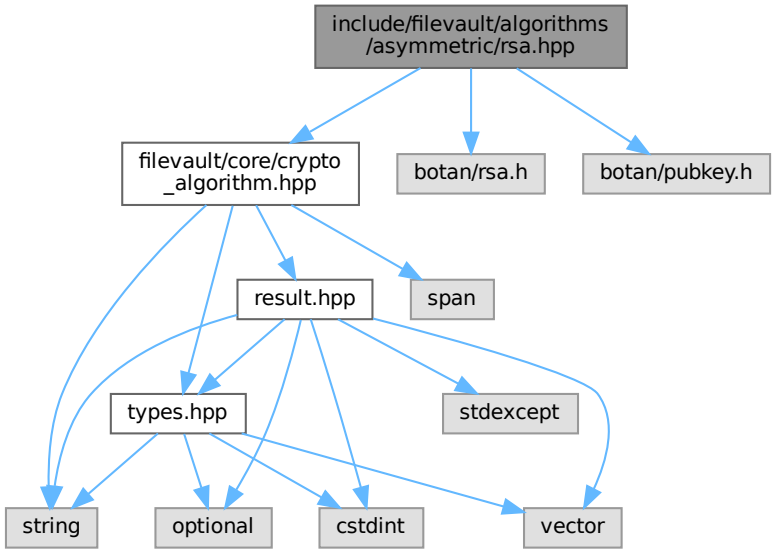
```
00222 } // namespace asymmetric
00223 } // namespace algorithms
00224 } // namespace filevault
00225
00226 #endif
```

10.3 include/filevault/algorithms/asymmetric/rsa.hpp File Reference

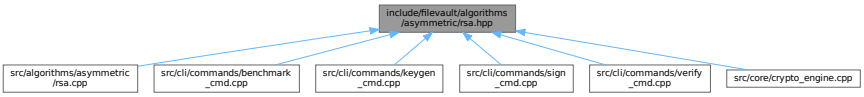
RSA asymmetric encryption.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/rsa.h>
#include <botan/pubkey.h>
```

Include dependency graph for rsa.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::algorithms::asymmetric::RSAKeyPair

  RSA key pair.
- class filevault::algorithms::asymmetric::RSA

  RSA asymmetric encryption algorithm.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::asymmetric

10.3.1 Detailed Description

RSA asymmetric encryption.
RSA (Rivest-Shamir-Adleman) is an asymmetric cryptographic algorithm used for secure data transmission and digital signatures.

10.4 rsa.hpp

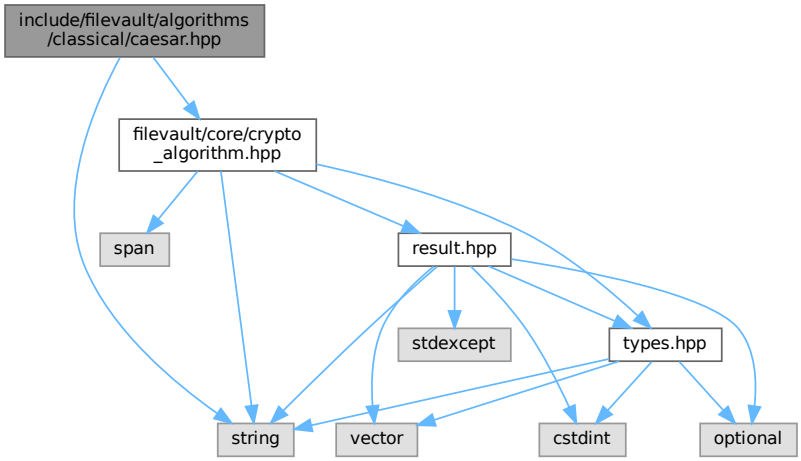
Go to the documentation of this file.

```
00001
00008
00009 #ifndef FILEVAULT_ALGORITHMS_ASYMMETRIC_RSA_HPP
00010 #define FILEVAULT_ALGORITHMS_ASYMMETRIC_RSA_HPP
00011
00012 #include "filevault/core/crypto_algorithm.hpp"
00013 #include <botan/rsa.h>
00014 #include <botan/pubkey.h>
00015
00016 namespace filevault {
00017 namespace algorithms {
00018 namespace asymmetric {
00019
00023 struct RSAKeyPair {
00024     std::vector<uint8_t> public_key;   // PEM or DER encoded
00025     std::vector<uint8_t> private_key;  // PEM or DER encoded
00026     size_t bits;
00027 };
00028
00041 class RSA : public core::ICryptoAlgorithm {
00042 public:
00047     explicit RSA(size_t key_bits = 2048);
00048
00049     ~RSA() override = default;
00050
00051     std::string name() const override;
00052     core::AlgorithmType type() const override;
00053
00060     core::CryptoResult encrypt(
00061         std::span<const uint8_t> plaintext,
00062         std::span<const uint8_t> key,
00063         const core::EncryptionConfig& config
00064     ) override;
00065
00072     core::CryptoResult decrypt(
00073         std::span<const uint8_t> ciphertext,
00074         std::span<const uint8_t> key,
00075         const core::EncryptionConfig& config
00076     ) override;
00077
00082     RSAKeyPair generate_key_pair();
00083
00090     std::vector<uint8_t> sign(
00091         std::span<const uint8_t> data,
00092         std::span<const uint8_t> private_key
00093     );
00094
00102     bool verify(
00103         std::span<const uint8_t> data,
00104         std::span<const uint8_t> signature,
00105         std::span<const uint8_t> public_key
00106     );
00107
00108     size_t key_size() const override { return key_bits_ / 8; }
00109
00114     size_t max_plaintext_size() const { return (key_bits_ / 8) - 66; }
00115
00116     bool requires_padding() const { return true; }  // OAEP padding
00117     bool is_authenticated() const { return false; }
00118     bool is_suitable_for(core::SecurityLevel level) const override;
00119
00120 private:
00121     size_t key_bits_;
00122     core::AlgorithmType type_;
00123 };
00124
00125 } // namespace asymmetric
00126 } // namespace algorithms
00127 } // namespace filevault
00128
00129 #endif
```

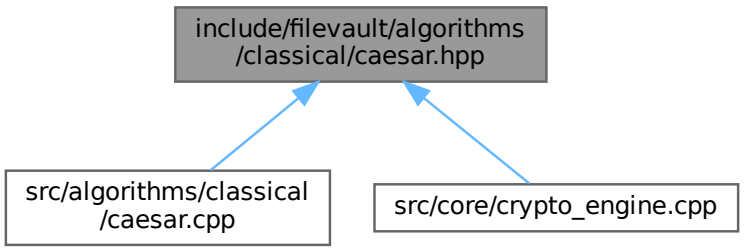
10.5 include/filevault/algorithms/classical/caesar.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <string>
```

Include dependency graph for caesar.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::classical::Caesar

  Caesar Cipher - Educational only.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.6 caesar.hpp

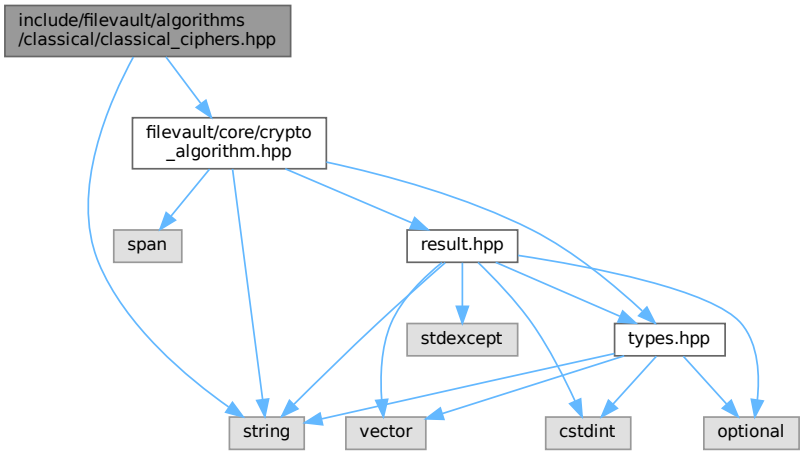
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "filevault/core/crypto_algorithm.hpp"
00004 #include <string>
00005
00006 namespace filevault {
00007 namespace algorithms {
00008 namespace classical {
00009
00021 class Caesar : public core::ICryptoAlgorithm {
00022 public:
00023     explicit Caesar(int shift = 3);
00024
00025     std::string name() const override { return "Caesar"; }
00026     core::AlgorithmType type() const override { return core::AlgorithmType::CAESAR; }
00027     size_t key_size() const override { return 4; } // Min for Argon2, use first byte as shift
00028
00029     core::CryptoResult encrypt(
00030         std::span<const uint8_t> plaintext,
00031         std::span<const uint8_t> key,
00032         const core::EncryptionConfig& config) override;
00033
00034     core::CryptoResult decrypt(
00035         std::span<const uint8_t> ciphertext,
00036         std::span<const uint8_t> key,
00037         const core::EncryptionConfig& config) override;
00038
00039     bool is_suitable_for(core::SecurityLevel level) const override {
00040         (void)level; // Educational only - not for real security
00041         return false;
00042     }
00043
00049     static std::string brute_force(const std::string& ciphertext);
00050
00051 private:
00052     int shift_;
00053
00054     char shift_char(char ch, int shift) const;
00055 };
00056
00057 } // namespace classical
00058 } // namespace algorithms
00059 } // namespace filevault
```

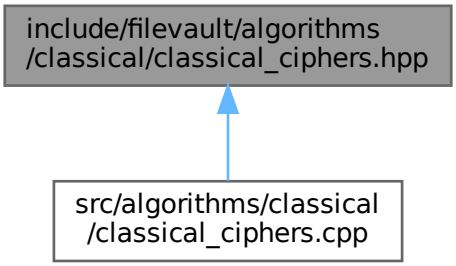
10.7 include/filevault/algorithms/classical/classical_ciphers.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <string>
```

Include dependency graph for classical_ciphers.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::classical::Caesar

  Caesar Cipher - Educational only.
- class filevault::algorithms::classical::Vigenere

  Vigenre Cipher - Polyalphabetic substitution.
- class filevault::algorithms::classical::Playfair

  Playfair Cipher - Digraph substitution.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.8 classical_ciphers.hpp

Go to the documentation of this file.

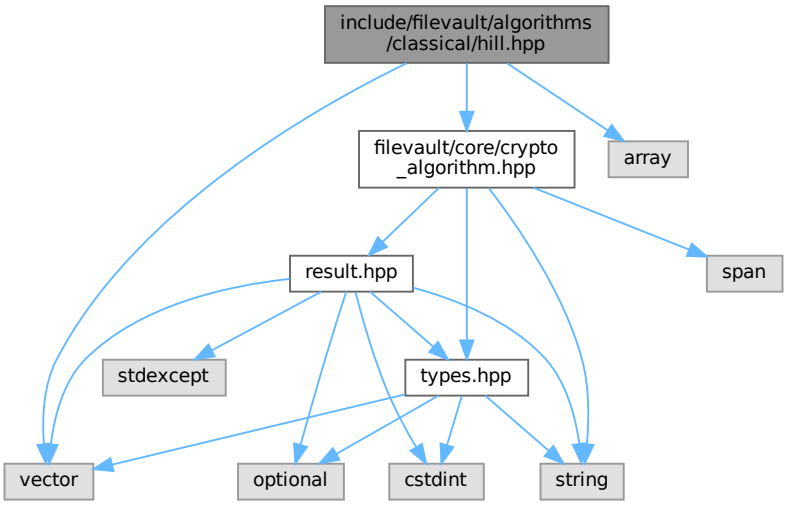
```
00001 #pragma once
00002
00003 #include "filevault/core/crypto_algorithm.hpp"
00004 #include <string>
00005
00006 namespace filevault {
00007 namespace algorithms {
00008 namespace classical {
00009
00019 class Caesar : public core::ICryptoAlgorithm {
00020 public:
00021     explicit Caesar(int shift = 3);
00022
00023     std::string name() const override { return "Caesar"; }
00024     core::AlgorithmType type() const override { return core::AlgorithmType::CAESAR; }
00025     size_t key_size() const override { return 4; } // Min for Argon2, use first byte as shift
00026
00027     core::CryptoResult encrypt(
00028         std::span<const uint8_t> plaintext,
00029         std::span<const uint8_t> key,
00030         const core::EncryptionConfig& config) override;
00031
00032     core::CryptoResult decrypt(
00033         std::span<const uint8_t> ciphertext,
00034         std::span<const uint8_t> key,
00035         const core::EncryptionConfig& config) override;
00036
00037     bool is_suitable_for(core::SecurityLevel level) const override {
00038         (void)level; // Educational only - not for real security
00039         return false;
```

```
00040     }
00041
00042     // Educational methods
00043     static std::string brute_force(const std::string& ciphertext);
00044     static double frequency_analysis(const std::string& text);
00045
00046 private:
00047     int shift_;
00048
00049     char shift_char(char ch, int shift) const;
00050 };
00051
00061 class Vigenere : public core::ICryptoAlgorithm {
00062 public:
00063     explicit Vigenere(const std::string& keyword = "KEY");
00064
00065     std::string name() const override { return "Vigenere"; }
00066     core::AlgorithmType type() const override { return core::AlgorithmType::VIGENERE; }
00067     size_t key_size() const override { return 32; } // Standard key size
00068
00069     core::CryptoResult encrypt(
00070         std::span<const uint8_t> plaintext,
00071         std::span<const uint8_t> key,
00072         const core::EncryptionConfig& config) override;
00073
00074     core::CryptoResult decrypt(
00075         std::span<const uint8_t> ciphertext,
00076         std::span<const uint8_t> key,
00077         const core::EncryptionConfig& config) override;
00078
00079     bool is_suitable_for(core::SecurityLevel level) const override {
00080         (void)level; // Educational only - not for real security
00081         return false;
00082     }
00083
00084     // Educational methods
00085     static std::vector<size_t> kasiski_examination(const std::string& ciphertext);
00086     static size_t estimate_key_length(const std::string& ciphertext);
00087
00088 private:
00089     std::string keyword_;
00090 };
00091
00101 class Playfair : public core::ICryptoAlgorithm {
00102 public:
00103     explicit Playfair(const std::string& keyword = "KEYWORD");
00104
00105     std::string name() const override { return "Playfair"; }
00106     core::AlgorithmType type() const override { return core::AlgorithmType::PLAYFAIR; }
00107     size_t key_size() const override { return 32; } // Standard key size
00108
00109     core::CryptoResult encrypt(
00110         std::span<const uint8_t> plaintext,
00111         std::span<const uint8_t> key,
00112         const core::EncryptionConfig& config) override;
00113
00114     core::CryptoResult decrypt(
00115         std::span<const uint8_t> ciphertext,
00116         std::span<const uint8_t> key,
00117         const core::EncryptionConfig& config) override;
00118
00119     bool is_suitable_for(core::SecurityLevel level) const override {
00120         (void)level; // Educational only - not for real security
00121         return false;
00122     }
00123
00124 private:
00125     char matrix_[5][5];
00126     void build_matrix(const std::string& keyword);
00127     std::pair<int, int> find_position(char ch) const;
00128 };
00129
00130 } // namespace classical
00131 } // namespace algorithms
00132 } // namespace filevault
```

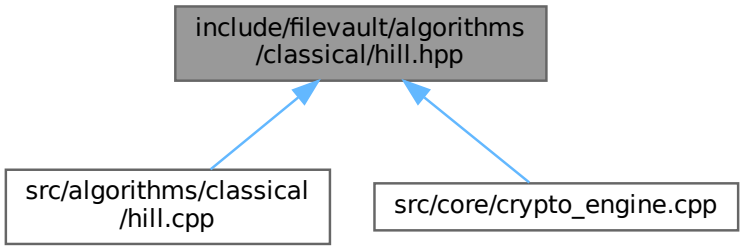
10.9 include/filevault/algorithms/classical/hill.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <vector>
#include <array>
```

Include dependency graph for hill.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::classical::HillCipher

  Hill cipher (2x2 matrix).

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.10 hill.hpp

Go to the documentation of this file.

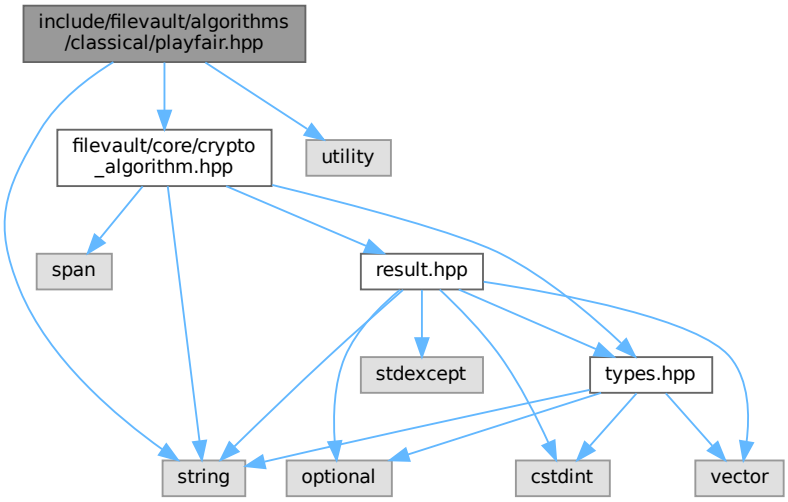
```
00001 #ifndef FILEVAULT_ALGORITHMS_CLASSICAL_HILL_HPP
00002 #define FILEVAULT_ALGORITHMS_CLASSICAL_HILL_HPP
```

```
00003
00004 #include "filevault/core/crypto_algorithm.hpp"
00005 #include <vector>
00006 #include <array>
00007
00008 namespace filevault {
00009 namespace algorithms {
00010 namespace classical {
00011
00018 class HillCipher : public core::ICryptoAlgorithm {
00019 public:
00020     std::string name() const override { return "Hill"; }
00021
00022     core::AlgorithmType type() const override {
00023         return core::AlgorithmType::HILL;
00024     }
00025
00026     core::CryptoResult encrypt(
00027         std::span<const uint8_t> plaintext,
00028         std::span<const uint8_t> key,
00029         const core::EncryptionConfig& config
00030     ) override;
00031
00032     core::CryptoResult decrypt(
00033         std::span<const uint8_t> ciphertext,
00034         std::span<const uint8_t> key,
00035         const core::EncryptionConfig& config
00036     ) override;
00037
00038     size_t key_size() const override { return 4; }  // 2x2 matrix = 4 values
00039
00040     bool is_suitable_for(core::SecurityLevel level) const override {
00041         // Educational cipher - suitable for learning only
00042         (void)level;
00043         return false;  // Not recommended for real encryption
00044     }
00045
00046 private:
00047     using Matrix2x2 = std::array<int, 4>;  // [a,b,c,d] for matrix [[a,b],[c,d]]
00048
00049     Matrix2x2 parse_key(std::span<const uint8_t> key);
00050     Matrix2x2 invert_matrix(const Matrix2x2& matrix);
00051     int mod_inverse(int a, int m);
00052     int determinant(const Matrix2x2& matrix);
00053     bool is_valid_key(const Matrix2x2& matrix);
00054
00055     std::string encrypt_block(const std::string& block, const Matrix2x2& key_matrix);
00056     std::string decrypt_block(const std::string& block, const Matrix2x2& inv_matrix);
00057 };
00058
00059 } // namespace classical
00060 } // namespace algorithms
00061 } // namespace filevault
00062
00063 #endif
```

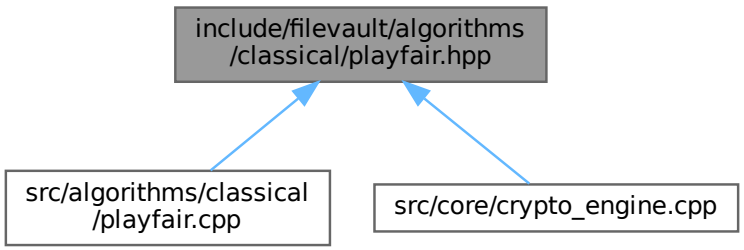
10.11 include/filevault/algorithms/classical/playfair.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <string>
#include <utility>
```

Include dependency graph for playfair.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::classical::Playfair

  Playfair Cipher - Digraph substitution.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.12 playfair.hpp

Go to the documentation of this file.

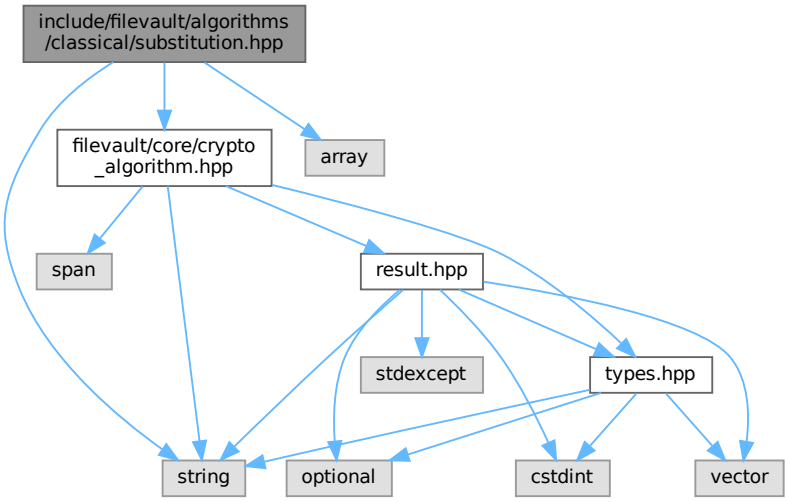
```
00001 #pragma once
00002
```

```
00003 #include "filevault/core/crypto_algorithm.hpp"
00004 #include <string>
00005 #include <utility>
00006
00007 namespace filevault {
00008 namespace algorithms {
00009 namespace classical {
00010
00022 class Playfair : public core::ICryptoAlgorithm {
00023 public:
00024     explicit Playfair(const std::string& keyword = "KEYWORD");
00025
00026     std::string name() const override { return "Playfair"; }
00027     core::AlgorithmType type() const override { return core::AlgorithmType::PLAYFAIR; }
00028     size_t key_size() const override { return 32; } // Standard key size
00029
00030     core::CryptoResult encrypt(
00031         std::span<const uint8_t> plaintext,
00032         std::span<const uint8_t> key,
00033         const core::EncryptionConfig& config) override;
00034
00035     core::CryptoResult decrypt(
00036         std::span<const uint8_t> ciphertext,
00037         std::span<const uint8_t> key,
00038         const core::EncryptionConfig& config) override;
00039
00040     bool is_suitable_for(core::SecurityLevel level) const override {
00041         (void)level;
00042         return false;
00043     }
00044
00045 private:
00046     char matrix_[5][5];
00047
00048     void build_matrix(const std::string& keyword);
00049     std::pair<int, int> find_position(char ch) const;
00050 };
00051
00052 } // namespace classical
00053 } // namespace algorithms
00054 } // namespace filevault
```

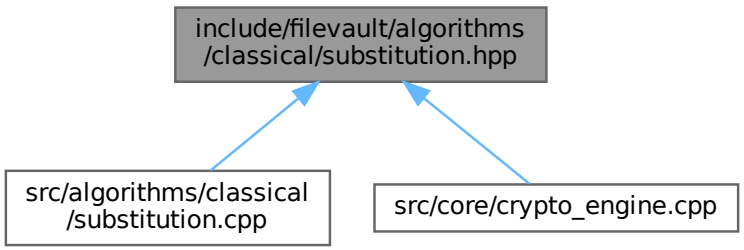
10.13 include/filevault/algorithms/classical/substitution.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <array>
#include <string>
```

Include dependency graph for substitution.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::classical::SubstitutionCipher

  Simple substitution cipher.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.14 substitution.hpp

Go to the documentation of this file.

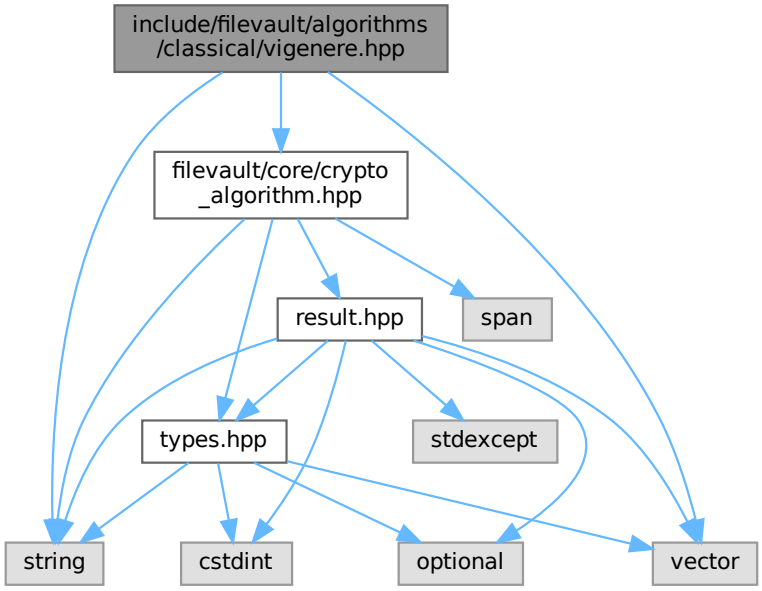
```
00001 #ifndef FILEVAULT_ALGORITHMS_CLASSICAL_SUBSTITUTION_HPP
00002 #define FILEVAULT_ALGORITHMS_CLASSICAL_SUBSTITUTION_HPP
```

```
00003
00004 #include "filevault/core/crypto_algorithm.hpp"
00005 #include <array>
00006 #include <string>
00007
00008 namespace filevault {
00009 namespace algorithms {
00010 namespace classical {
00011
00018 class SubstitutionCipher : public core::ICryptoAlgorithm {
00019 public:
00020     std::string name() const override { return "Substitution"; }
00021
00022     core::AlgorithmType type() const override {
00023         return core::AlgorithmType::SUBSTITUTION;
00024     }
00025
00026     core::CryptoResult encrypt(
00027         std::span<const uint8_t> plaintext,
00028         std::span<const uint8_t> key,
00029         const core::EncryptionConfig& config
00030     ) override;
00031
00032     core::CryptoResult decrypt(
00033         std::span<const uint8_t> ciphertext,
00034         std::span<const uint8_t> key,
00035         const core::EncryptionConfig& config
00036     ) override;
00037
00038     size_t key_size() const override { return 26; }  // 26-letter alphabet
00039
00040     bool is_suitable_for(core::SecurityLevel level) const override {
00041         // Educational cipher - suitable for learning only
00042         (void)level;
00043         return false;  // Not recommended for real encryption
00044     }
00045
00046 private:
00047     using SubstitutionMap = std::array<char, 26>;
00048
00049     SubstitutionMap parse_key(std::span<const uint8_t> key);
00050     SubstitutionMap create_reverse_map(const SubstitutionMap& forward_map);
00051     bool is_valid_key(const SubstitutionMap& map);
00052
00053     std::string apply_substitution(const std::string& text, const SubstitutionMap& map);
00054 };
00055
00056 } // namespace classical
00057 } // namespace algorithms
00058 } // namespace filevault
00059
00060 #endif
```

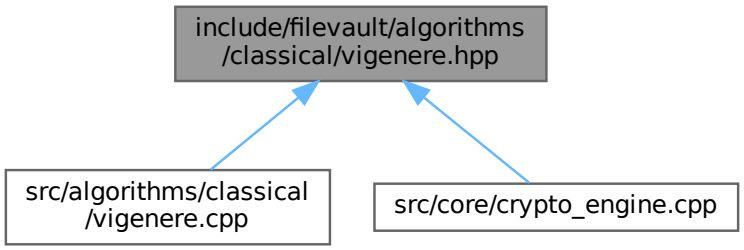
10.15 include/filevault/algorithms/classical/vigenere.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <string>
#include <vector>
```

Include dependency graph for vigenere.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::classical::Vigenere

  Vigenre Cipher - Polyalphabetic substitution.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.16 vigenere.hpp

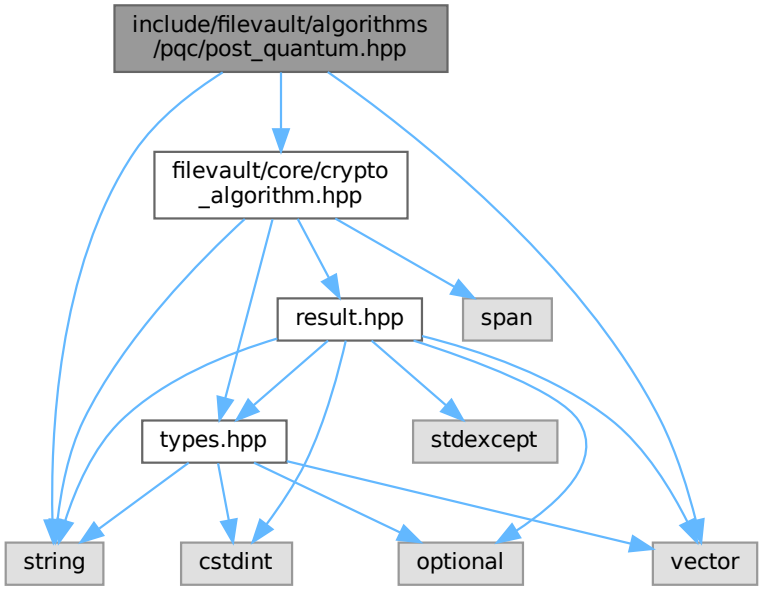
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "filevault/core/crypto_algorithm.hpp"
00004 #include <string>
00005 #include <vector>
00006
00007 namespace filevault {
00008 namespace algorithms {
00009 namespace classical {
00010
00022 class Vigenere : public core::ICryptoAlgorithm {
00023 public:
00024     explicit Vigenere(const std::string& keyword = "KEY");
00025
00026     std::string name() const override { return "Vigenere"; }
00027     core::AlgorithmType type() const override { return core::AlgorithmType::VIGENERE; }
00028     size_t key_size() const override { return 32; } // Standard key size
00029
00030     core::CryptoResult encrypt(
00031         std::span<const uint8_t> plaintext,
00032         std::span<const uint8_t> key,
00033         const core::EncryptionConfig& config) override;
00034
00035     core::CryptoResult decrypt(
00036         std::span<const uint8_t> ciphertext,
00037         std::span<const uint8_t> key,
00038         const core::EncryptionConfig& config) override;
00039
00040     bool is_suitable_for(core::SecurityLevel level) const override {
00041         (void)level;
00042         return false;
00043     }
00044
00045 private:
00046     std::string keyword_;
00047 };
00048
00049 } // namespace classical
00050 } // namespace algorithms
00051 } // namespace filevault
```

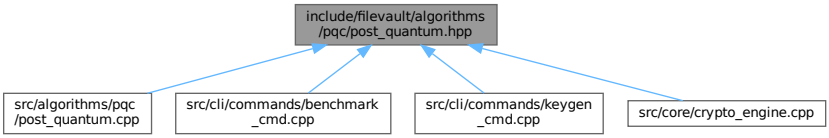
10.17 include/filevault/algorithms/pqc/post_quantum.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <string>
#include <vector>
```

Include dependency graph for post_quantum.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::algorithms::pqc::PQKeyPair

  Key pair for post-quantum algorithms.
- class filevault::algorithms::pqc::Kyber

  Kyber Key Encapsulation Mechanism (KEM).
- class filevault::algorithms::pqc::Dilithium

  Dilithium Digital Signature Algorithm.
- class filevault::algorithms::pqc::KyberHybrid

  Hybrid encryption combining Kyber with AES-GCM.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::pqc

10.18 post_quantum.hpp

Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_ALGORITHMS_POST_QUANTUM_HPP
00002 #define FILEVAULT_ALGORITHMS_POST_QUANTUM_HPP
00003
00004 #include "filevault/core/crypto_algorithm.hpp"
00005 #include <string>
00006 #include <vector>
00007
00008 namespace filevault {
00009 namespace algorithms {
00010 namespace pqc {
00011
00015 struct PQKeyPair {
00016     std::vector<uint8_t> public_key;
00017     std::vector<uint8_t> private_key;
00018     std::string algorithm;
00019 };
00020
00036 class Kyber : public core::ICryptoAlgorithm {
00037 public:
00038     enum class Variant {
00039         Kyber512,   // AES-128 equivalent
00040         Kyber768,   // AES-192 equivalent (recommended)
00041         Kyber1024   // AES-256 equivalent
00042     };
00043
00044     explicit Kyber(Variant variant = Variant::Kyber768);
00045     ~Kyber() override = default;
00046
00047     std::string name() const override;
00048     core::AlgorithmType type() const override;
00049
00061     core::CryptoResult encrypt(
00062         std::span<const uint8_t> plaintext,
00063         std::span<const uint8_t> key,
00064         const core::EncryptionConfig& config
00065     ) override;
00066
00077     core::CryptoResult decrypt(
00078         std::span<const uint8_t> ciphertext,
00079         std::span<const uint8_t> key,
00080         const core::EncryptionConfig& config
00081     ) override;
00082
00083     size_t key_size() const override;
00084     bool is_suitable_for(core::SecurityLevel level) const override;
00085
00089     PQKeyPair generate_keypair();
00090
00094     size_t public_key_size() const;
00095
00099     size_t private_key_size() const;
00100
00104     size_t ciphertext_size() const;
00105
00109     size_t shared_secret_size() const { return 32; }
00110
00111 private:
00112     Variant variant_;
00113     std::string botan_name_;
00114     core::AlgorithmType type_;
00115 };
00116
00132 class Dilithium {
00133 public:
00134     enum class Variant {
00135         Dilithium2,  // Level 2 security
00136         Dilithium3,  // Level 3 security (recommended)
00137         Dilithium5   // Level 5 security
00138     };
00139
00140     explicit Dilithium(Variant variant = Variant::Dilithium3);
00141
00142     std::string name() const;
00143
00147     PQKeyPair generate_keypair();
00148
00155     std::vector<uint8_t> sign(
00156         std::span<const uint8_t> message,
00157         std::span<const uint8_t> private_key
00158     );
00159
00167     bool verify(
00168         std::span<const uint8_t> message,
```

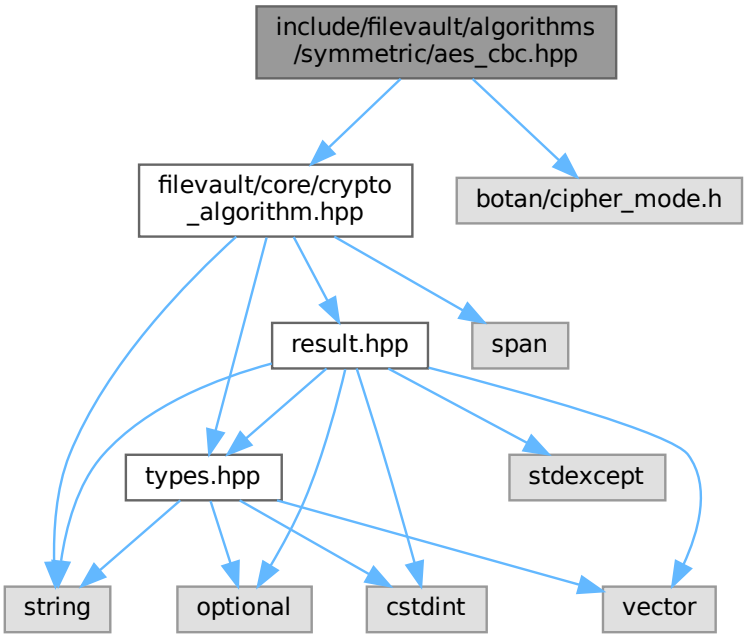
```
00169         std::span<const uint8_t> signature,
00170         std::span<const uint8_t> public_key
00171     );
00172
00176     size_t public_key_size() const;
00177
00181     size_t private_key_size() const;
00182
00186     size_t signature_size() const;
00187
00188 private:
00189     Variant variant_;
00190     std::string botan_name_;
00191 };
00192
00202 class KyberHybrid : public core::ICryptoAlgorithm {
00203 public:
00204     explicit KyberHybrid(Kyber::Variant variant = Kyber::Variant::Kyber768);
00205     ~KyberHybrid() override = default;
00206
00207     std::string name() const override;
00208     core::AlgorithmType type() const override;
00209
00222     core::CryptoResult encrypt(
00223         std::span<const uint8_t> plaintext,
00224         std::span<const uint8_t> key,
00225         const core::EncryptionConfig& config
00226     ) override;
00227
00240     core::CryptoResult decrypt(
00241         std::span<const uint8_t> ciphertext,
00242         std::span<const uint8_t> key,
00243         const core::EncryptionConfig& config
00244     ) override;
00245
00246     size_t key_size() const override;
00247     bool is_suitable_for(core::SecurityLevel level) const override;
00248
00252     PQKeyPair generate_keypair();
00253
00254 private:
00255     Kyber kyber_;
00256 };
00257
00258 } // namespace pqc
00259 } // namespace algorithms
00260 } // namespace filevault
00261
00262 #endif // FILEVAULT_ALGORITHMS_POST_QUANTUM_HPP
```

10.19 include/filevault/algorithms/symmetric/aes_cbc.hpp File Reference

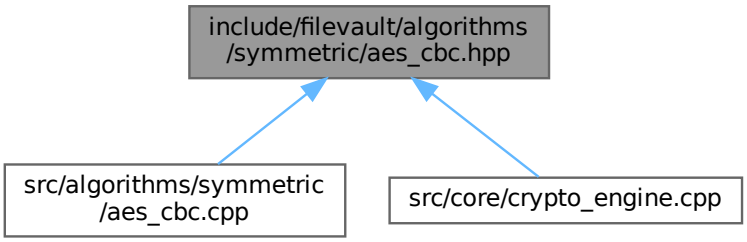
AES-CBC encryption implementation.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/cipher_mode.h>
```

Include dependency graph for aes_cbc.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::AES_CBC

  AES-CBC encryption (legacy mode, not authenticated).

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.19.1 Detailed Description

AES-CBC encryption implementation.
AES in Cipher Block Chaining mode - classic block cipher mode. NOT authenticated - use with HMAC for integrity protection.

Warning

CBC mode is NOT AEAD - no built-in authentication! Consider using AES-GCM instead for authenticated encryption.

10.20 aes_cbc.hpp

Go to the documentation of this file.

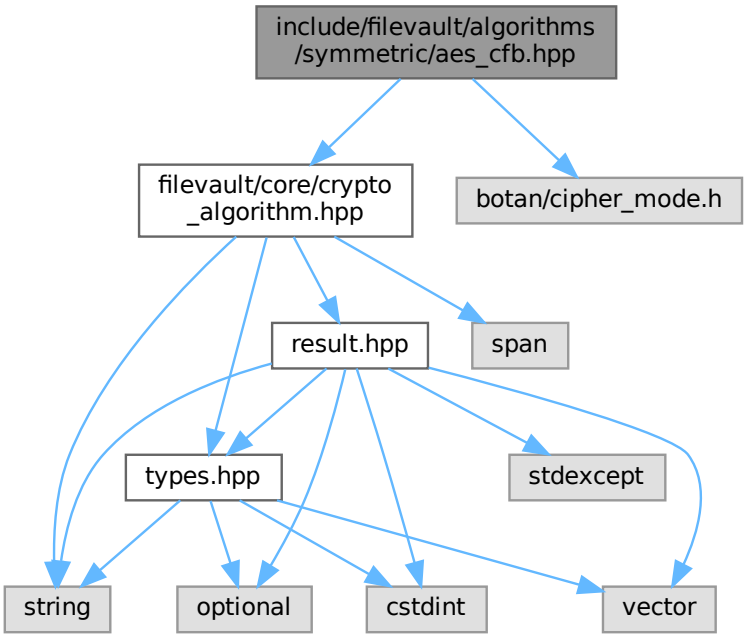
```
00001
00011
00012 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CBC_HPP
00013 #define FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CBC_HPP
00014
00015 #include "filevault/core/crypto_algorithm.hpp"
00016 #include <botan/cipher_mode.h>
00017
00018 namespace filevault {
00019 namespace algorithms {
00020 namespace symmetric {
00021
00033 class AES_CBC : public core::ICryptoAlgorithm {
00034 public:
00035     explicit AES_CBC(size_t key_bits = 256);
00036     virtual ~AES_CBC() = default;
00037
00038     std::string name() const override;
00039     core::AlgorithmType type() const override;
00040
00041     core::CryptoResult encrypt(
00042         std::span<const uint8_t> plaintext,
00043         std::span<const uint8_t> key,
00044         const core::EncryptionConfig& config
00045     ) override;
00046
00047     core::CryptoResult decrypt(
00048         std::span<const uint8_t> ciphertext,
00049         std::span<const uint8_t> key,
00050         const core::EncryptionConfig& config
00051     ) override;
00052
00053     size_t key_size() const override { return key_bits_ / 8; }
00054     size_t iv_size() const { return 16; }  // AES block size
00055     size_t block_size() const { return 16; }
00056
00057     bool is_suitable_for(core::SecurityLevel level) const override;
00058
00059 private:
00060     size_t key_bits_;
00061     core::AlgorithmType type_;
00062     std::string botan_name_;
00063 };
00064
00065 } // namespace symmetric
00066 } // namespace algorithms
00067 } // namespace filevault
00068
00069 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CBC_HPP
```

10.21 include/filevault/algorithms/symmetric/aes_cfb.hpp File Reference

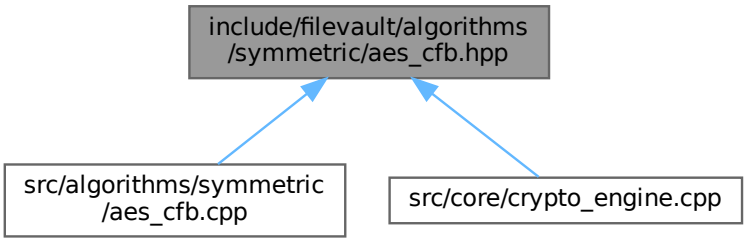
AES-CFB (Cipher Feedback) mode encryption.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/cipher_mode.h>
```

Include dependency graph for aes_cfb.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::AES_CFB

  AES-CFB encryption algorithm.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.21.1 Detailed Description

AES-CFB (Cipher Feedback) mode encryption.
CFB mode is a stream cipher mode that turns a block cipher into a self-synchronizing stream cipher. It's similar to CBC but encrypts the previous ciphertext block instead of XORing.

10.22 aes_cfb.hpp

Go to the documentation of this file.

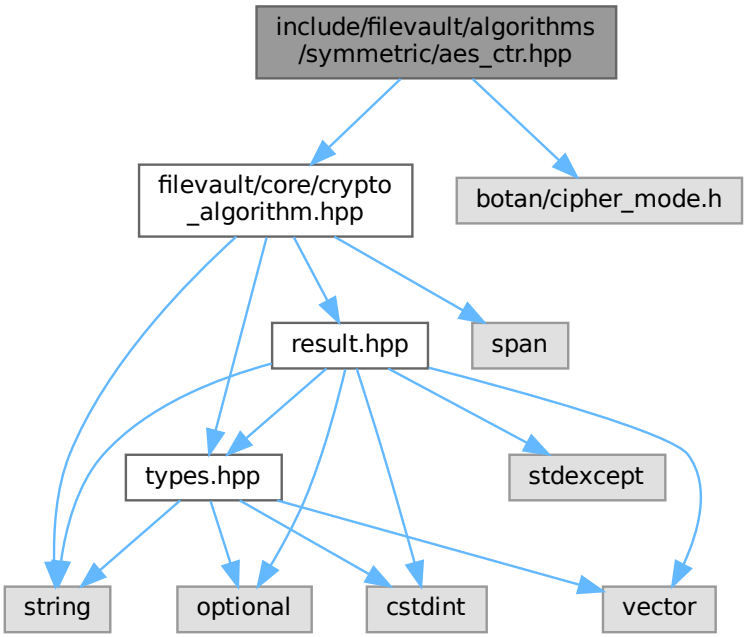
```
00001
00009
00010 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CFB_HPP
00011 #define FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CFB_HPP
00012
00013 #include "filevault/core/crypto_algorithm.hpp"
00014 #include <botan/cipher_mode.h>
00015
00016 namespace filevault {
00017 namespace algorithms {
00018 namespace symmetric {
00019
00029 class AES_CFB : public core::ICryptoAlgorithm {
00030 public:
00035     explicit AES_CFB(size_t key_bits = 256);
00036
00037     ~AES_CFB() override = default;
00038
00039     std::string name() const override;
00040     core::AlgorithmType type() const override;
00041
00042     core::CryptoResult encrypt(
00043         std::span<const uint8_t> plaintext,
00044         std::span<const uint8_t> key,
00045         const core::EncryptionConfig& config
00046     ) override;
00047
00048     core::CryptoResult decrypt(
00049         std::span<const uint8_t> ciphertext,
00050         std::span<const uint8_t> key,
00051         const core::EncryptionConfig& config
00052     ) override;
00053
00054     size_t key_size() const override { return key_bits_ / 8; }
00055     size_t iv_size() const { return 16; }  // AES block size
00056     size_t block_size() const { return 16; }
00057
00058     bool requires_padding() const { return false; }
00059     bool is_authenticated() const { return false; }
00060     bool is_suitable_for(core::SecurityLevel level) const override;
00061
00062 private:
00063     size_t key_bits_;
00064     core::AlgorithmType type_;
00065     std::string botan_name_;
00066 };
00067
00068 } // namespace symmetric
00069 } // namespace algorithms
00070 } // namespace filevault
00071
00072 #endif
```

10.23 include/filevault/algorithms/symmetric/aes_ctr.hpp File Reference

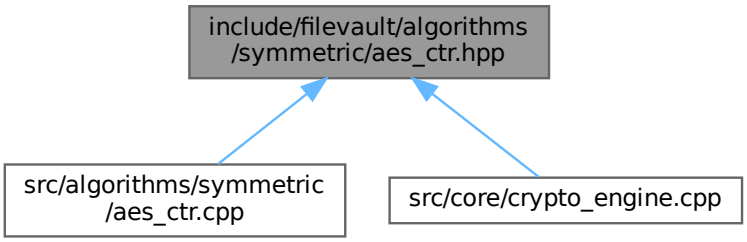
AES-CTR (Counter mode) encryption implementation.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/cipher_mode.h>
```

Include dependency graph for aes_ctr.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::AES_CTR

  AES-CTR encryption (stream cipher mode, not authenticated).

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.23.1 Detailed Description

AES-CTR (Counter mode) encryption implementation.
AES in Counter mode - stream cipher mode. NOT authenticated - use with HMAC for integrity protection.

10.24 aes_ctr.hpp

Go to the documentation of this file.

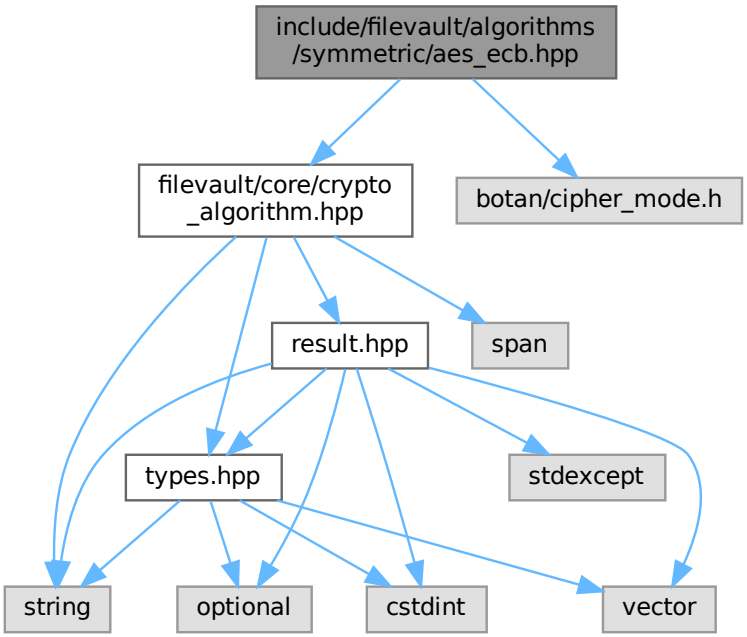
```
00001
00008
00009 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CTR_HPP
00010 #define FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CTR_HPP
00011
00012 #include "filevault/core/crypto_algorithm.hpp"
00013 #include <botan/cipher_mode.h>
00014
00015 namespace filevault {
00016 namespace algorithms {
00017 namespace symmetric {
00018
00028 class AES_CTR : public core::ICryptoAlgorithm {
00029 public:
00030     explicit AES_CTR(size_t key_bits = 256);
00031     virtual ~AES_CTR() = default;
00032
00033     std::string name() const override;
00034     core::AlgorithmType type() const override;
00035
00036     core::CryptoResult encrypt(
00037         std::span<const uint8_t> plaintext,
00038         std::span<const uint8_t> key,
00039         const core::EncryptionConfig& config
00040     ) override;
00041
00042     core::CryptoResult decrypt(
00043         std::span<const uint8_t> ciphertext,
00044         std::span<const uint8_t> key,
00045         const core::EncryptionConfig& config
00046     ) override;
00047
00048     size_t key_size() const override { return key_bits_ / 8; }
00049     size_t nonce_size() const { return 16; }  // Full block as IV/counter
00050
00051     bool is_suitable_for(core::SecurityLevel level) const override;
00052
00053 private:
00054     size_t key_bits_;
00055     core::AlgorithmType type_;
00056     std::string botan_name_;
00057 };
00058
00059 } // namespace symmetric
00060 } // namespace algorithms
00061 } // namespace filevault
00062
00063 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_AES_CTR_HPP
```

10.25 include/filevault/algorithms/symmetric/aes_ecb.hpp File Reference

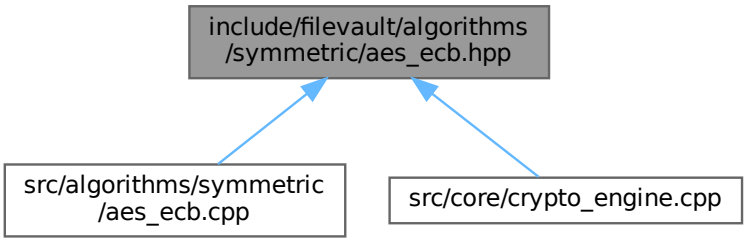
AES-ECB (Electronic Codebook) mode encryption.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/cipher_mode.h>
```

Include dependency graph for aes_ecb.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::AES_ECB

  AES-ECB encryption algorithm.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.25.1 Detailed Description

AES-ECB (Electronic Codebook) mode encryption.
WARNING: ECB mode is NOT secure for most purposes! Each block is encrypted independently, which leaks patterns. This is provided for educational purposes and legacy compatibility only.

10.26 aes_ecb.hpp

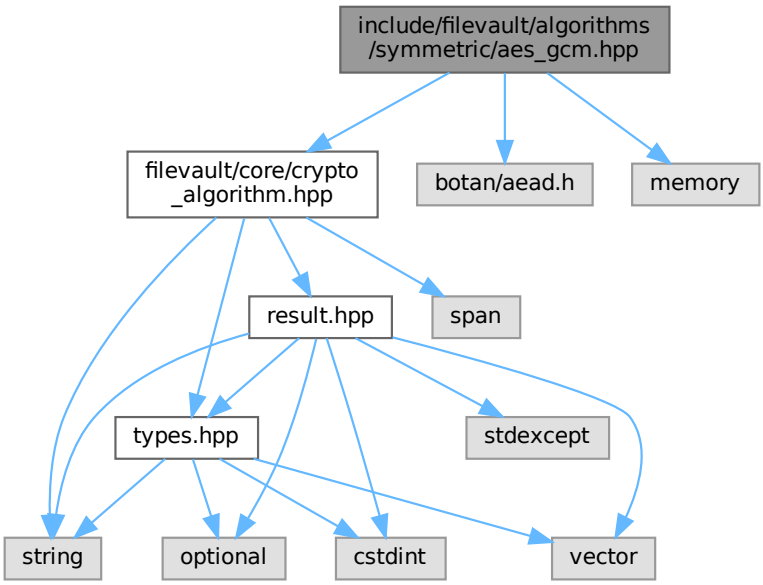
Go to the documentation of this file.

```
00001
00009
00010 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_AES_ECB_HPP
00011 #define FILEVAULT_ALGORITHMS_SYMMETRIC_AES_ECB_HPP
00012
00013 #include "filevault/core/crypto_algorithm.hpp"
00014 #include <botan/cipher_mode.h>
00015
00016 namespace filevault {
00017 namespace algorithms {
00018 namespace symmetric {
00019
00033 class AES_ECB : public core::ICryptoAlgorithm {
00034 public:
00039     explicit AES_ECB(size_t key_bits = 256);
00040
00041     ~AES_ECB() override = default;
00042
00043     std::string name() const override;
00044     core::AlgorithmType type() const override;
00045
00046     core::CryptoResult encrypt(
00047         std::span<const uint8_t> plaintext,
00048         std::span<const uint8_t> key,
00049         const core::EncryptionConfig& config
00050     ) override;
00051
00052     core::CryptoResult decrypt(
00053         std::span<const uint8_t> ciphertext,
00054         std::span<const uint8_t> key,
00055         const core::EncryptionConfig& config
00056     ) override;
00057
00058     size_t key_size() const override { return key_bits_ / 8; }
00059     size_t block_size() const { return 16; }
00060
00061     bool requires_padding() const { return true; }
00062     bool is_authenticated() const { return false; }
00063     bool is_suitable_for(core::SecurityLevel level) const override;
00064
00065 private:
00066     size_t key_bits_;
00067     core::AlgorithmType type_;
00068     std::string botan_name_;
00069 };
00070
00071 } // namespace symmetric
00072 } // namespace algorithms
00073 } // namespace filevault
00074
00075 #endif
```

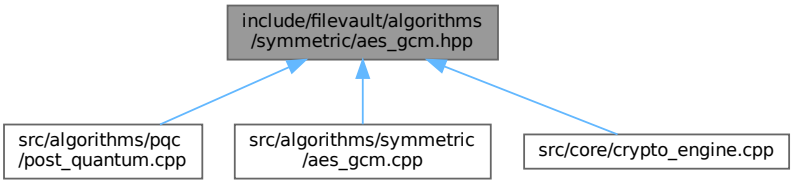
10.27 include/filevault/algorithms/symmetric/aes_gcm.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/aead.h>
#include <memory>
```

Include dependency graph for aes_gcm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::AES_GCM

  AES-GCM AEAD encryption.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.28 aes_gcm.hpp

Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_AES_GCM_HPP
00002 #define FILEVAULT_ALGORITHMS_SYMMETRIC_AES_GCM_HPP
00003
```

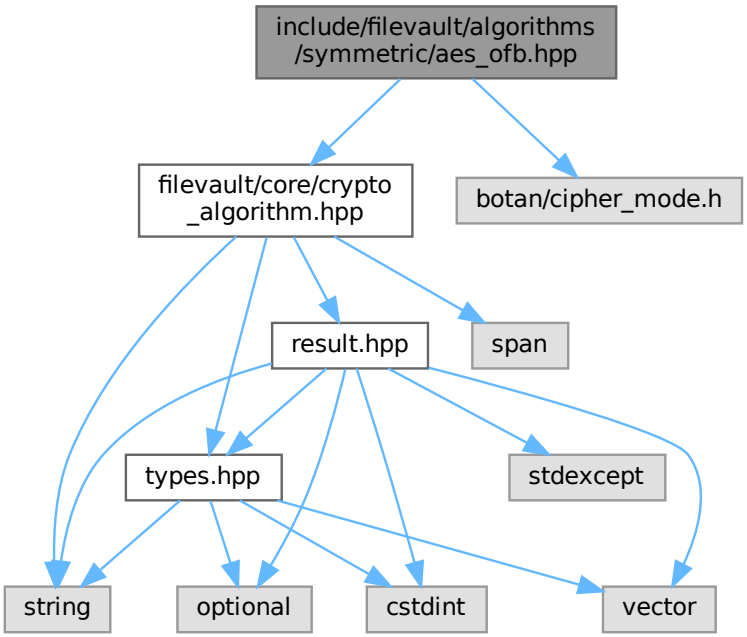
```
00004 #include "filevault/core/crypto_algorithm.hpp"
00005 #include <botan/aead.h>
00006 #include <memory>
00007
00008 namespace filevault {
00009 namespace algorithms {
00010 namespace symmetric {
00011
00015 class AES_GCM : public core::ICryptoAlgorithm {
00016 public:
00017     explicit AES_GCM(size_t key_bits = 256);
00018     virtual ~AES_GCM() = default;
00019
00020     std::string name() const override;
00021     core::AlgorithmType type() const override;
00022
00023     core::CryptoResult encrypt(
00024         std::span<const uint8_t> plaintext,
00025         std::span<const uint8_t> key,
00026         const core::EncryptionConfig& config
00027     ) override;
00028
00029     core::CryptoResult decrypt(
00030         std::span<const uint8_t> ciphertext,
00031         std::span<const uint8_t> key,
00032         const core::EncryptionConfig& config
00033     ) override;
00034
00035     size_t key_size() const override { return key_bits_ / 8; }
00036     size_t nonce_size() const { return 12; } // GCM standard
00037     size_t tag_size() const { return 16; }    // 128-bit tag
00038
00039     bool is_suitable_for(core::SecurityLevel level) const override;
00040
00041 private:
00042     size_t key_bits_;
00043     core::AlgorithmType type_;
00044     std::string botan_name_;
00045 };
00046
00047 } // namespace symmetric
00048 } // namespace algorithms
00049 } // namespace filevault
00050
00051 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_AES_GCM_HPP
```

10.29 include/filevault/algorithms/symmetric/aes_ofb.hpp File Reference

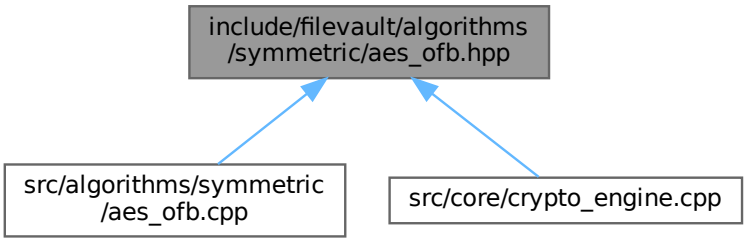
AES-OFB (Output Feedback) mode encryption.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/cipher_mode.h>
```

Include dependency graph for aes_ofb.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::AES_OFB

  AES-OFB encryption algorithm.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.29.1 Detailed Description

AES-OFB (Output Feedback) mode encryption.
OFB mode is a stream cipher mode that generates a keystream independent of plaintext/ciphertext. Pre-computation possible.

10.30 aes_ofb.hpp

Go to the documentation of this file.

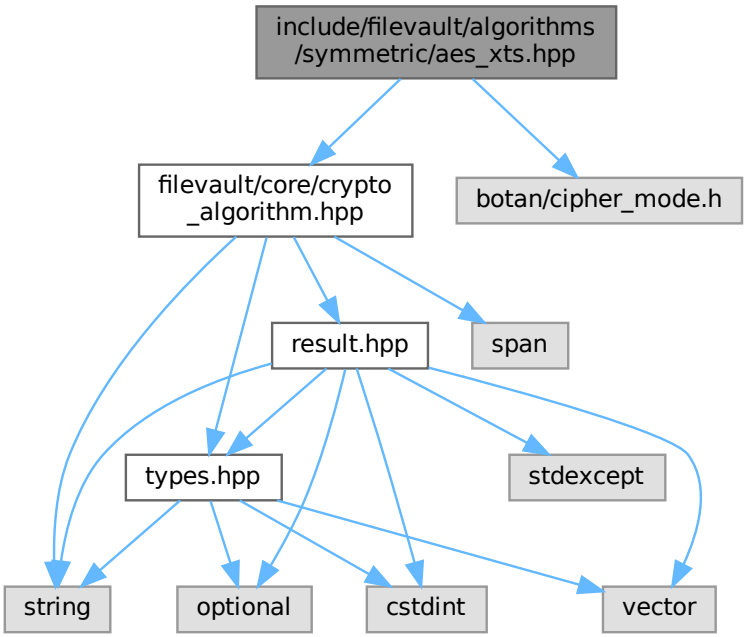
```
00001
00008
00009 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_AES_OFB_HPP
00010 #define FILEVAULT_ALGORITHMS_SYMMETRIC_AES_OFB_HPP
00011
00012 #include "filevault/core/crypto_algorithm.hpp"
00013 #include <botan/cipher_mode.h>
00014
00015 namespace filevault {
00016 namespace algorithms {
00017 namespace symmetric {
00018
00028 class AES_OFB : public core::ICryptoAlgorithm {
00029 public:
00034     explicit AES_OFB(size_t key_bits = 256);
00035
00036     ~AES_OFB() override = default;
00037
00038     std::string name() const override;
00039     core::AlgorithmType type() const override;
00040
00041     core::CryptoResult encrypt(
00042         std::span<const uint8_t> plaintext,
00043         std::span<const uint8_t> key,
00044         const core::EncryptionConfig& config
00045     ) override;
00046
00047     core::CryptoResult decrypt(
00048         std::span<const uint8_t> ciphertext,
00049         std::span<const uint8_t> key,
00050         const core::EncryptionConfig& config
00051     ) override;
00052
00053     size_t key_size() const override { return key_bits_ / 8; }
00054     size_t iv_size() const { return 16; }
00055     size_t block_size() const { return 16; }
00056
00057     bool requires_padding() const { return false; }
00058     bool is_authenticated() const { return false; }
00059     bool is_suitable_for(core::SecurityLevel level) const override;
00060
00061 private:
00062     size_t key_bits_;
00063     core::AlgorithmType type_;
00064     std::string botan_name_;
00065 };
00066
00067 } // namespace symmetric
00068 } // namespace algorithms
00069 } // namespace filevault
00070
00071 #endif
```

10.31 include/filevault/algorithms/symmetric/aes_xts.hpp File Reference

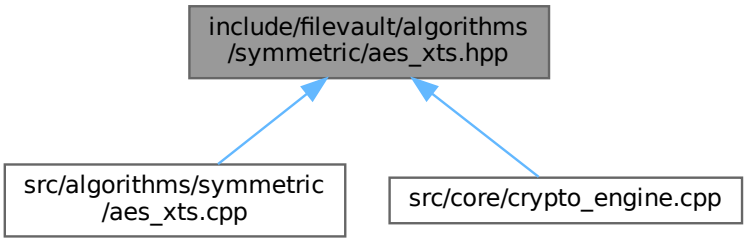
AES-XTS (XEX-based Tweaked-codebook mode with ciphertext Stealing).

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/cipher_mode.h>
```

Include dependency graph for aes_xts.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::AES_XTS

  AES-XTS encryption algorithm.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.31.1 Detailed Description

AES-XTS (XEX-based Tweaked-codebook mode with ciphertext Stealing).
XTS mode is specifically designed for disk encryption. It uses two keys and a "tweak" value (typically sector number).

10.32 aes_xts.hpp

Go to the documentation of this file.

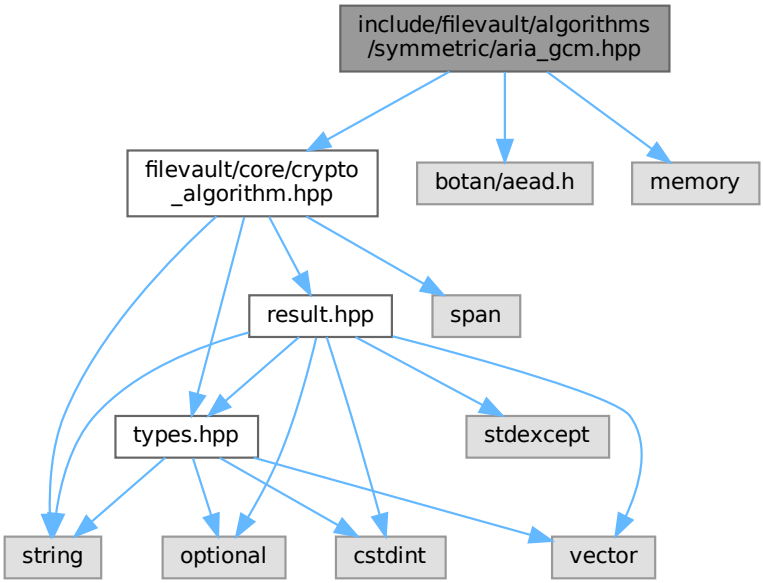
```
00001
00008
00009 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_AES_XTS_HPP
00010 #define FILEVAULT_ALGORITHMS_SYMMETRIC_AES_XTS_HPP
00011
00012 #include "filevault/core/crypto_algorithm.hpp"
00013 #include <botan/cipher_mode.h>
00014
00015 namespace filevault {
00016 namespace algorithms {
00017 namespace symmetric {
00018
00029 class AES_XTS : public core::ICryptoAlgorithm {
00030 public:
00036     explicit AES_XTS(size_t key_bits = 256);
00037
00038     ~AES_XTS() override = default;
00039
00040     std::string name() const override;
00041     core::AlgorithmType type() const override;
00042
00043     core::CryptoResult encrypt(
00044         std::span<const uint8_t> plaintext,
00045         std::span<const uint8_t> key,
00046         const core::EncryptionConfig& config
00047     ) override;
00048
00049     core::CryptoResult decrypt(
00050         std::span<const uint8_t> ciphertext,
00051         std::span<const uint8_t> key,
00052         const core::EncryptionConfig& config
00053     ) override;
00054
00055     // XTS uses double key (two AES keys)
00056     size_t key_size() const override { return (key_bits_ / 8) * 2; }
00057     size_t tweak_size() const { return 16; }  // Tweak/IV size
00058     size_t block_size() const { return 16; }
00059
00060     bool requires_padding() const { return false; }  // Uses ciphertext stealing
00061     bool is_authenticated() const { return false; }
00062     bool is_suitable_for(core::SecurityLevel level) const override;
00063
00064 private:
00065     size_t key_bits_;
00066     core::AlgorithmType type_;
00067     std::string botan_name_;
00068 };
00069
00070 } // namespace symmetric
00071 } // namespace algorithms
00072 } // namespace filevault
00073
00074 #endif
```

10.33 include/filevault/algorithms/symmetric/aria_gcm.hpp File Reference

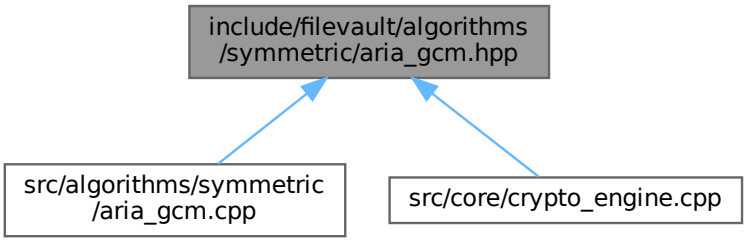
ARIA-GCM AEAD encryption algorithm.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/aead.h>
#include <memory>
```

Include dependency graph for aria_gcm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::ARIA_GCM

  ARIA-GCM AEAD encryption.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.33.1 Detailed Description

ARIA-GCM AEAD encryption algorithm.

ARIA is a Korean block cipher standard, designed by the Korean Agency for Technology and Standards (KATS). It was adopted as a Korean government standard (KS X 1213) and is ISO/IEC 18033-3 certified.

10.34 aria_gcm.hpp

Go to the documentation of this file.

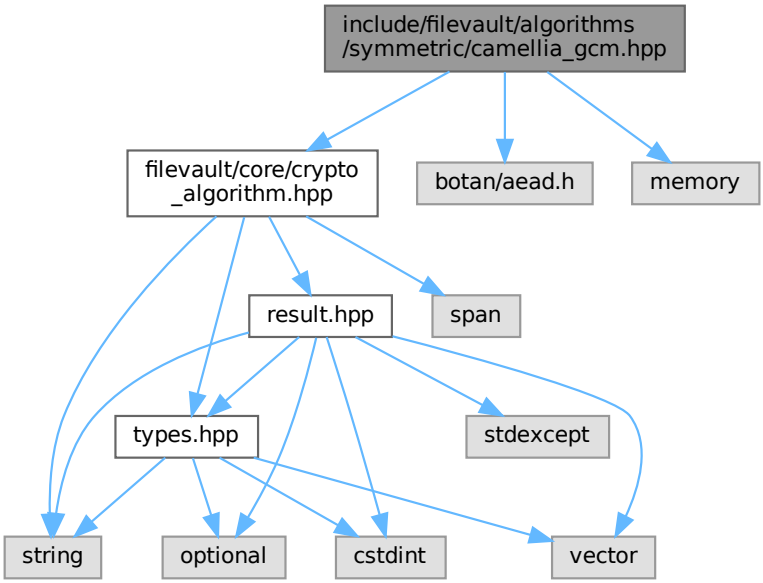
```
00001
00009
00010 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_ARIA_GCM_HPP
00011 #define FILEVAULT_ALGORITHMS_SYMMETRIC_ARIA_GCM_HPP
00012
00013 #include "filevault/core/crypto_algorithm.hpp"
00014 #include <botan/aead.h>
00015 #include <memory>
00016
00017 namespace filevault {
00018 namespace algorithms {
00019 namespace symmetric {
00020
00027 class ARIA_GCM : public core::ICryptoAlgorithm {
00028 public:
00033     explicit ARIA_GCM(size_t key_bits = 256);
00034     virtual ~ARIA_GCM() = default;
00035
00036     std::string name() const override;
00037     core::AlgorithmType type() const override;
00038
00039     core::CryptoResult encrypt(
00040         std::span<const uint8_t> plaintext,
00041         std::span<const uint8_t> key,
00042         const core::EncryptionConfig& config
00043     ) override;
00044
00045     core::CryptoResult decrypt(
00046         std::span<const uint8_t> ciphertext,
00047         std::span<const uint8_t> key,
00048         const core::EncryptionConfig& config
00049     ) override;
00050
00051     size_t key_size() const override { return key_bits_ / 8; }
00052     size_t nonce_size() const { return 12; }  // GCM standard
00053     size_t tag_size() const { return 16; }    // 128-bit tag
00054
00055     bool is_suitable_for(core::SecurityLevel level) const override;
00056
00057 private:
00058     size_t key_bits_;
00059     core::AlgorithmType type_;
00060     std::string botan_name_;
00061 };
00062
00063 } // namespace symmetric
00064 } // namespace algorithms
00065 } // namespace filevault
00066
00067 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_ARIA_GCM_HPP
```

10.35 include/filevault/algorithms/symmetric/camellia_gcm.hpp File Reference

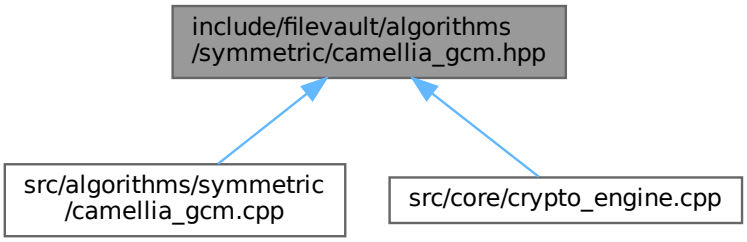
Camellia-GCM AEAD encryption algorithm.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/aead.h>
#include <memory>
```

Include dependency graph for camellia_gcm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::Camellia_GCM

  Camellia-GCM AEAD encryption.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.35.1 Detailed Description

Camellia-GCM AEAD encryption algorithm.

Camellia is a Japanese symmetric key block cipher developed by Mitsubishi Electric and NTT. It's approved by ISO/IEC, CRYPTREC, and NESSIE. It has similar security level to AES.

10.36 camellia_gcm.hpp

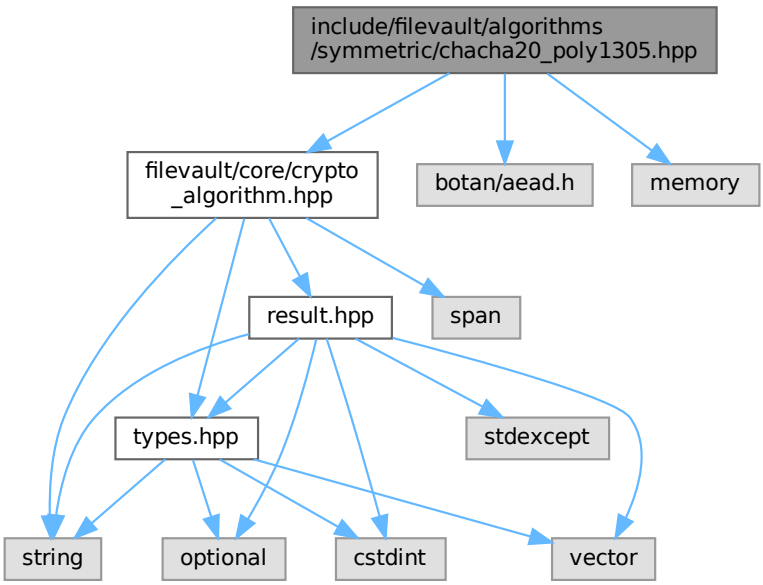
Go to the documentation of this file.

```
00001
00009
00010 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_CAMELLIA_GCM_HPP
00011 #define FILEVAULT_ALGORITHMS_SYMMETRIC_CAMELLIA_GCM_HPP
00012
00013 #include "filevault/core/crypto_algorithm.hpp"
00014 #include <botan/aead.h>
00015 #include <memory>
00016
00017 namespace filevault {
00018 namespace algorithms {
00019 namespace symmetric {
00020
00027 class Camellia_GCM : public core::ICryptoAlgorithm {
00028 public:
00033     explicit Camellia_GCM(size_t key_bits = 256);
00034     virtual ~Camellia_GCM() = default;
00035
00036     std::string name() const override;
00037     core::AlgorithmType type() const override;
00038
00039     core::CryptoResult encrypt(
00040         std::span<const uint8_t> plaintext,
00041         std::span<const uint8_t> key,
00042         const core::EncryptionConfig& config
00043     ) override;
00044
00045     core::CryptoResult decrypt(
00046         std::span<const uint8_t> ciphertext,
00047         std::span<const uint8_t> key,
00048         const core::EncryptionConfig& config
00049     ) override;
00050
00051     size_t key_size() const override { return key_bits_ / 8; }
00052     size_t nonce_size() const { return 12; }  // GCM standard
00053     size_t tag_size() const { return 16; }    // 128-bit tag
00054
00055     bool is_suitable_for(core::SecurityLevel level) const override;
00056
00057 private:
00058     size_t key_bits_;
00059     core::AlgorithmType type_;
00060     std::string botan_name_;
00061 };
00062
00063 } // namespace symmetric
00064 } // namespace algorithms
00065 } // namespace filevault
00066
00067 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_CAMELLIA_GCM_HPP
```

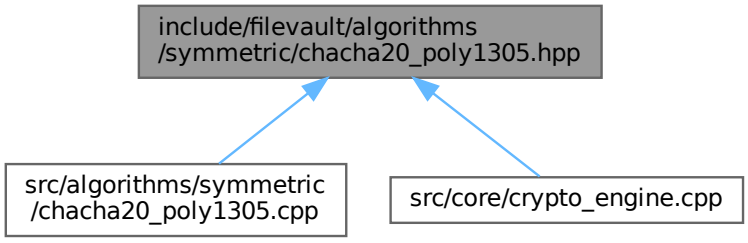
10.37 include/filevault/algorithms/symmetric/chacha20_poly1305.hpp File Reference

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/aead.h>
#include <memory>
```

Include dependency graph for chacha20_poly1305.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::ChaCha20Poly1305

  ChaCha20-Poly1305 AEAD encryption.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.38 chacha20_poly1305.hpp

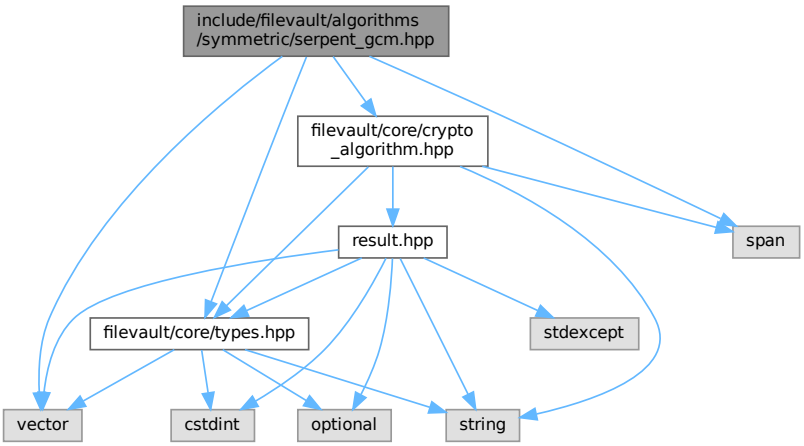
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_CHACHA20_POLY1305_HPP
00002 #define FILEVAULT_ALGORITHMS_SYMMETRIC_CHACHA20_POLY1305_HPP
00003
00004 #include "filevault/core/crypto_algorithm.hpp"
00005 #include <botan/aead.h>
00006 #include <memory>
00007
00008 namespace filevault {
00009 namespace algorithms {
00010 namespace symmetric {
00011
00031 class ChaCha20Poly1305 : public core::ICryptoAlgorithm {
00032 public:
00033     ChaCha20Poly1305();
00034     virtual ~ChaCha20Poly1305() = default;
00035
00036     std::string name() const override;
00037     core::AlgorithmType type() const override;
00038
00039     core::CryptoResult encrypt(
00040         std::span<const uint8_t> plaintext,
00041         std::span<const uint8_t> key,
00042         const core::EncryptionConfig& config
00043     ) override;
00044
00045     core::CryptoResult decrypt(
00046         std::span<const uint8_t> ciphertext,
00047         std::span<const uint8_t> key,
00048         const core::EncryptionConfig& config
00049     ) override;
00050
00051     size_t key_size() const override { return 32; }  // 256 bits
00052     size_t nonce_size() const { return 12; }         // 96 bits (RFC 8439)
00053     size_t tag_size() const { return 16; }           // 128 bits
00054
00055     bool is_suitable_for(core::SecurityLevel level) const override;
00056 };
00057
00058 } // namespace symmetric
00059 } // namespace algorithms
00060 } // namespace filevault
00061
00062 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_CHACHA20_POLY1305_HPP
```

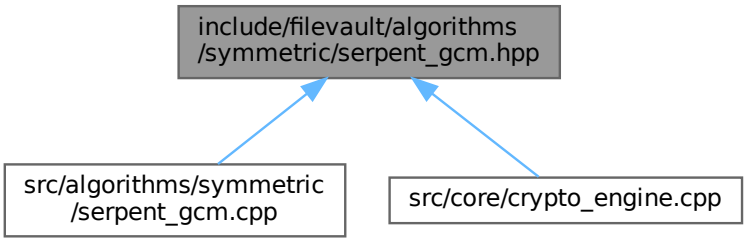
10.39 include/filevault/algorithms/symmetric/serpent_gcm.hpp File Reference

```
#include "filevault/core/types.hpp"
#include "filevault/core/crypto_algorithm.hpp"
#include <vector>
#include <span>
```

Include dependency graph for serpent_gcm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::Serpent_GCM

  Serpent-256-GCM implementation.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.40 serpent_gcm.hpp

Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_SERPENT_GCM_HPP
00002 #define FILEVAULT_ALGORITHMS_SYMMETRIC_SERPENT_GCM_HPP
00003
00004 #include "filevault/core/types.hpp"
00005 #include "filevault/core/crypto_algorithm.hpp"
00006 #include <vector>
```

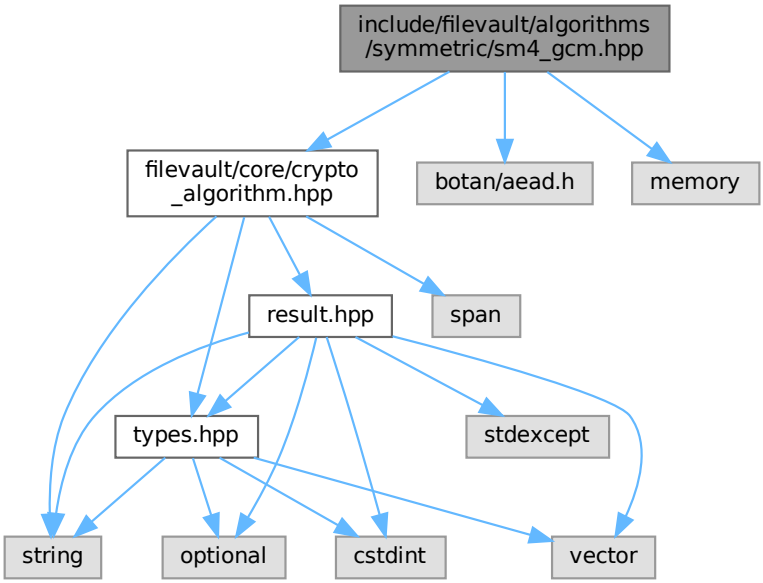
```
00007 #include <span>
00008
00009 namespace filevault {
00010 namespace algorithms {
00011 namespace symmetric {
00012
00036 class Serpent_GCM : public core::ICryptoAlgorithm {
00037 public:
00041     Serpent_GCM();
00042
00046     std::string name() const override;
00047
00051     core::AlgorithmType type() const override;
00052
00061     core::CryptoResult encrypt(
00062         std::span<const uint8_t> plaintext,
00063         std::span<const uint8_t> key,
00064         const core::EncryptionConfig& config
00065     ) override;
00066
00075     core::CryptoResult decrypt(
00076         std::span<const uint8_t> ciphertext,
00077         std::span<const uint8_t> key,
00078         const core::EncryptionConfig& config
00079     ) override;
00080
00084     size_t key_size() const override;
00085
00089     bool is_suitable_for(core::SecurityLevel level) const override;
00090
00091 private:
00102     std::vector<uint8_t> process_gcm(
00103         std::span<const uint8_t> input,
00104         std::span<const uint8_t> key,
00105         std::span<const uint8_t> nonce,
00106         std::span<const uint8_t> tag,
00107         bool encrypt
00108     );
00109 };
00110
00111 } // namespace symmetric
00112 } // namespace algorithms
00113 } // namespace filevault
00114
00115 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_SERPENT_GCM_HPP
```

10.41 include/filevault/algorithms/symmetric/sm4_gcm.hpp File Reference

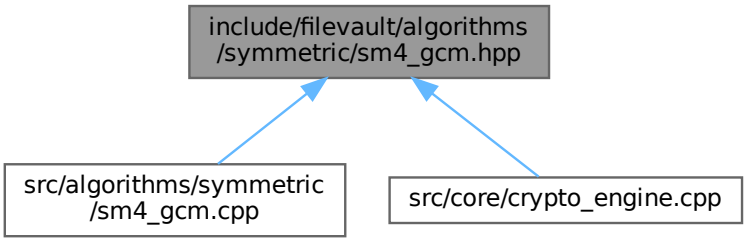
SM4-GCM AEAD encryption algorithm.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/aead.h>
#include <memory>
```

Include dependency graph for sm4_gcm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::SM4_GCM

  SM4-GCM AEAD encryption.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.41.1 Detailed Description

SM4-GCM AEAD encryption algorithm.

SM4 (formerly SMS4) is a Chinese national standard block cipher used in wireless LAN standard WAPI. It was made a national standard (GB/T 32907-2016) in 2016 and is widely used in China. SM4 uses 128-bit key and 128-bit block size.

10.42 sm4_gcm.hpp

Go to the documentation of this file.

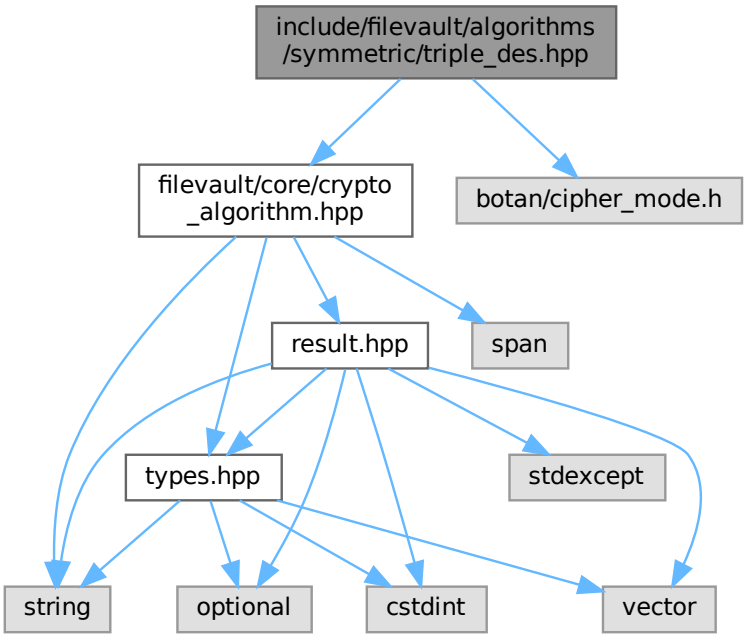
```
00001
00010
00011 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_SM4_GCM_HPP
00012 #define FILEVAULT_ALGORITHMS_SYMMETRIC_SM4_GCM_HPP
00013
00014 #include "filevault/core/crypto_algorithm.hpp"
00015 #include <botan/aead.h>
00016 #include <memory>
00017
00018 namespace filevault {
00019 namespace algorithms {
00020 namespace symmetric {
00021
00028 class SM4_GCM : public core::ICryptoAlgorithm {
00029 public:
00030     SM4_GCM();
00031     virtual ~SM4_GCM() = default;
00032
00033     std::string name() const override;
00034     core::AlgorithmType type() const override;
00035
00036     core::CryptoResult encrypt(
00037         std::span<const uint8_t> plaintext,
00038         std::span<const uint8_t> key,
00039         const core::EncryptionConfig& config
00040     ) override;
00041
00042     core::CryptoResult decrypt(
00043         std::span<const uint8_t> ciphertext,
00044         std::span<const uint8_t> key,
00045         const core::EncryptionConfig& config
00046     ) override;
00047
00048     size_t key_size() const override { return 16; }  // 128 bits only
00049     size_t nonce_size() const { return 12; }  // GCM standard
00050     size_t tag_size() const { return 16; }    // 128-bit tag
00051
00052     bool is_suitable_for(core::SecurityLevel level) const override;
00053 };
00054
00055 } // namespace symmetric
00056 } // namespace algorithms
00057 } // namespace filevault
00058
00059 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_SM4_GCM_HPP
```

10.43 include/filevault/algorithms/symmetric/triple_des.hpp File Reference

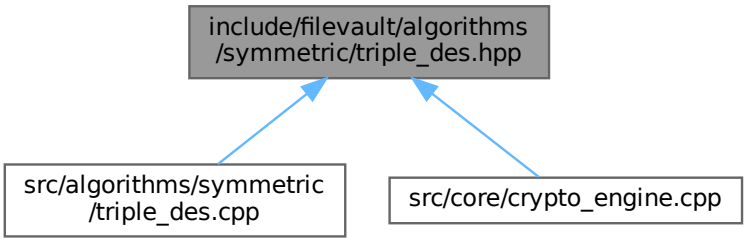
Triple-DES (3DES) encryption implementation.

```
#include "filevault/core/crypto_algorithm.hpp"
#include <botan/cipher_mode.h>
```

Include dependency graph for triple_des.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::TripleDES

  Triple-DES (3DES/TDEA) encryption.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.43.1 Detailed Description

Triple-DES (3DES) encryption implementation.
Triple-DES provides 112-bit effective security (168-bit key). Legacy algorithm - use AES for new applications.

10.44 triple_des.hpp

Go to the documentation of this file.

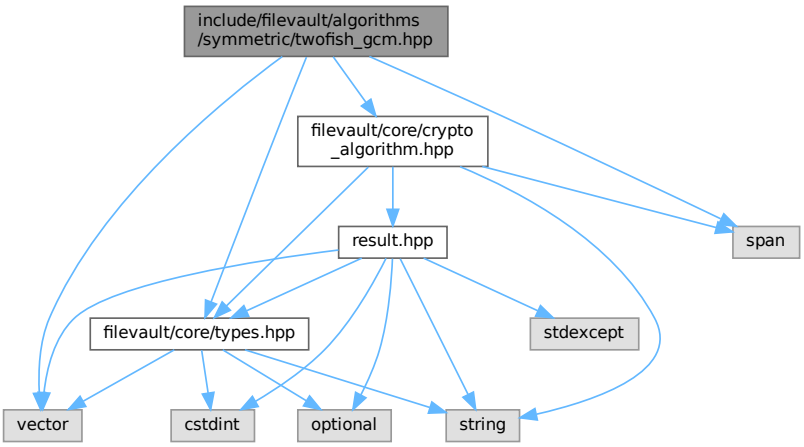
```
00001
00008
00009 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_TRIPLE_DES_HPP
00010 #define FILEVAULT_ALGORITHMS_SYMMETRIC_TRIPLE_DES_HPP
00011
00012 #include "filevault/core/crypto_algorithm.hpp"
00013 #include <botan/cipher_mode.h>
00014
00015 namespace filevault {
00016 namespace algorithms {
00017 namespace symmetric {
00018
00030 class TripleDES : public core::ICryptoAlgorithm {
00031 public:
00032     explicit TripleDES();
00033     virtual ~TripleDES() = default;
00034
00035     std::string name() const override { return "3DES-CBC"; }
00036     core::AlgorithmType type() const override { return core::AlgorithmType::TRIPLE_DES_CBC; }
00037
00038     core::CryptoResult encrypt(
00039         std::span<const uint8_t> plaintext,
00040         std::span<const uint8_t> key,
00041         const core::EncryptionConfig& config
00042     ) override;
00043
00044     core::CryptoResult decrypt(
00045         std::span<const uint8_t> ciphertext,
00046         std::span<const uint8_t> key,
00047         const core::EncryptionConfig& config
00048     ) override;
00049
00050     size_t key_size() const override { return 24; }  // 168 bits (3 x 56 bits)
00051     size_t iv_size() const { return 8; }  // 64-bit block
00052     size_t block_size() const { return 8; }
00053
00054     bool is_suitable_for(core::SecurityLevel level) const override;
00055 };
00056
00057 } // namespace symmetric
00058 } // namespace algorithms
00059 } // namespace filevault
00060
00061 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_TRIPLE_DES_HPP
```

10.45 include/filevault/algorithms/symmetric/twofish_gcm.hpp File Reference

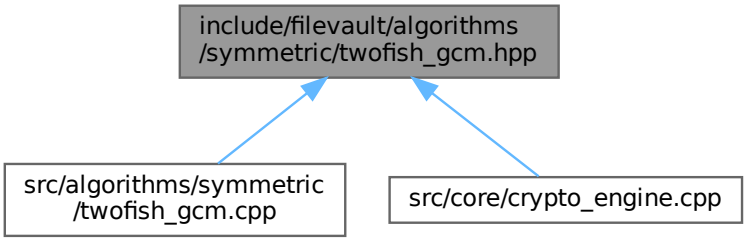
Twofish-256-GCM encryption algorithm.

```
#include "filevault/core/types.hpp"
#include "filevault/core/crypto_algorithm.hpp"
#include <vector>
#include <span>
```

Include dependency graph for twofish_gcm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::algorithms::symmetric::Twofish_GCM

  Twofish-256-GCM implementation.

Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.45.1 Detailed Description

Twofish-256-GCM encryption algorithm.
Twofish is a 128-bit block cipher designed by Bruce Schneier et al. It was an AES finalist and is known for its flexibility and security.

Author

FileVault Team

Date

2024

10.46 twofish_gcm.hpp

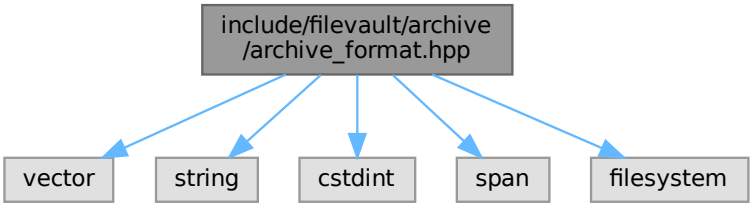
Go to the documentation of this file.

```
00001
00011
00012 #ifndef FILEVAULT_ALGORITHMS_SYMMETRIC_TWOFISH_GCM_HPP
00013 #define FILEVAULT_ALGORITHMS_SYMMETRIC_TWOFISH_GCM_HPP
00014
00015 #include "filevault/core/types.hpp"
00016 #include "filevault/core/crypto_algorithm.hpp"
00017 #include <vector>
00018 #include <span>
00019
00020 namespace filevault {
00021 namespace algorithms {
00022 namespace symmetric {
00023
00053 class Twofish_GCM : public core::ICryptoAlgorithm {
00054 public:
00059     explicit Twofish_GCM(size_t key_bits = 256);
00060
00064     std::string name() const override;
00065
00069     core::AlgorithmType type() const override;
00070
00079     core::CryptoResult encrypt(
00080         std::span<const uint8_t> plaintext,
00081         std::span<const uint8_t> key,
00082         const core::EncryptionConfig& config
00083     ) override;
00084
00093     core::CryptoResult decrypt(
00094         std::span<const uint8_t> ciphertext,
00095         std::span<const uint8_t> key,
00096         const core::EncryptionConfig& config
00097     ) override;
00098
00102     size_t key_size() const override;
00103
00107     size_t nonce_size() const { return 12; }
00108
00112     size_t tag_size() const { return 16; }
00113
00117     bool is_suitable_for(core::SecurityLevel level) const override;
00118
00119 private:
00120     size_t key_bits_;
00121     core::AlgorithmType type_;
00122     std::string botan_name_;
00123
00127     std::vector<uint8_t> process_gcm(
00128         std::span<const uint8_t> input,
00129         std::span<const uint8_t> key,
00130         std::span<const uint8_t> nonce,
00131         std::span<const uint8_t> tag,
00132         bool encrypt
00133     );
00134 };
00135
00136 } // namespace symmetric
00137 } // namespace algorithms
00138 } // namespace filevault
00139
00140 #endif // FILEVAULT_ALGORITHMS_SYMMETRIC_TWOFISH_GCM_HPP
```

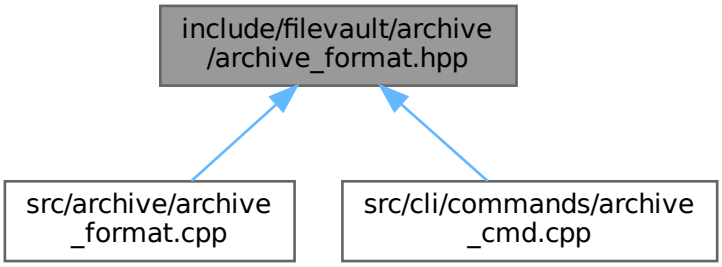
10.47 include/filevault/archive/archive_format.hpp File Reference

```
#include <vector>
#include <string>
#include <cstdint>
#include <span>
#include <filesystem>
```

Include dependency graph for archive_format.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::archive::FileEntry

  Archive file entry metadata.
- class filevault::archive::ArchiveFormat

  Simple archive format handler.

Namespaces

- namespace filevault
- namespace filevault::archive

10.48 archive_format.hpp

Go to the documentation of this file.

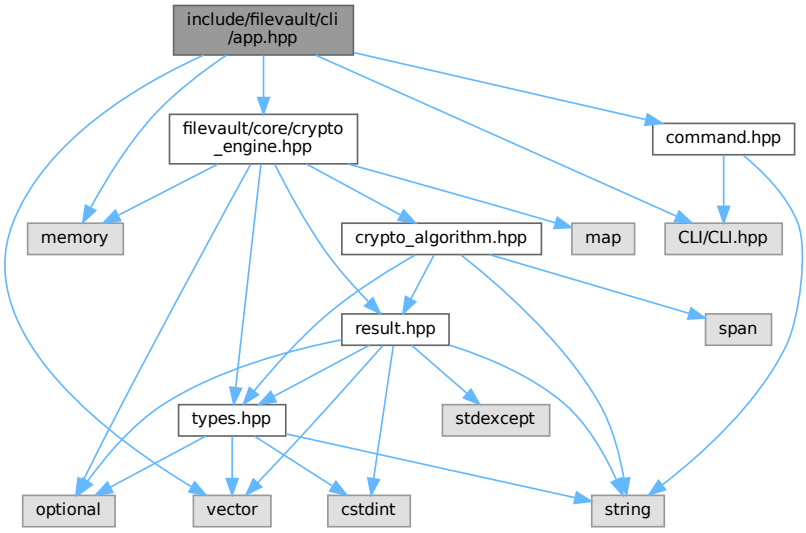
```
00001 #ifndef FILEVAULT_ARCHIVE_FORMAT_HPP
00002 #define FILEVAULT_ARCHIVE_FORMAT_HPP
00003
00004 #include <vector>
00005 #include <string>
00006 #include <cstdint>
00007 #include <span>
00008 #include <filesystem>
00009
00010 namespace filevault::archive {
00011
```

```
00015 struct FileEntry {
00016     std::string filename;
00017     uint64_t file_size;
00018     uint64_t offset;           // Offset in data section
00019     uint64_t modified_time;    // Unix timestamp
00020     uint32_t permissions;      // File permissions
00021
00022     std::vector<uint8_t> serialize() const;
00023     static FileEntry deserialize(std::span<const uint8_t> data, size_t& offset);
00024 };
00025
00034 class ArchiveFormat {
00035 public:
00036     static constexpr char MAGIC[7] = "FVARCH";
00037     static constexpr uint8_t VERSION = 1;
00038
00042     static std::vector<uint8_t> create_archive(
00043         const std::vector<std::filesystem::path>& files
00044     );
00045
00049     static bool extract_archive(
00050         std::span<const uint8_t> archive_data,
00051         const std::filesystem::path& output_dir
00052     );
00053
00057     static std::vector<FileEntry> list_files(
00058         std::span<const uint8_t> archive_data
00059     );
00060
00061 private:
00062     static void write_uint32(std::vector<uint8_t>& buffer, uint32_t value);
00063     static void write_uint64(std::vector<uint8_t>& buffer, uint64_t value);
00064     static uint32_t read_uint32(std::span<const uint8_t> data, size_t& offset);
00065     static uint64_t read_uint64(std::span<const uint8_t> data, size_t& offset);
00066     static std::string read_string(std::span<const uint8_t> data, size_t& offset);
00067 };
00068
00069 } // namespace filevault::archive
00070
00071 #endif // FILEVAULT_ARCHIVE_FORMAT_HPP
```

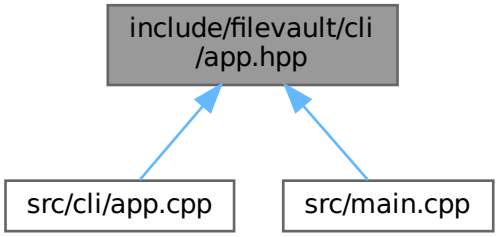
10.49 include/filevault/cli/app.hpp File Reference

```
#include <memory>
#include <vector>
#include <CLI/CLI.hpp>
#include "command.hpp"
#include "filevault/core/crypto_engine.hpp"
```

Include dependency graph for app.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::Application

  Main CLI application.

Namespaces

- namespace filevault
- namespace filevault::cli

10.50 app.hpp

Go to the documentation of this file.

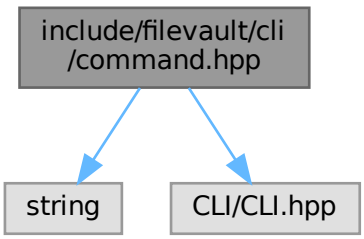
```
00001 #ifndef FILEVAULT_CLI_APP_HPP
00002 #define FILEVAULT_CLI_APP_HPP
00003
```

```
00004 #include <memory>
00005 #include <vector>
00006 #include <CLI/CLI.hpp>
00007 #include "command.hpp"
00008 #include "filevault/core/crypto_engine.hpp"
00009
00010 namespace filevault {
00011 namespace cli {
00012
00016 class Application {
00017 public:
00018     Application();
00019     ~Application();
00020
00024     void initialize();
00025
00029     int run(int argc, char** argv);
00030
00031 private:
00032     void register_commands();
00033     void setup_logging();
00034
00035     CLI::App app_;
00036     std::unique_ptr<core::CryptoEngine> engine_;
00037     std::vector<std::unique_ptr<ICommand>> commands_;
00038
00039     // Global options
00040     bool verbose_ = false;
00041     std::string log_level_ = "info";
00042 };
00043
00044 } // namespace cli
00045 } // namespace filevault
00046
00047 #endif // FILEVAULT_CLI_APP_HPP
```

10.51 include/filevault/cli/command.hpp File Reference

```
#include <string>
#include <CLI/CLI.hpp>
```

Include dependency graph for command.hpp:



This graph shows which files directly or indirectly include this file:

Classes

- class filevault::cli::ICommand

  Base interface for CLI commands.

Namespaces

- namespace filevault
- namespace filevault::cli

10.52 command.hpp

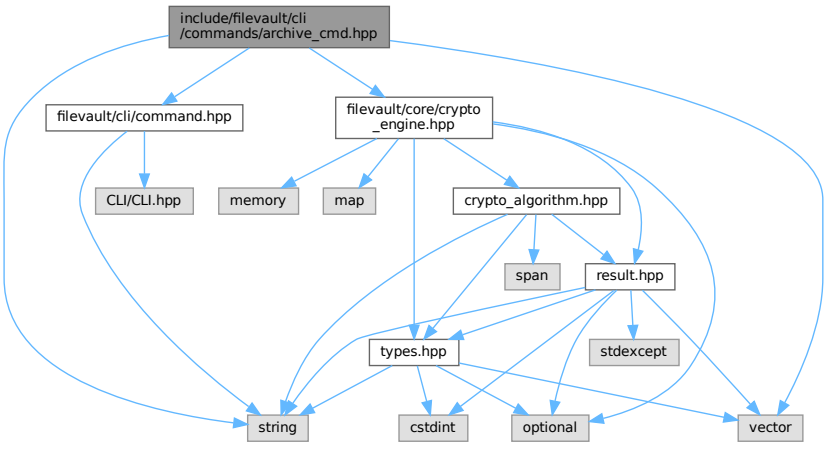
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMAND_HPP
00002 #define FILEVAULT_CLI_COMMAND_HPP
00003
00004 #include <string>
00005 #include <CLI/CLI.hpp>
00006
00007 namespace filevault {
00008 namespace cli {
00009
00013 class ICommand {
00014 public:
00015     virtual ~ICommand() = default;
00016
00020     virtual std::string name() const = 0;
00021
00025     virtual std::string description() const = 0;
00026
00030     virtual void setup(CLI::App& app) = 0;
00031
00035     virtual int execute() = 0;
00036 };
00037
00038 } // namespace cli
00039 } // namespace filevault
00040
00041 #endif // FILEVAULT_CLI_COMMAND_HPP
```

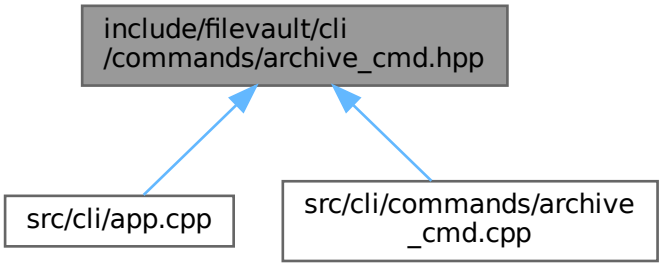
10.53 include/filevault/cli/commands/archive_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
#include <string>
#include <vector>
```

Include dependency graph for archive_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::commands::ArchiveCommand

  Archive command - compress and encrypt multiple files.

Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.54 archive_cmd.hpp

Go to the documentation of this file.

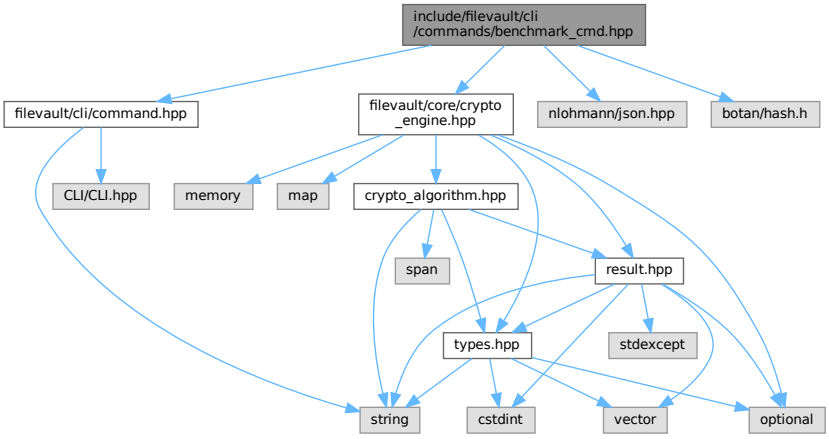
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_ARCHIVE_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_ARCHIVE_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006 #include <string>
00007 #include <vector>
00008
00009 namespace filevault::cli::commands {
00010
00017 class ArchiveCommand : public cli::ICommand {
00018 public:
00019     explicit ArchiveCommand(core::CryptoEngine& engine)
00020         : engine_(engine) {}
00021
00022     std::string name() const override { return "archive"; }
00023
00024     std::string description() const override {
00025         return "Create encrypted archive from multiple files";
00026     }
00027
00028     void setup(CLI::App& app) override;
00029     int execute() override;
00030
00031 private:
00032     int do_create();
00033     int do_extract();
00034
00035     core::CryptoEngine& engine_;
00036
00037     // Options
00038     std::vector<std::string> input_files_;
00039     std::string output_file_;
00040     std::string password_;
00041     std::string algorithm_ = "aes-256-gcm";
00042     std::string compression_ = "zlib";
00043     std::string kdf_ = "argon2id";
```

```
00044     std::string security_level_ = "medium";
00045     bool extract_ = false;
00046     bool verbose_ = false;
00047     std::string extract_dir_ = ".";
00048 };
00049
00050 } // namespace filevault::cli::commands
00051
00052 #endif // FILEVAULT_CLI_COMMANDS_ARCHIVE_CMD_HPP
```

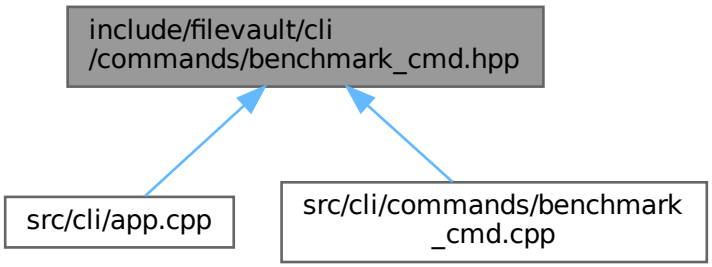
10.55 include/filevault/cli/commands/benchmark_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
#include <nlohmann/json.hpp>
#include <botan/hash.h>
```

Include dependency graph for benchmark_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::cli::BenchmarkResult

- struct filevault::cli::AsymmetricBenchmarkResult
- struct filevault::cli::PQCBenchmarkResult
- struct filevault::cli::SignatureBenchmarkResult
- class filevault::cli::BenchmarkCommand

Namespaces

- namespace filevault
- namespace filevault::cli

10.56 benchmark_cmd.hpp

Go to the documentation of this file.

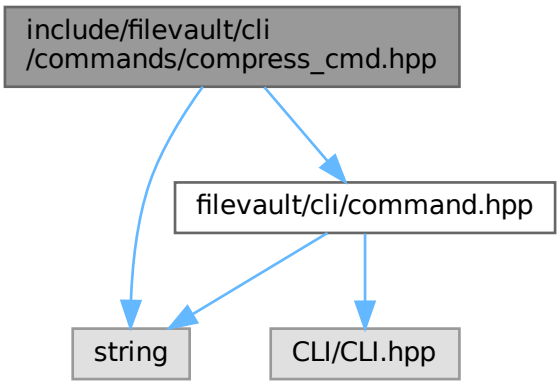
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_BENCHMARK_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_BENCHMARK_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006 #include <nlohmann/json.hpp>
00007 #include <botan/hash.h>
00008
00009 namespace filevault {
00010 namespace cli {
00011
00012 struct BenchmarkResult {
00013     std::string algorithm;
00014     double encrypt_ms = 0;
00015     double decrypt_ms = 0;
00016     double encrypt_mbps = 0;
00017     double decrypt_mbps = 0;
00018     bool success = false;
00019 };
00020
00021 struct AsymmetricBenchmarkResult {
00022     std::string algorithm;
00023     double keygen_ms = 0;
00024     double encrypt_ms = 0;
00025     double decrypt_ms = 0;
00026     bool success = false;
00027 };
00028
00029 struct PQCBenchmarkResult {
00030     std::string algorithm;
00031     double keygen_ms = 0;
00032     double encaps_ms = 0;
00033     double decaps_ms = 0;
00034     bool success = false;
00035 };
00036
00037 struct SignatureBenchmarkResult {
00038     std::string algorithm;
00039     double keygen_ms = 0;
00040     double sign_ms = 0;
00041     double verify_ms = 0;
00042     bool success = false;
00043 };
00044
00045 class BenchmarkCommand : public ICommand {
00046 public:
00047     explicit BenchmarkCommand(core::CryptoEngine& engine);
00048
00049     std::string name() const override { return "benchmark"; }
00050     std::string description() const override { return "Benchmark all cryptographic algorithms"; }
00051
00052     void setup(CLI::App& app) override;
00053     int execute() override;
00054
00055 private:
00056     // Benchmark functions
00057     void benchmark_symmetric(nlohmann::json& json_results);
00058     void benchmark_asymmetric(nlohmann::json& json_results);
00059     void benchmark_pqc(nlohmann::json& json_results);
00060     void benchmark_kdf(nlohmann::json& json_results);
00061     void benchmark_compression(nlohmann::json& json_results);
00062     void benchmark_hash(nlohmann::json& json_results);
00063
00064     // Algorithm-specific benchmarks
00065     BenchmarkResult benchmark_algorithm(core::AlgorithmType algo_type);
00066     AsymmetricBenchmarkResult benchmark_asymmetric_algorithm(core::AlgorithmType algo_type);
```

```
00067     PQCBenchmarkResult benchmark_pqc_algorithm(core::AlgorithmType algo_type);
00068     BenchmarkResult benchmark_hybrid_algorithm(core::AlgorithmType algo_type);
00069     SignatureBenchmarkResult benchmark_signature_algorithm(core::AlgorithmType algo_type);
00070
00071     // Helpers
00072     std::string get_platform_info();
00073     void save_json_output(const nlohmann::json& results);
00074     void save_log_output(const std::string& log_content);
00075
00076     core::CryptoEngine& engine_;
00077     std::string algorithm_;
00078     std::string output_file_;
00079     size_t data_size_ = 1048576;  // 1MB default
00080     int iterations_ = 5;
00081     bool all_ = false;
00082     bool json_output_ = false;
00083     bool pqc_only_ = false;
00084     bool symmetric_only_ = false;
00085     bool asymmetric_only_ = false;
00086     bool hash_only_ = false;
00087     bool kdf_only_ = false;
00088     bool compression_only_ = false;
00089 };
00090
00091 } // namespace cli
00092 } // namespace filevault
00093
00094 #endif
```

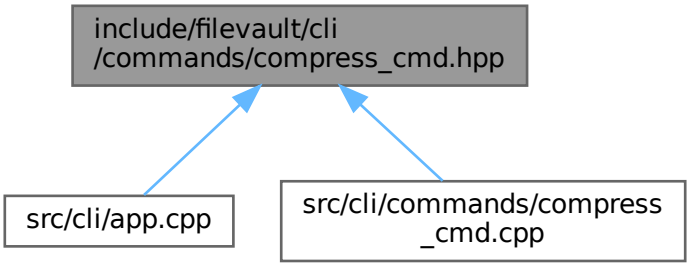
10.57 include/filevault/cli/commands/compress_cmd.hpp File Reference

```
#include <string>
#include "filevault/cli/command.hpp"
```

Include dependency graph for compress_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::CompressCommand

  Compress command - standalone compression/decompression.

Namespaces

- namespace filevault
- namespace filevault::cli

10.58 compress_cmd.hpp

Go to the documentation of this file.

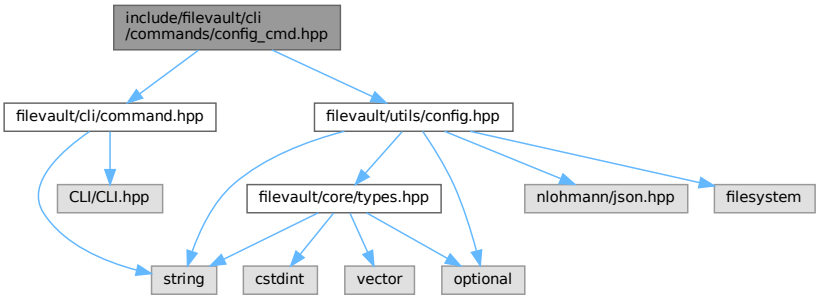
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_COMPRESS_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_COMPRESS_CMD_HPP
00003
00004 #include <string>
00005 #include "filevault/cli/command.hpp"
00006
00007 namespace filevault::cli {
00008
00019 class CompressCommand : public ICommand {
00020 public:
00021     CompressCommand() = default;
00022     ~CompressCommand() override = default;
00023
00024     std::string name() const override { return "compress"; }
00025     std::string description() const override { return "Compress or decompress files"; }
00026     void setup(CLI::App& app) override;
00027     int execute() override;
00028 private:
00029     CLI::App* subcommand_ = nullptr;
00030     // Input/output files
00031     std::string input_file_;
00032     std::string output_file_;
00033
00034     // Compression options
00035     std::string algorithm_ = "zlib";     // zlib, bzip2, lzma
00036     int level_ = 6;                       // 1-9
00037     bool decompress_ = false;             // Decompress mode
00038     bool verbose_ = false;
00039     bool benchmark_ = false;              // Show timing info
00040     bool auto_detect_ = false;            // Auto-detect algorithm
00041
00042     // Helper methods
00043     int do_compress();
00044     int do_decompress();
00045     std::string detect_algorithm(const std::string& path);
00046     std::string generate_output_path(const std::string& input, bool compressing);
00047 };
00048
```

```
00049 } // namespace filevault::cli
00050
00051 #endif // FILEVAULT_CLI_COMMANDS_COMPRESS_CMD_HPP
```

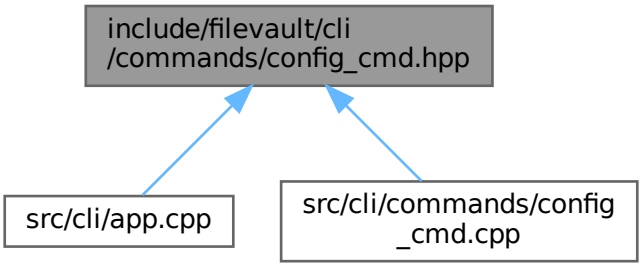
10.59 include/filevault/cli/commands/config_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/utils/config.hpp"
```

Include dependency graph for config_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::ConfigCommand

  Config command - manage FileVault configuration.

Namespaces

- namespace filevault
- namespace filevault::cli

10.60 config_cmd.hpp

Go to the documentation of this file.

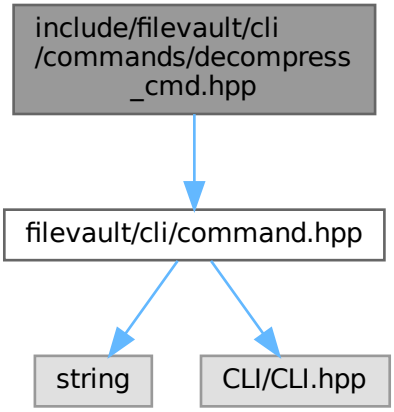
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_CONFIG_CMD_HPP
```

```
00002 #define FILEVAULT_CLI_COMMANDS_CONFIG_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/utils/config.hpp"
00006
00007 namespace filevault {
00008 namespace cli {
00009
00013 class ConfigCommand : public ICommand {
00014 public:
00015     ConfigCommand() = default;
00016
00017     std::string name() const override { return "config"; }
00018     std::string description() const override { return "Manage FileVault configuration"; }
00019
00020     void setup(CLI::App& app) override;
00021     int execute() override;
00022
00023 private:
00024     int show_config();
00025     int set_value();
00026     int reset_config();
00027
00028     std::string subcommand_;
00029     std::string key_;
00030     std::string value_;
00031 };
00032
00033 } // namespace cli
00034 } // namespace filevault
00035
00036 #endif // FILEVAULT_CLI_COMMANDS_CONFIG_CMD_HPP
```

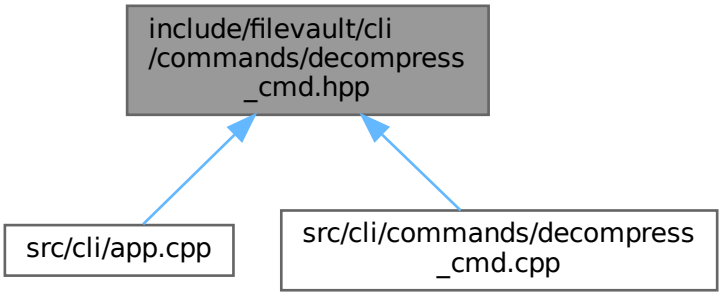
10.61 include/filevault/cli/commands/decompress_cmd.hpp File Reference

`#include "filevault/cli/command.hpp"`

Include dependency graph for decompress_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::DecompressCommand

  Decompress command - alias for compress -d.

Namespaces

- namespace filevault
- namespace filevault::cli

10.62 decompress_cmd.hpp

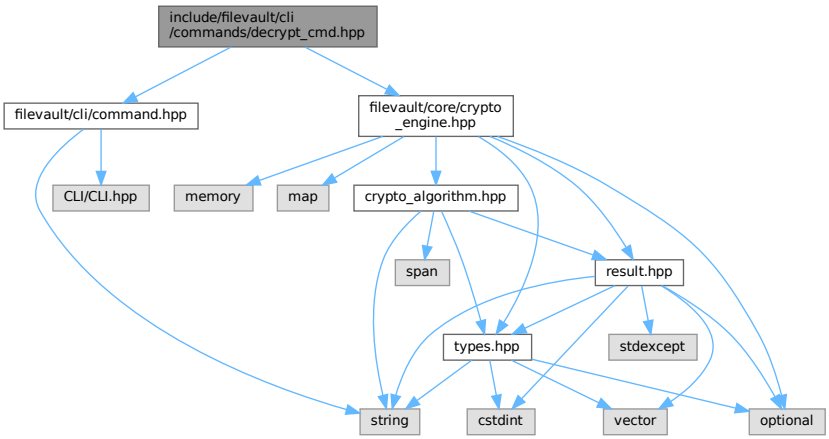
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMANDS_DECOMPRESS_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_DECOMPRESS_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005
00006 namespace filevault::cli {
00007
00013 class DecompressCommand : public ICommand {
00014 public:
00015     std::string name() const override { return "decompress"; }
00016     std::string description() const override { return "Decompress a compressed file"; }
00017
00018     void setup(CLI::App& app) override;
00019     int execute() override;
00020
00021 private:
00022     std::string detect_algorithm(const std::string& path);
00023     std::string generate_output_path(const std::string& input);
00024
00025     CLI::App* subcommand_ = nullptr;
00026     std::string input_file_;
00027     std::string output_file_;
00028     std::string algorithm_;
00029     bool auto_detect_ = true;  // Auto-detect by default for decompress
00030     bool verbose_ = false;
00031     bool benchmark_ = false;
00032 };
00033
00034 } // namespace filevault::cli
00035
00036 #endif
```

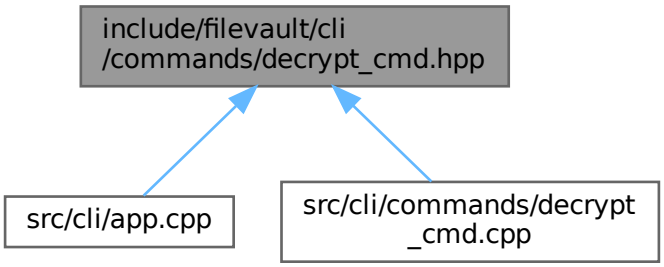
10.63 include/filevault/cli/commands/decrypt_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
```

Include dependency graph for decrypt_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::DecryptCommand

Namespaces

- namespace filevault
- namespace filevault::cli

10.64 decrypt_cmd.hpp

Go to the documentation of this file.

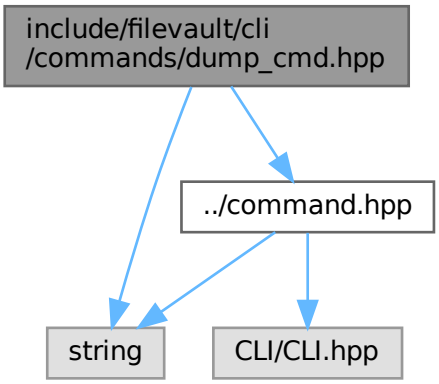
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_DECRYPT_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_DECRYPT_CMD_HPP
00003
```

```
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006
00007 namespace filevault {
00008 namespace cli {
00009
00010 class DecryptCommand : public ICommand {
00011 public:
00012     explicit DecryptCommand(core::CryptoEngine& engine);
00013
00014     std::string name() const override { return "decrypt"; }
00015     std::string description() const override { return "Decrypt a file"; }
00016
00017     void setup(CLI::App& app) override;
00018     int execute() override;
00019
00020 private:
00021     core::CryptoEngine& engine_;
00022     std::string input_file_;
00023     std::string output_file_;
00024     std::string password_;
00025     bool verbose_ = false;
00026     bool no_progress_ = false;
00027 };
00028
00029 } // namespace cli
00030 } // namespace filevault
00031
00032 #endif
```

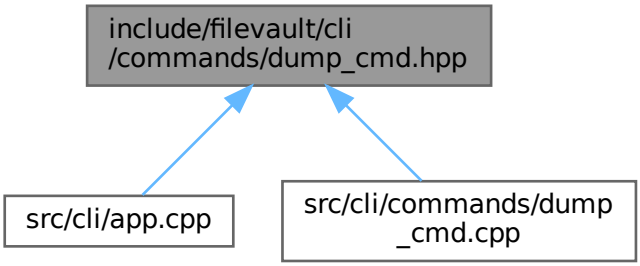
10.65 include/filevault/cli/commands/dump_cmd.hpp File Reference

```
#include "../command.hpp"
#include <string>
```

Include dependency graph for dump_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::commands::DumpCommand

  Command to dump file content in various formats (hex, binary, base64).

Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.66 dump_cmd.hpp

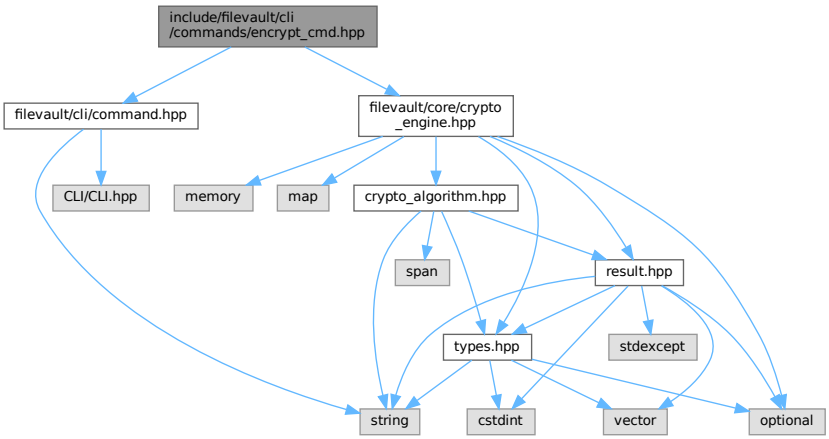
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMANDS_DUMP_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_DUMP_CMD_HPP
00003
00004 #include "../command.hpp"
00005 #include <string>
00006
00007 namespace filevault {
00008 namespace cli {
00009 namespace commands {
00010
00014 class DumpCommand : public ICommand {
00015 public:
00016     DumpCommand();
00017
00018     std::string name() const override { return name_; }
00019     std::string description() const override { return description_; }
00020
00021     void setup(CLI::App& app) override;
00022     int execute() override;
00023
00024 private:
00025     std::string name_ = "dump";
00026     std::string description_ = "View file content in hex, binary, or base64 format";
00027
00028     std::string file_path_;
00029     std::string format_ = "hex";  // Default format
00030     size_t max_bytes_ = 0;  // 0 = show all
00031     bool show_offset_ = true;
00032     bool show_ascii_ = true;
00033 };
00034
00035 } // namespace commands
00036 } // namespace cli
00037 } // namespace filevault
00038
00039 #endif // FILEVAULT_CLI_COMMANDS_DUMP_CMD_HPP
```

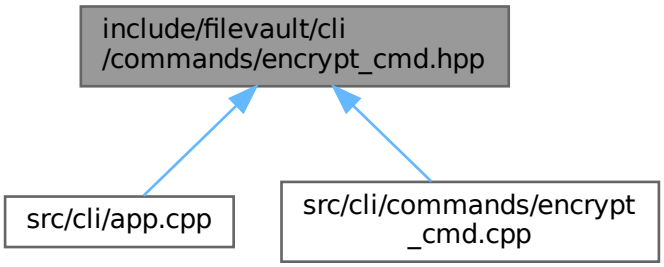
10.67 include/filevault/cli/commands/encrypt_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
```

Include dependency graph for encrypt_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::EncryptCommand

  Encrypt command.

Namespaces

- namespace filevault
- namespace filevault::cli

10.68 encrypt_cmd.hpp

Go to the documentation of this file.

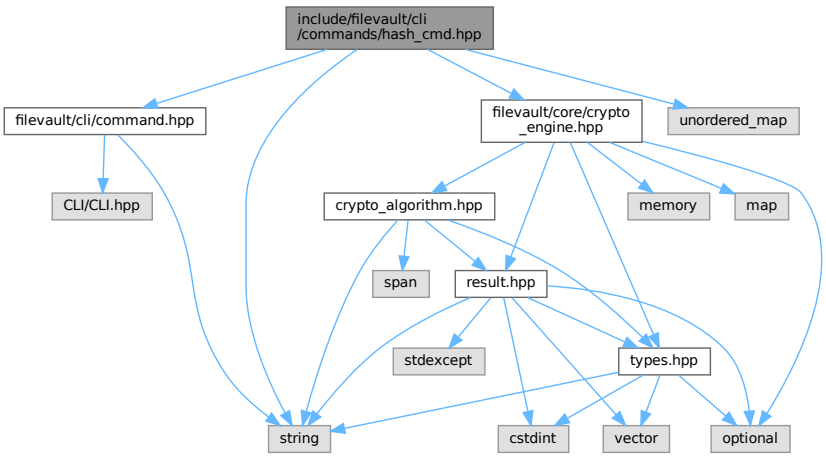
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_ENCRYPT_CMD_HPP
```

```
00002 #define FILEVAULT_CLI_COMMANDS_ENCRYPT_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006
00007 namespace filevault {
00008 namespace cli {
00009
00013 class EncryptCommand : public ICommand {
00014 public:
00015     explicit EncryptCommand(core::CryptoEngine& engine);
00016
00017     std::string name() const override { return "encrypt"; }
00018     std::string description() const override { return "Encrypt a file"; }
00019
00020     void setup(CLI::App& app) override;
00021     int execute() override;
00022
00023 private:
00024     core::CryptoEngine& engine_;
00025
00026     // Command options
00027     std::string input_file_;
00028     std::string output_file_;
00029     std::string password_;
00030     std::string mode_;  // Mode preset: basic/standard/advanced
00031     std::string algorithm_ = "aes-256-gcm";
00032     std::string security_level_ = "medium";
00033     std::string kdf_ = "argon2id";
00034     std::string compression_type_ = "none";
00035     int compression_level_ = 6;
00036     bool verbose_ = false;
00037     bool no_progress_ = false;
00038     bool force_weak_password_ = false;
00039 };
00040
00041 } // namespace cli
00042 } // namespace filevault
00043
00044 #endif // FILEVAULT_CLI_COMMANDS_ENCRYPT_CMD_HPP
```

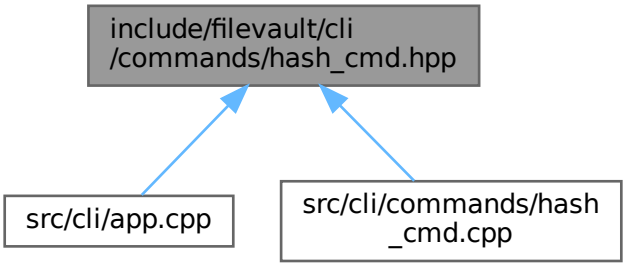
10.69 include/filevault/cli/commands/hash_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
#include <string>
#include <unordered_map>
```

Include dependency graph for hash_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::HashCommand

  Hash command - Calculate cryptographic hashes.

Namespaces

- namespace filevault
- namespace filevault::cli

10.70 hash_cmd.hpp

Go to the documentation of this file.

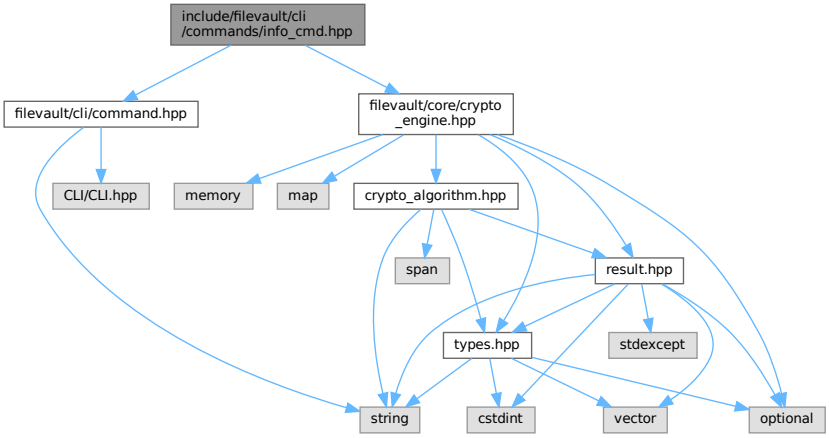
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_HASH_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_HASH_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006 #include <string>
00007 #include <unordered_map>
00008
00009 namespace filevault {
00010 namespace cli {
00011
00024 class HashCommand : public ICommand {
00025 public:
00026     explicit HashCommand(core::CryptoEngine& engine);
00027
00028     std::string name() const override { return "hash"; }
00029     std::string description() const override {
00030         return "Calculate cryptographic hash of files (MD5, SHA1-3, BLAKE2)";
00031     }
00032
00033     void setup(CLI::App& app) override;
00034     int execute() override;
00035
00036 private:
00037     core::CryptoEngine& engine_;
00038
00039     // Options
00040     std::string input_file_;
00041     std::string output_file_;
00042     std::string algorithm_ = "sha256";
00043     std::string output_format_ = "hex";
00044     std::string verify_hash_;
00045     std::string hmac_key_;
00046     bool uppercase_ = false;
00047     bool no_filename_ = false;
00048     bool verbose_ = false;
00049     bool benchmark_ = false;
00050
```

```
00051     // Helper methods
00052     std::string get_botan_algorithm_name(const std::string& algo);
00053     bool is_secure_algorithm(const std::string& algo);
00054
00055     std::string calculate_file_hash(
00056         const std::string& filepath,
00057         const std::string& algorithm
00058     );
00059
00060     std::string calculate_file_hmac(
00061         const std::string& filepath,
00062         const std::string& hash_algorithm,
00063         const std::string& key
00064     );
00065
00066     int verify_mode(const std::string& calculated_hash);
00067 };
00068
00069 } // namespace cli
00070 } // namespace filevault
00071
00072 #endif // FILEVAULT_CLI_COMMANDS_HASH_CMD_HPP
```

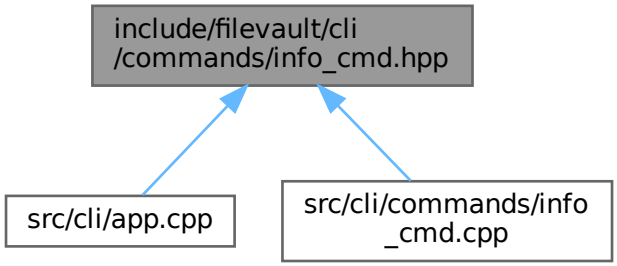
10.71 include/filevault/cli/commands/info_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
```

Include dependency graph for info_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::InfoCommand

  Info command - display encrypted file metadata.
- struct filevault::cli::InfoCommand::FileInfo

  Parse encrypted file header.

Namespaces

- namespace filevault
- namespace filevault::cli

10.72 info_cmd.hpp

Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMANDS_INFO_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_INFO_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006
00007 namespace filevault {
00008 namespace cli {
00009
00020 class InfoCommand : public ICommand {
00021 public:
00022     explicit InfoCommand(core::CryptoEngine& engine);
00023
00024     std::string name() const override { return "info"; }
00025     std::string description() const override {
00026         return "Display information about encrypted file";
00027     }
00028
00029     void setup(CLI::App& app) override;
00030     int execute() override;
00031
00032 private:
00033     core::CryptoEngine& engine_;
00034     std::string input_file_;
00035     bool verbose_ = false;
00036
00040     struct FileInfo {
00041         size_t file_size = 0;
00042         size_t salt_size = 0;
00043         size_t nonce_size = 0;
00044         size_t tag_size = 0;
00045         size_t data_size = 0;
00046         size_t header_size = 0;
00047
00048         // Enhanced format fields
```

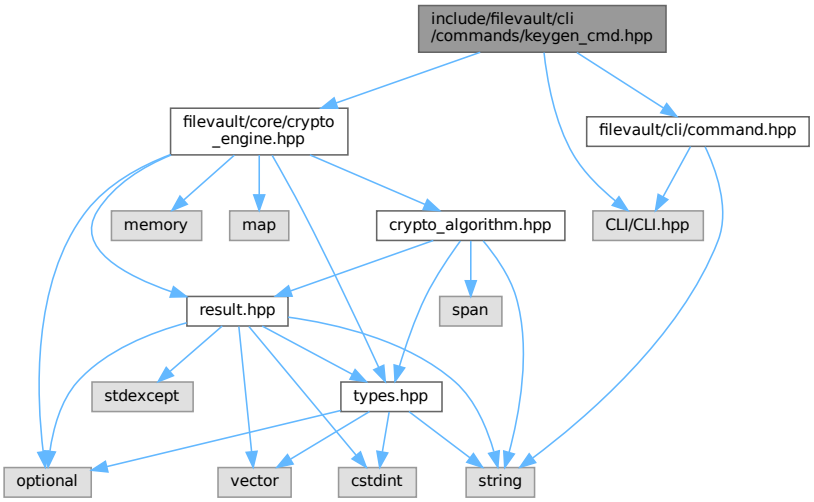
```
00049         bool has_header = false;
00050         std::string version;
00051         std::string algorithm;
00052         std::string kdf;
00053         std::string compression;
00054         bool compressed = false;
00055     };
00056
00057     FileInfo parse_file(const std::string& path);
00058     void display_info(const FileInfo& info);
00059 };
00060
00061 } // namespace cli
00062 } // namespace filevault
00063
00064 #endif // FILEVAULT_CLI_COMMANDS_INFO_CMD_HPP
```

10.73 include/filevault/cli/commands/keygen_cmd.hpp File Reference

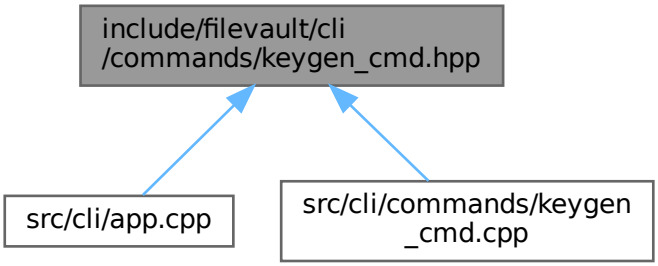
Key generation command for asymmetric encryption.

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
#include <CLI/CLI.hpp>
```

Include dependency graph for keygen_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::KeygenCommand

  Command to generate key pairs for asymmetric encryption.

Namespaces

- namespace filevault
- namespace filevault::cli

10.73.1 Detailed Description

Key generation command for asymmetric encryption.

10.74 keygen_cmd.hpp

Go to the documentation of this file.

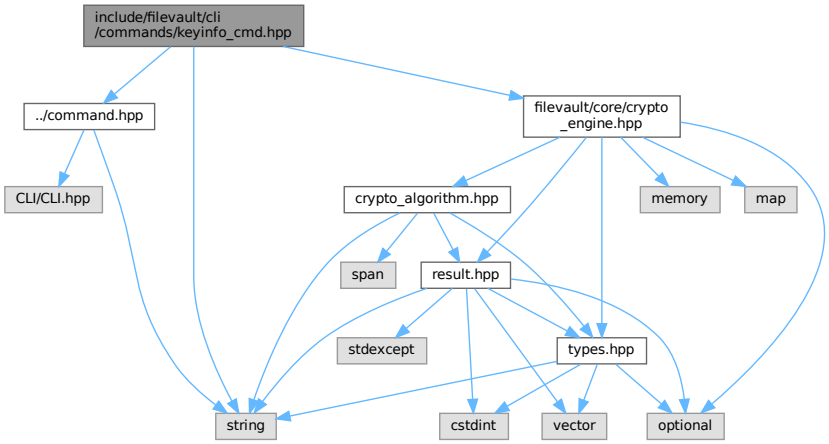
```
00001
00005
00006 #ifndef FILEVAULT_CLI_COMMANDS_KEYGEN_CMD_HPP
00007 #define FILEVAULT_CLI_COMMANDS_KEYGEN_CMD_HPP
00008
00009 #include "filevault/cli/command.hpp"
00010 #include "filevault/core/crypto_engine.hpp"
00011 #include <CLI/CLI.hpp>
00012
00013 namespace filevault {
00014 namespace cli {
00015
00019 class KeygenCommand : public ICommand {
00020 public:
00021     explicit KeygenCommand(core::CryptoEngine& engine);
00022
00023     std::string name() const override { return "keygen"; }
00024     std::string description() const override {
00025         return "Generate key pair for asymmetric/PQC encryption (RSA/ECC/Kyber/Dilithium)";
00026     }
00027
00028     void setup(CLI::App& app) override;
00029     int execute() override;
00030
00031 private:
00032     core::CryptoEngine& engine_;
00033     std::string algorithm_ = "rsa-2048";
00034     std::string output_prefix_ = "filevault_key";
00035     bool force_ = false;
00036     bool verbose_ = false;
00037 };
00038
```

```
00039 } // namespace cli
00040 } // namespace filevault
00041
00042 #endif
```

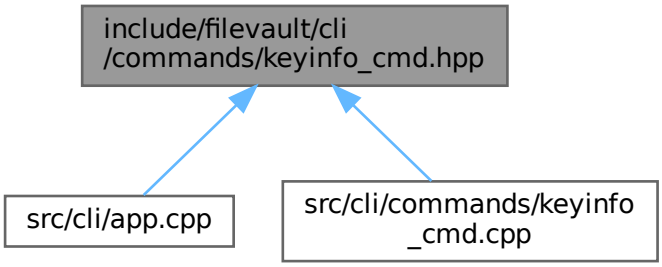
10.75 include/filevault/cli/commands/keyinfo_cmd.hpp File Reference

```
#include "../command.hpp"
#include "filevault/core/crypto_engine.hpp"
#include <string>
```

Include dependency graph for keyinfo_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::commands::KeyInfoCommand

  Command to display information about cryptographic keys.

Namespaces

- namespace filevault
- namespace filevault::cli

- namespace filevault::cli::commands

10.76 keyinfo_cmd.hpp

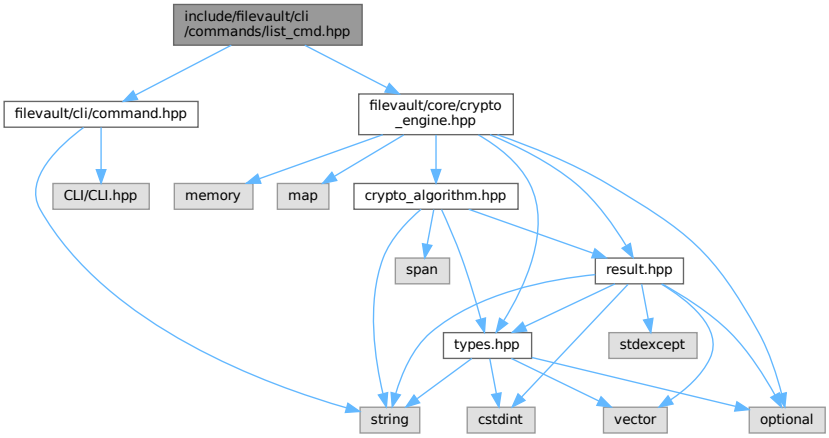
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMANDS_KEYINFO_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_KEYINFO_CMD_HPP
00003
00004 #include "../command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006 #include <string>
00007
00008 namespace filevault {
00009 namespace cli {
00010 namespace commands {
00011
00015 class KeyInfoCommand : public ICommand {
00016 public:
00017     explicit KeyInfoCommand(core::CryptoEngine& engine);
00018
00019     std::string name() const override { return name_; }
00020     std::string description() const override { return description_; }
00021
00022     void setup(CLI::App& app) override;
00023     int execute() override;
00024
00025 private:
00026     std::string name_ = "keyinfo";
00027     std::string description_ = "Display information about cryptographic keys";
00028
00029     core::CryptoEngine& engine_;
00030     std::string key_path_;
00031     bool show_public_ = false;
00032     bool check_pair_ = false;
00033     std::string pair_key_path_;
00034 };
00035
00036 } // namespace commands
00037 } // namespace cli
00038 } // namespace filevault
00039
00040 #endif // FILEVAULT_CLI_COMMANDS_KEYINFO_CMD_HPP
```

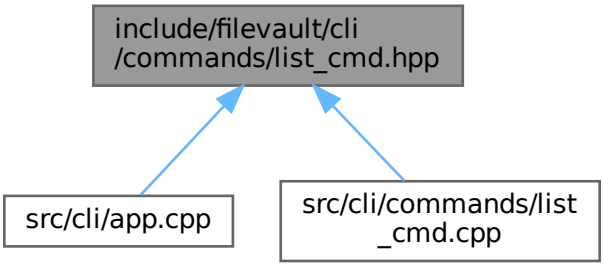
10.77 include/filevault/cli/commands/list_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include "filevault/core/crypto_engine.hpp"
```

Include dependency graph for list_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::ListCommand

Namespaces

- namespace filevault
- namespace filevault::cli

10.78 list_cmd.hpp

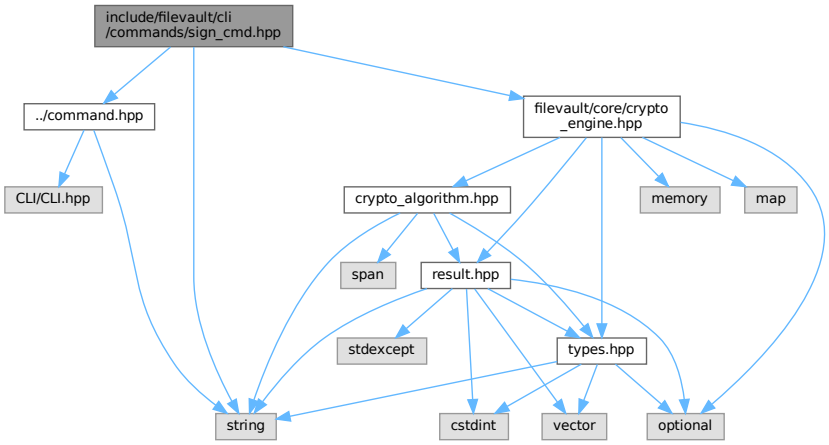
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMANDS_LIST_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_LIST_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006
00007 namespace filevault {
00008 namespace cli {
00009
00010 class ListCommand : public ICommand {
00011 public:
00012     explicit ListCommand(core::CryptoEngine& engine);
00013
00014     std::string name() const override { return "list"; }
00015     std::string description() const override { return "List available algorithms"; }
00016
00017     void setup(CLI::App& app) override;
00018     int execute() override;
00019
00020 private:
00021     core::CryptoEngine& engine_;
00022 };
00023
00024 } // namespace cli
00025 } // namespace filevault
00026
00027 #endif
```

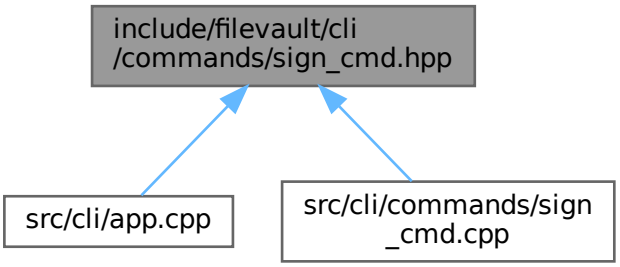
10.79 include/filevault/cli/commands/sign_cmd.hpp File Reference

```
#include "../command.hpp"
#include "filevault/core/crypto_engine.hpp"
#include <string>
```

Include dependency graph for sign_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::commands::SignCommand

  Command to create digital signature for a file.

Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.80 sign_cmd.hpp

Go to the documentation of this file.

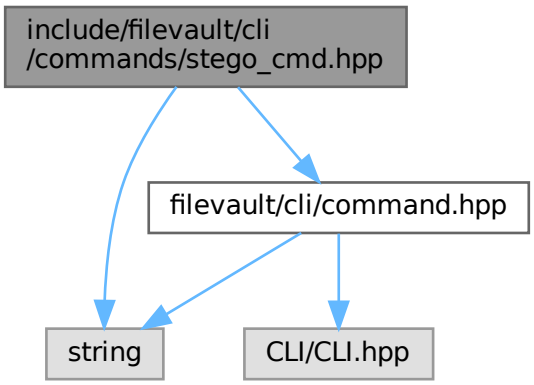
```
00001 #ifndef FILEVAULT_CLI_COMMANDS_SIGN_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_SIGN_CMD_HPP
00003
00004 #include "../command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
```

```
00006 #include <string>
00007
00008 namespace filevault {
00009 namespace cli {
00010 namespace commands {
00011
00015 class SignCommand : public ICommand {
00016 public:
00017     explicit SignCommand(core::CryptoEngine& engine);
00018
00019     std::string name() const override { return name_; }
00020     std::string description() const override { return description_; }
00021
00022     void setup(CLI::App& app) override;
00023     int execute() override;
00024
00025 private:
00026     std::string name_ = "sign";
00027     std::string description_ = "Create digital signature for a file";
00028
00029     core::CryptoEngine& engine_;
00030     std::string file_path_;
00031     std::string private_key_path_;
00032     std::string output_path_;
00033     std::string algorithm_ = "rsa";  // rsa, ecc, ed25519
00034 };
00035
00036 } // namespace commands
00037 } // namespace cli
00038 } // namespace filevault
00039
00040 #endif // FILEVAULT_CLI_COMMANDS_SIGN_CMD_HPP
```

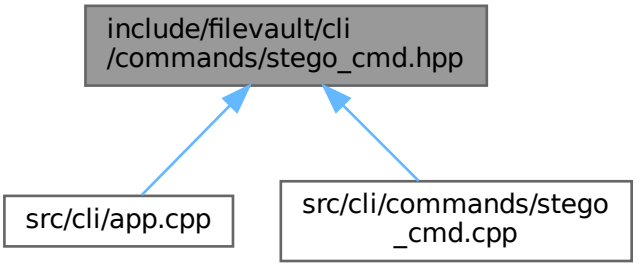
10.81 include/filevault/cli/commands/stego_cmd.hpp File Reference

```
#include "filevault/cli/command.hpp"
#include <string>
```

Include dependency graph for stego_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::commands::StegoCommand

  Steganography command for embedding/extracting data in images.

Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.82 stego_cmd.hpp

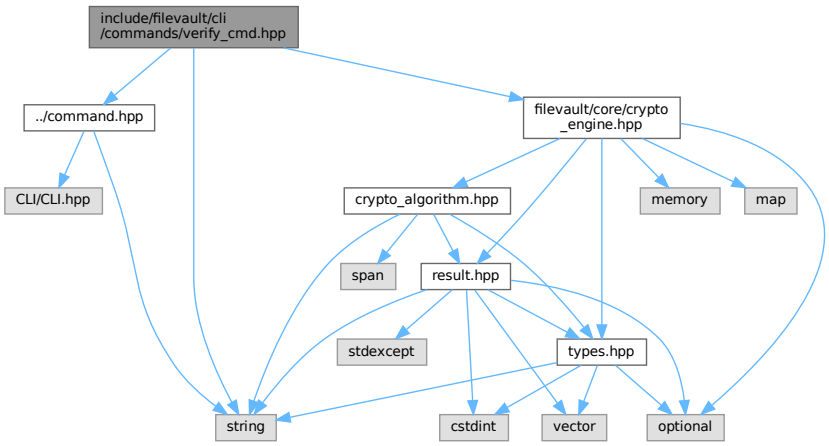
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMANDS_STEGO_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_STEGO_CMD_HPP
00003
00004 #include "filevault/cli/command.hpp"
00005 #include <string>
00006
00007 namespace filevault::cli::commands {
00008
00021 class StegoCommand : public ICommand {
00022 public:
00023     std::string name() const override { return "stego"; }
00024     std::string description() const override {
00025         return "Hide or extract data from images using steganography";
00026     }
00027
00028     void setup(CLI::App& app) override;
00029     int execute() override;
00030
00031 private:
00032     // Subcommand handlers
00033     int do_embed();
00034     int do_extract();
00035     int do_capacity();
00036
00037     // Options
00038     std::string operation_;           // "embed", "extract", or "capacity"
00039     std::string input_file_;          // Secret file (for embed) or stego image (for extract)
00040     std::string cover_image_;         // Cover image (for embed)
00041     std::string output_file_;         // Output path
00042     int bits_per_channel_ = 1;        // 1-4 bits per channel
00043     bool verbose_ = false;
00044 };
00045
00046 } // namespace filevault::cli::commands
00047
00048 #endif // FILEVAULT_CLI_COMMANDS_STEGO_CMD_HPP
```

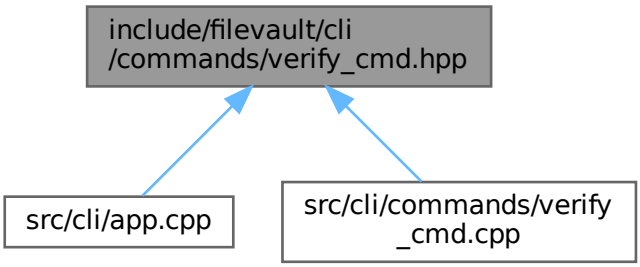
10.83 include/filevault/cli/commands/verify_cmd.hpp File Reference

```
#include "../command.hpp"
#include "filevault/core/crypto_engine.hpp"
#include <string>
```

Include dependency graph for verify_cmd.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::cli::commands::VerifyCommand

  Command to verify digital signature of a file.

Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.84 verify_cmd.hpp

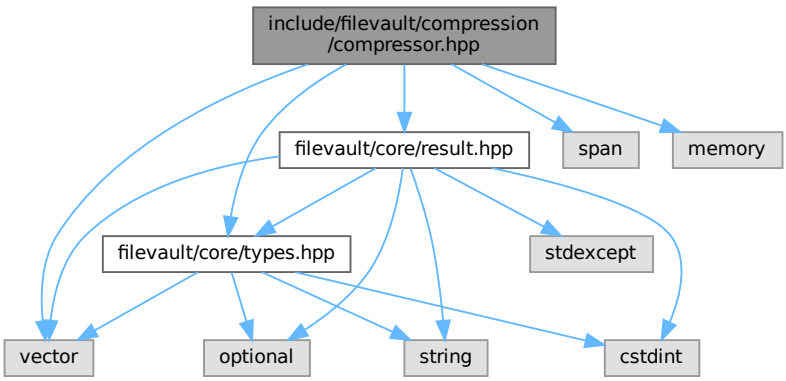
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CLI_COMMANDS_VERIFY_CMD_HPP
00002 #define FILEVAULT_CLI_COMMANDS_VERIFY_CMD_HPP
00003
00004 #include "../command.hpp"
00005 #include "filevault/core/crypto_engine.hpp"
00006 #include <string>
00007
00008 namespace filevault {
00009 namespace cli {
00010 namespace commands {
00011
00015 class VerifyCommand : public ICommand {
00016 public:
00017     explicit VerifyCommand(core::CryptoEngine& engine);
00018
00019     std::string name() const override { return name_; }
00020     std::string description() const override { return description_; }
00021
00022     void setup(CLI::App& app) override;
00023     int execute() override;
00024
00025 private:
00026     std::string name_ = "verify";
00027     std::string description_ = "Verify digital signature of a file";
00028
00029     core::CryptoEngine& engine_;
00030     std::string file_path_;
00031     std::string signature_path_;
00032     std::string public_key_path_;
00033     std::string algorithm_ = "rsa";
00034 };
00035
00036 } // namespace commands
00037 } // namespace cli
00038 } // namespace filevault
00039
00040 #endif // FILEVAULT_CLI_COMMANDS_VERIFY_CMD_HPP
```

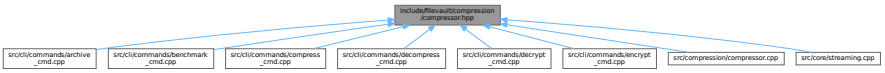
10.85 include/filevault/compression/compressor.hpp File Reference

```
#include "filevault/core/types.hpp"
#include "filevault/core/result.hpp"
#include <vector>
#include <span>
#include <memory>
```

Include dependency graph for compressor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::compression::CompressionResult

  Compression result.
- class filevault::compression::ICompressor

  Interface for compression algorithms.
- class filevault::compression::CompressionService

  Compression service - factory for compressors.
- class filevault::compression::ZlibCompressor

  ZLIB compressor (fast, good compression).
- class filevault::compression::Bzip2Compressor

  BZIP2 compressor (better ratio, slower).
- class filevault::compression::LzmaCompressor

  LZMA compressor (maximum compression, slowest).

Namespaces

- namespace filevault
- namespace filevault::compression

10.86 compressor.hpp

Go to the documentation of this file.

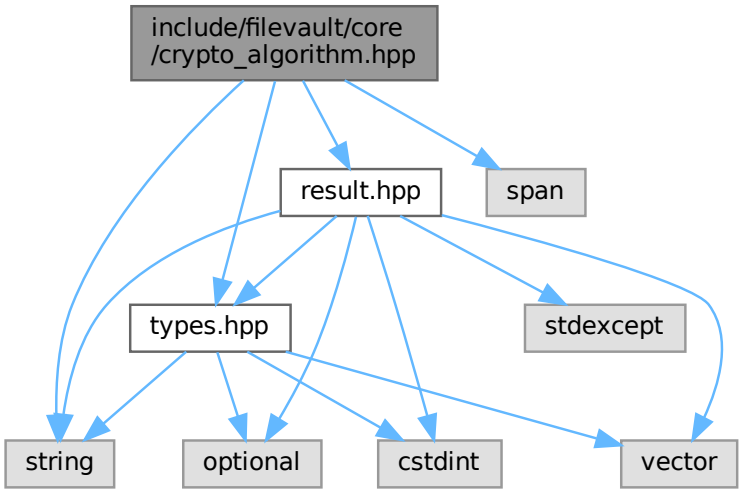
```
00001 #ifndef FILEVAULT_COMPRESSION_COMPRESSOR_HPP
00002 #define FILEVAULT_COMPRESSION_COMPRESSOR_HPP
00003
00004 #include "filevault/core/types.hpp"
00005 #include "filevault/core/result.hpp"
00006 #include <vector>
00007 #include <span>
00008 #include <memory>
00009
00010 namespace filevault {
00011 namespace compression {
00012
00016 struct CompressionResult {
00017     bool success = false;
00018     std::string error_message;
00019     std::vector<uint8_t> data;
00020     size_t original_size = 0;
00021     size_t compressed_size = 0;
00022     double compression_ratio = 0.0;  // Percentage saved
00023     double processing_time_ms = 0.0;
00024 };
00025
00029 class ICompressor {
00030 public:
00031     virtual ~ICompressor() = default;
00032
00036     virtual std::string name() const = 0;
00037
00044     virtual CompressionResult compress(
00045         std::span<const uint8_t> input,
00046         int level = 6
00047     ) = 0;
00048
00054     virtual CompressionResult decompress(
00055         std::span<const uint8_t> input
00056     ) = 0;
00057 };
00058
```

```
00062 class CompressionService {
00063 public:
00067     static std::unique_ptr<ICompressor> create(core::CompressionType type);
00068
00072     static std::string get_algorithm_name(core::CompressionType type);
00073
00077     static core::CompressionType parse_algorithm(const std::string& name);
00078 };
00079
00083 class ZlibCompressor : public ICompressor {
00084 public:
00085     std::string name() const override { return "zlib"; }
00086
00087     CompressionResult compress(
00088         std::span<const uint8_t> input,
00089         int level = 6
00090     ) override;
00091
00092     CompressionResult decompress(
00093         std::span<const uint8_t> input
00094     ) override;
00095 };
00096
00100 class Bzip2Compressor : public ICompressor {
00101 public:
00102     std::string name() const override { return "bzip2"; }
00103
00104     CompressionResult compress(
00105         std::span<const uint8_t> input,
00106         int level = 6
00107     ) override;
00108
00109     CompressionResult decompress(
00110         std::span<const uint8_t> input
00111     ) override;
00112 };
00113
00117 class LzmaCompressor : public ICompressor {
00118 public:
00119     std::string name() const override { return "lzma"; }
00120
00121     CompressionResult compress(
00122         std::span<const uint8_t> input,
00123         int level = 6
00124     ) override;
00125
00126     CompressionResult decompress(
00127         std::span<const uint8_t> input
00128     ) override;
00129 };
00130
00131 } // namespace compression
00132 } // namespace filevault
00133
00134 #endif // FILEVAULT_COMPRESSION_COMPRESSOR_HPP
```

10.87 include/filevault/core/crypto_algorithm.hpp File Reference

```
#include <string>
#include <span>
#include "types.hpp"
#include "result.hpp"
```

Include dependency graph for crypto_algorithm.hpp:



This graph shows which files directly or indirectly include this file:

Classes

- class filevault::core::ICryptoAlgorithm

  Interface for cryptographic algorithms.

Namespaces

- namespace filevault
- namespace filevault::core

10.88 crypto_algorithm.hpp

Go to the documentation of this file.

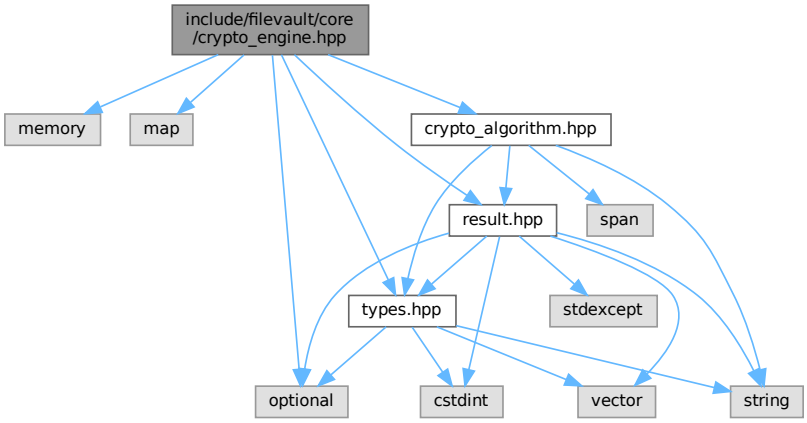
```
00001 #ifndef FILEVAULT_CORE_CRYPTO_ALGORITHM_HPP
00002 #define FILEVAULT_CORE_CRYPTO_ALGORITHM_HPP
00003
00004 #include <string>
00005 #include <span>
00006 #include "types.hpp"
00007 #include "result.hpp"
00008
00009 namespace filevault {
00010 namespace core {
00011
00015 class ICryptoAlgorithm {
00016 public:
00017     virtual ~ICryptoAlgorithm() = default;
00018
00022     virtual std::string name() const = 0;
00023
00027     virtual AlgorithmType type() const = 0;
00028
00036     virtual CryptoResult encrypt(
```

```
00037         std::span<const uint8_t> plaintext,
00038         std::span<const uint8_t> key,
00039         const EncryptionConfig& config
00040     ) = 0;
00041
00049     virtual CryptoResult decrypt(
00050         std::span<const uint8_t> ciphertext,
00051         std::span<const uint8_t> key,
00052         const EncryptionConfig& config
00053     ) = 0;
00054
00058     virtual size_t key_size() const = 0;
00059
00063     virtual bool is_suitable_for(SecurityLevel level) const = 0;
00064 };
00065
00066 } // namespace core
00067 } // namespace filevault
00068
00069 #endif // FILEVAULT_CORE_CRYPTO_ALGORITHM_HPP
```

10.89 include/filevault/core/crypto_engine.hpp File Reference

```
#include <memory>
#include <map>
#include <optional>
#include "crypto_algorithm.hpp"
#include "types.hpp"
#include "result.hpp"
```

Include dependency graph for crypto_engine.hpp:



This graph shows which files directly or indirectly include this file:

Classes

- class filevault::core::CryptoEngine

  Main cryptographic engine Manages algorithms and provides key derivation.

Namespaces

- namespace filevault
- namespace filevault::core

10.90 crypto_engine.hpp

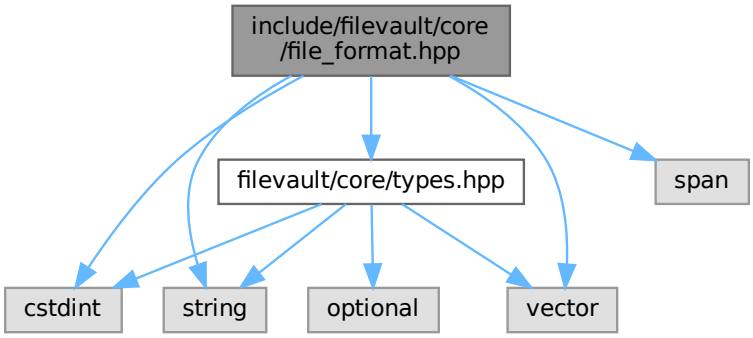
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CORE_CRYPTO_ENGINE_HPP
00002 #define FILEVAULT_CORE_CRYPTO_ENGINE_HPP
00003
00004 #include <memory>
00005 #include <map>
00006 #include <optional>
00007 #include "crypto_algorithm.hpp"
00008 #include "types.hpp"
00009 #include "result.hpp"
00010
00011 namespace filevault {
00012 namespace core {
00013
00018 class CryptoEngine {
00019 public:
00020     CryptoEngine();
00021     ~CryptoEngine();
00022
00023     // Prevent copying
00024     CryptoEngine(const CryptoEngine&) = delete;
00025     CryptoEngine& operator=(const CryptoEngine&) = delete;
00026
00030     void initialize();
00031
00035     void register_algorithm(std::unique_ptr<ICryptoAlgorithm> algorithm);
00036
00040     ICryptoAlgorithm* get_algorithm(AlgorithmType type);
00041
00049     std::vector<uint8_t> derive_key(
00050         const std::string& password,
00051         const std::vector<uint8_t>& salt,
00052         const EncryptionConfig& config
00053     );
00054
00058     static std::vector<uint8_t> generate_salt(size_t length = 32);
00059
00064     static std::vector<uint8_t> generate_nonce(size_t length = 12);
00065
00069     static std::string algorithm_name(AlgorithmType type);
00070
00074     static std::string kdf_name(KDFType type);
00075
00079     static std::string security_level_name(SecurityLevel level);
00080
00084     static std::optional<AlgorithmType> parse_algorithm(const std::string& name);
00085
00089     static std::optional<KDFType> parse_kdf(const std::string& name);
00090
00094     static std::optional<SecurityLevel> parse_security_level(const std::string& name);
00095
00096 private:
00097     std::map<AlgorithmType, std::unique_ptr<ICryptoAlgorithm» algorithms_;
00098 };
00099
00100 } // namespace core
00101 } // namespace filevault
00102
00103 #endif // FILEVAULT_CORE_CRYPTO_ENGINE_HPP
```

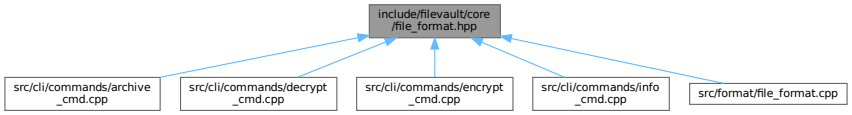
10.91 include/filevault/core/file_format.hpp File Reference

```
#include "filevault/core/types.hpp"
#include <vector>
#include <string>
#include <cstdint>
#include <span>
```

Include dependency graph for file_format.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::core::Argon2Params

  KDF parameters for Argon2.
- struct filevault::core::PBKDF2Params

  KDF parameters for PBKDF2.
- struct filevault::core::ScryptParams

  KDF parameters for Scrypt.
- struct filevault::core::FileHeader

  File format header.
- class filevault::core::FileFormatHandler

  File format handler.

Namespaces

- namespace filevault
- namespace filevault::core

Enumerations

- enum class filevault::core::AlgorithmID : uint8_t {
  filevault::core::UNKNOWN = 0x00 , filevault::core::AES_128_GCM = 0x01 , filevault::core::AES_192_GCM = 0x02 , filevault::core::AES_256_GCM = 0x03 ,
  filevault::core::CHACHA20_POLY1305 = 0x04 , filevault::core::SERPENT_256_GCM = 0x05 , filevault::core::TWOFISH_128_GC = 0x06 , filevault::core::TWOFISH_192_GCM = 0x07 ,
  filevault::core::TWOFISH_256_GCM = 0x08 , filevault::core::CAMELLIA_128_GCM = 0x09 , filevault::core::CAMELLIA_192_GC = 0x0A , filevault::core::CAMELLIA_256_GCM = 0x0B ,

filevault::core::ARIA_128_GCM = 0x0C , filevault::core::ARIA_192_GCM = 0x0D , filevault::core::ARIA_256_GCM = 0x0E , filevault::core::SM4_GCM = 0x0F ,
filevault::core::CAESAR = 0x10 , filevault::core::VIGENERE = 0x11 , filevault::core::PLAYFAIR = 0x12 , filevault::core::SUBSTITUTION = 0x13 ,
filevault::core::HILL = 0x14 , filevault::core::AES_128_CBC = 0x20 , filevault::core::AES_192_CBC = 0x21 , filevault::core::AES_256_CBC = 0x22 ,
filevault::core::AES_128_CTR = 0x23 , filevault::core::AES_192_CTR = 0x24 , filevault::core::AES_256_CTR = 0x25 , filevault::core::AES_128_CFB = 0x26 ,
filevault::core::AES_192_CFB = 0x27 , filevault::core::AES_256_CFB = 0x28 , filevault::core::AES_128_OFB = 0x29 , filevault::core::AES_192_OFB = 0x2A ,
filevault::core::AES_256_OFB = 0x2B , filevault::core::AES_128_ECB = 0x2C , filevault::core::AES_192_ECB = 0x2D , filevault::core::AES_256_ECB = 0x2E ,
filevault::core::AES_128_XTS = 0x2F , filevault::core::AES_256_XTS = 0x30 , filevault::core::TRIPLE_DES_CBC = 0x40 , filevault::core::RSA_2048 = 0x50 ,
filevault::core::RSA_3072 = 0x51 , filevault::core::RSA_4096 = 0x52 , filevault::core::ECC_P256 = 0x60 , filevault::core::ECC_P384 = 0x61 ,
filevault::core::ECC_P521 = 0x62 }

Algorithm identifiers.

- enum class filevault::core::KDFID : uint8_t {
filevault::core::NONE = 0x00 , filevault::core::ARGON2ID = 0x01 , filevault::core::ARGON2I = 0x02 , filevault::core::PBKDF2_SHA256 = 0x03 ,
filevault::core::PBKDF2_SHA512 = 0x04 , filevault::core::SCRYPT = 0x05 }

  KDF identifiers.
- enum class filevault::core::CompressionID : uint8_t { filevault::core::NONE = 0x00 , filevault::core::ZLIB = 0x01 , filevault::core::BZIP2 = 0x02 , filevault::core::LZMA = 0x03 }

  Compression identifiers.

Variables

- constexpr uint8_t filevault::core::FILE_FORMAT_MAGIC [8] = {'F', 'V', 'A', 'U', 'L', 'T', '0', '1'}

  FileVault file format version 1.0.
- constexpr uint8_t filevault::core::FILE_FORMAT_VERSION_MAJOR = 1
- constexpr uint8_t filevault::core::FILE_FORMAT_VERSION_MINOR = 0

10.92 file_format.hpp

Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CORE_FILE_FORMAT_HPP
00002 #define FILEVAULT_CORE_FILE_FORMAT_HPP
00003
00004 #include "filevault/core/types.hpp"
00005 #include <vector>
00006 #include <string>
00007 #include <cstdint>
00008 #include <span>
00009
00010 namespace filevault {
00011 namespace core {
00012
00035
00036 constexpr uint8_t FILE_FORMAT_MAGIC[8] = {'F', 'V', 'A', 'U', 'L', 'T', '0', '1'};
00037 constexpr uint8_t FILE_FORMAT_VERSION_MAJOR = 1;
00038 constexpr uint8_t FILE_FORMAT_VERSION_MINOR = 0;
00039
00043 enum class AlgorithmID : uint8_t {
00044     UNKNOWN = 0x00,
00045     AES_128_GCM = 0x01,
00046     AES_192_GCM = 0x02,
00047     AES_256_GCM = 0x03,
00048     CHACHA20_POLY1305 = 0x04,
00049     SERPENT_256_GCM = 0x05,
00050     // Twofish family
00051     TWOFISH_128_GCM = 0x06,
00052     TWOFISH_192_GCM = 0x07,
00053     TWOFISH_256_GCM = 0x08,
00054     // International standards
```

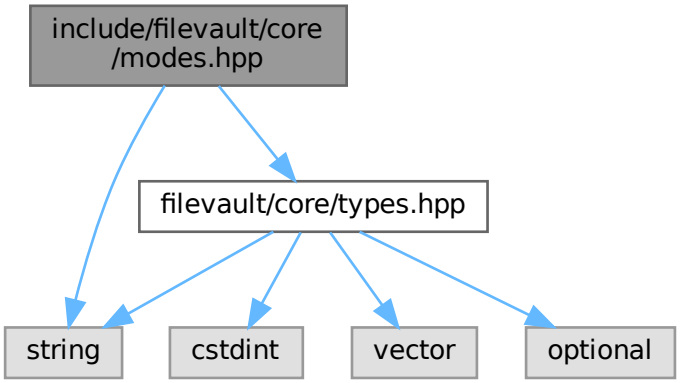
```
00055     CAMELLIA_128_GCM = 0x09,
00056     CAMELLIA_192_GCM = 0x0A,
00057     CAMELLIA_256_GCM = 0x0B,
00058     ARIA_128_GCM = 0x0C,
00059     ARIA_192_GCM = 0x0D,
00060     ARIA_256_GCM = 0x0E,
00061     SM4_GCM = 0x0F,
00062     // Classical ciphers
00063     CAESAR = 0x10,
00064     VIGENERE = 0x11,
00065     PLAYFAIR = 0x12,
00066     SUBSTITUTION = 0x13,
00067     HILL = 0x14,
00068     // Non-AEAD modes (CBC, CTR)
00069     AES_128_CBC = 0x20,
00070     AES_192_CBC = 0x21,
00071     AES_256_CBC = 0x22,
00072     AES_128_CTR = 0x23,
00073     AES_192_CTR = 0x24,
00074     AES_256_CTR = 0x25,
00075     // CFB mode
00076     AES_128_CFB = 0x26,
00077     AES_192_CFB = 0x27,
00078     AES_256_CFB = 0x28,
00079     // OFB mode
00080     AES_128_OFB = 0x29,
00081     AES_192_OFB = 0x2A,
00082     AES_256_OFB = 0x2B,
00083     // ECB mode (insecure)
00084     AES_128_ECB = 0x2C,
00085     AES_192_ECB = 0x2D,
00086     AES_256_ECB = 0x2E,
00087     // XTS mode (disk encryption)
00088     AES_128_XTS = 0x2F,
00089     AES_256_XTS = 0x30,
00090     // Legacy algorithms
00091     TRIPLE_DES_CBC = 0x40,
00092     // Asymmetric (RSA)
00093     RSA_2048 = 0x50,
00094     RSA_3072 = 0x51,
00095     RSA_4096 = 0x52,
00096     // Asymmetric (ECC)
00097     ECC_P256 = 0x60,
00098     ECC_P384 = 0x61,
00099     ECC_P521 = 0x62
00100 };
00101
00105 enum class KDFID : uint8_t {
00106     NONE = 0x00,
00107     ARGON2ID = 0x01,
00108     ARGON2I = 0x02,
00109     PBKDF2_SHA256 = 0x03,
00110     PBKDF2_SHA512 = 0x04,
00111     SCRYPT = 0x05
00112 };
00113
00117 enum class CompressionID : uint8_t {
00118     NONE = 0x00,
00119     ZLIB = 0x01,
00120     BZIP2 = 0x02,
00121     LZMA = 0x03
00122 };
00123
00127 struct Argon2Params {
00128     uint32_t memory_kb = 65536;     // 64 MB default
00129     uint32_t iterations = 3;
00130     uint32_t parallelism = 4;
00131
00132     std::vector<uint8_t> serialize() const;
00133     static Argon2Params deserialize(std::span<const uint8_t> data);
00134 };
00135
00139 struct PBKDF2Params {
00140     uint32_t iterations = 100000;
00141
00142     std::vector<uint8_t> serialize() const;
00143     static PBKDF2Params deserialize(std::span<const uint8_t> data);
00144 };
00145
00149 struct ScryptParams {
00150     uint32_t n = 32768;             // CPU/memory cost
00151     uint32_t r = 8;                 // Block size
00152     uint32_t p = 1;                 // Parallelization
00153
00154     std::vector<uint8_t> serialize() const;
00155     static ScryptParams deserialize(std::span<const uint8_t> data);
00156 };
```

```
00157
00161 struct FileHeader {
00162     uint8_t magic[8];
00163     uint8_t version_major;
00164     uint8_t version_minor;
00165     AlgorithmID algorithm;
00166     KDFID kdf;
00167     CompressionID compression;
00168     uint8_t reserved[3];
00169     std::vector<uint8_t> salt;
00170     std::vector<uint8_t> kdf_params;
00171     std::vector<uint8_t> nonce;
00172     bool compressed;
00173
00177     bool is_valid() const;
00178
00182     size_t size() const;
00183
00187     std::vector<uint8_t> serialize() const;
00188
00193     static std::pair<FileHeader, size_t> deserialize(std::span<const uint8_t> data);
00194 };
00195
00199 class FileFormatHandler {
00200 public:
00204     static FileHeader create_header(
00205         AlgorithmType algo_type,
00206         KDFType kdf_type,
00207         const EncryptionConfig& config,
00208         std::span<const uint8_t> salt,
00209         std::span<const uint8_t> nonce,
00210         bool compressed
00211     );
00212
00216     static bool write_file(
00217         const std::string& path,
00218         const FileHeader& header,
00219         std::span<const uint8_t> ciphertext,
00220         std::span<const uint8_t> auth_tag
00221     );
00222
00226     static std::tuple<FileHeader, std::vector<uint8_t>, std::vector<uint8_t» read_file(
00227         const std::string& path
00228     );
00229
00233     static AlgorithmID to_algorithm_id(AlgorithmType type);
00234
00238     static AlgorithmType from_algorithm_id(AlgorithmID id);
00239
00243     static KDFID to_kdf_id(KDFType type);
00244
00248     static KDFType from_kdf_id(KDFID id);
00249
00253     static CompressionID to_compression_id(const std::string& type);
00254
00258     static std::string from_compression_id(CompressionID id);
00259
00263     static bool is_legacy_format(const std::string& path);
00264
00268     static std::tuple<std::vector<uint8_t>, std::vector<uint8_t>, std::vector<uint8_t»
    read_legacy_file(
00269         const std::string& path
00270     );
00271 };
00272
00273 } // namespace core
00274 } // namespace filevault
00275
00276 #endif
```

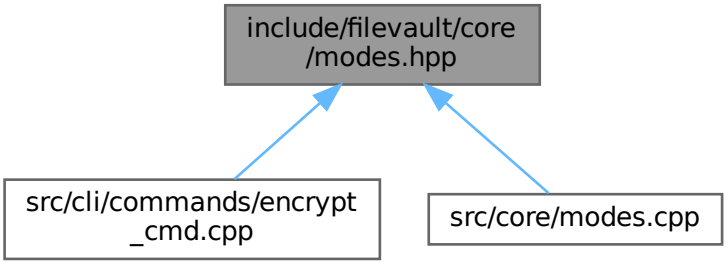
10.93 include/filevault/core/modes.hpp File Reference

```
#include "filevault/core/types.hpp"
#include <string>
```

Include dependency graph for modes.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::core::ModePreset

  Mode presets for different user levels.

Namespaces

- namespace filevault
- namespace filevault::core
- namespace filevault::core::presets

10.94 modes.hpp

Go to the documentation of this file.

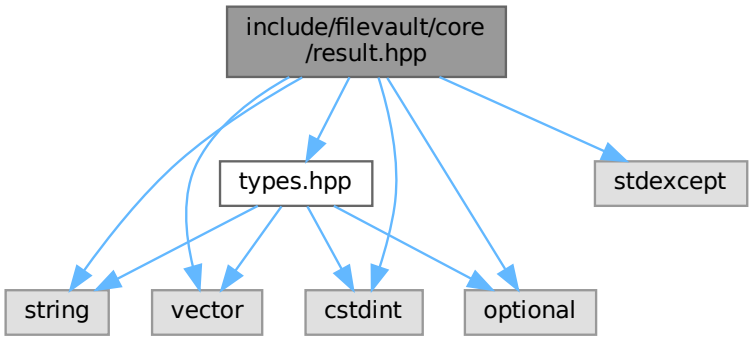
```
00001 #ifndef FILEVAULT_CORE_MODES_HPP
00002 #define FILEVAULT_CORE_MODES_HPP
00003
```

```
00004 #include "filevault/core/types.hpp"
00005 #include <string>
00006
00007 namespace filevault {
00008 namespace core {
00009
00017 struct ModePreset {
00018     UserMode mode;
00019     AlgorithmType algorithm;
00020     KDFType kdf;
00021     SecurityLevel security_level;
00022     CompressionType compression;
00023     int compression_level;
00024
00025     // KDF parameters
00026     uint32_t kdf_iterations;
00027     uint32_t kdf_memory_kb;
00028     uint32_t kdf_parallelism;
00029
00030     std::string name() const;
00031     std::string description() const;
00032
00036     void apply_to(EncryptionConfig& config) const;
00037
00041     static ModePreset get_preset(UserMode mode);
00042
00046     static UserMode parse_mode(const std::string& name);
00047
00051     static std::vector<ModePreset> get_all_presets();
00052 };
00053
00054 // Predefined presets
00055 namespace presets {
00056     extern const ModePreset BASIC;
00057     extern const ModePreset STANDARD;
00058     extern const ModePreset ADVANCED;
00059 }
00060
00061 } // namespace core
00062 } // namespace filevault
00063
00064 #endif // FILEVAULT_CORE_MODES_HPP
```

10.95 include/filevault/core/result.hpp File Reference

```
#include <string>
#include <vector>
#include <cstdint>
#include <optional>
#include <stdexcept>
#include "types.hpp"
```

Include dependency graph for result.hpp:



This graph shows which files directly or indirectly include this file:

Classes

- struct filevault::core::CryptoResult

  Result of cryptographic operations.
- struct filevault::core::Result< T >

  Generic result type.
- struct filevault::core::Result< void >

  Specialization for void.

Namespaces

- namespace filevault
- namespace filevault::core

10.96 result.hpp

Go to the documentation of this file.

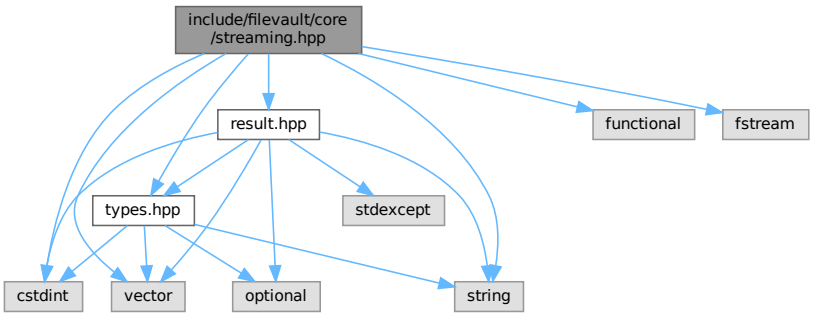
```
00001 #ifndef FILEVAULT_CORE_RESULT_HPP
00002 #define FILEVAULT_CORE_RESULT_HPP
00003
00004 #include <string>
00005 #include <vector>
00006 #include <cstdint>
00007 #include <optional>
00008 #include <stdexcept>
00009 #include "types.hpp"
00010
00011 namespace filevault {
00012 namespace core {
00013
00017 struct CryptoResult {
00018     bool success = false;
00019     std::string error_message;
00020     std::vector<uint8_t> data;
00021
00022     // Metadata
00023     AlgorithmType algorithm_used;
00024     size_t original_size = 0;
00025     size_t final_size = 0;
00026     double processing_time_ms = 0.0;
00027
00028     // Additional info
00029     std::optional<std::vector<uint8_t» salt;
00030     std::optional<std::vector<uint8_t» nonce;
00031     std::optional<std::vector<uint8_t» tag;
00032 };
00033
00037 template<typename T>
00038 struct Result {
00039     bool success = false;
00040     std::string error_message;
00041     T value;
00042
00043     // Success constructor
00044     static Result<T> ok(T val) {
00045         Result<T> r;
00046         r.success = true;
00047         r.value = std::move(val);
00048         return r;
00049     }
00050
00051     // Error constructor
00052     static Result<T> error(const std::string& msg) {
00053         Result<T> r;
00054         r.success = false;
00055         r.error_message = msg;
00056         return r;
```

```
00057     }
00058
00059     // Check if result is ok
00060     explicit operator bool() const { return success; }
00061
00062     // Get value (throws if error)
00063     T& unwrap() {
00064         if (!success) {
00065             throw std::runtime_error("Attempted to unwrap error: " + error_message);
00066         }
00067         return value;
00068     }
00069 };
00070
00074 template<>
00075 struct Result<void> {
00076     bool success = false;
00077     std::string error_message;
00078
00079     static Result<void> ok() {
00080         Result<void> r;
00081         r.success = true;
00082         return r;
00083     }
00084
00085     static Result<void> error(const std::string& msg) {
00086         Result<void> r;
00087         r.success = false;
00088         r.error_message = msg;
00089         return r;
00090     }
00091
00092     explicit operator bool() const { return success; }
00093 };
00094
00095 } // namespace core
00096 } // namespace filevault
00097
00098 #endif // FILEVAULT_CORE_RESULT_HPP
```

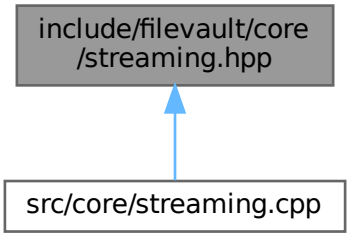
10.97 include/filevault/core/streaming.hpp File Reference

```
#include <cstdint>
#include <vector>
#include <string>
#include <functional>
#include <fstream>
#include "types.hpp"
#include "result.hpp"
```

Include dependency graph for streaming.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::core::ChunkInfo

  Chunk information for streaming encryption.
- struct filevault::core::StreamingConfig

  Configuration for streaming encryption.
- struct filevault::core::StreamingResult

  Result of streaming operation.
- class filevault::core::StreamingCrypto

  Streaming encryption/decryption for large files.

Namespaces

- namespace filevault
- namespace filevault::core

Typedefs

- using filevault::core::StreamProgressCallback = std::function<bool(const ChunkInfo& info)>

  Progress callback for streaming operations.

10.98 streaming.hpp

Go to the documentation of this file.

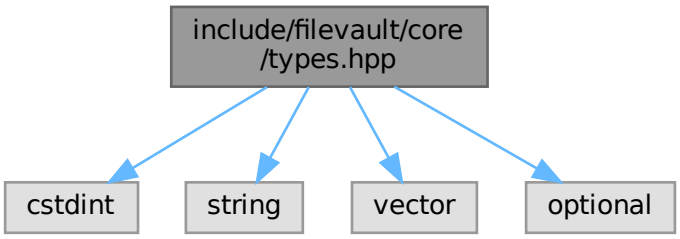
```
00001 #ifndef FILEVAULT_CORE_STREAMING_HPP
00002 #define FILEVAULT_CORE_STREAMING_HPP
00003
00004 #include <cstdint>
00005 #include <vector>
00006 #include <string>
00007 #include <functional>
00008 #include <fstream>
00009 #include "types.hpp"
00010 #include "result.hpp"
00011
00012 namespace filevault {
00013 namespace core {
00014
00018 struct ChunkInfo {
00019     size_t chunk_index;
00020     size_t chunk_size;
00021     size_t total_chunks;
00022     size_t bytes_processed;
00023     size_t total_bytes;
00024 };
00025
```

```
00031 using StreamProgressCallback = std::function<bool(const ChunkInfo& info)>;
00032
00036 struct StreamingConfig {
00037     size_t chunk_size = 64 * 1024 * 1024;  // 64MB default chunk size
00038     AlgorithmType algorithm = AlgorithmType::AES_256_GCM;
00039     KDFType kdf = KDFType::ARGON2ID;
00040     SecurityLevel level = SecurityLevel::STRONG;
00041     CompressionType compression = CompressionType::NONE;
00042     int compression_level = 6;
00043     StreamProgressCallback progress_callback = nullptr;
00044 };
00045
00049 struct StreamingResult {
00050     bool success = false;
00051     std::string error_message;
00052     size_t bytes_processed = 0;
00053     size_t chunks_processed = 0;
00054     double processing_time_ms = 0.0;
00055     double throughput_mbps = 0.0;
00056 };
00057
00071 class StreamingCrypto {
00072 public:
00081     static StreamingResult encrypt_file(
00082         const std::string& input_path,
00083         const std::string& output_path,
00084         const std::string& password,
00085         const StreamingConfig& config = {}
00086     );
00087
00096     static StreamingResult decrypt_file(
00097         const std::string& input_path,
00098         const std::string& output_path,
00099         const std::string& password,
00100         StreamProgressCallback progress_callback = nullptr
00101     );
00102
00109     static bool should_use_streaming(
00110         const std::string& file_path,
00111         size_t threshold = 100 * 1024 * 1024
00112     );
00113
00118     static size_t get_recommended_chunk_size();
00119
00120 private:
00124     static std::vector<uint8_t> derive_chunk_nonce(
00125         const std::vector<uint8_t>& base_nonce,
00126         size_t chunk_index
00127     );
00128
00132     static bool write_stream_header(
00133         std::ofstream& file,
00134         const StreamingConfig& config,
00135         const std::vector<uint8_t>& salt,
00136         const std::vector<uint8_t>& base_nonce,
00137         size_t total_size,
00138         size_t chunk_count
00139     );
00140
00144     static bool read_stream_header(
00145         std::ifstream& file,
00146         StreamingConfig& config,
00147         std::vector<uint8_t>& salt,
00148         std::vector<uint8_t>& base_nonce,
00149         size_t& original_size,
00150         size_t& chunk_count
00151     );
00152 };
00153
00154 } // namespace core
00155 } // namespace filevault
00156
00157 #endif // FILEVAULT_CORE_STREAMING_HPP
```

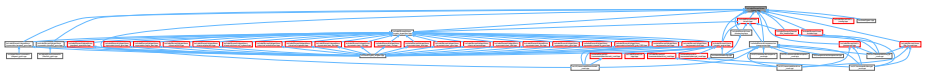
10.99 include/filevault/core/types.hpp File Reference

```
#include <cstdint>
#include <string>
#include <vector>
#include <optional>
```

Include dependency graph for types.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct filevault::core::EncryptionConfig

  Configuration for encryption operations.
- struct filevault::core::HashConfig

  Configuration for hashing operations.
- struct filevault::core::PasswordAnalysis

  Password strength analysis result.

Namespaces

- namespace filevault
- namespace filevault::core

Enumerations

- enum class filevault::core::AlgorithmType {
  filevault::core::AES_128_GCM , filevault::core::AES_192_GCM , filevault::core::AES_256_GCM , filevault::core::CHACHA20_P
  ,
  filevault::core::SERPENT_256_GCM , filevault::core::TWOFISH_128_GCM , filevault::core::TWOFISH_192_GCM
  , filevault::core::TWOFISH_256_GCM ,
  filevault::core::CAMELLIA_128_GCM , filevault::core::CAMELLIA_192_GCM , filevault::core::CAMELLIA_256_GCM
  , filevault::core::ARIA_128_GCM ,
  filevault::core::ARIA_192_GCM , filevault::core::ARIA_256_GCM , filevault::core::SM4_GCM , filevault::core::AES_128_CBC
  ,
  filevault::core::AES_192_CBC , filevault::core::AES_256_CBC , filevault::core::AES_128_CTR , filevault::core::AES_192_CTR
  ,
  filevault::core::AES_256_CTR , filevault::core::AES_128_CFB , filevault::core::AES_192_CFB , filevault::core::AES_256_CFB
  ,
  filevault::core::AES_128_OFB , filevault::core::AES_192_OFB , filevault::core::AES_256_OFB , filevault::core::AES_128_ECB
  ,
  filevault::core::AES_192_ECB , filevault::core::AES_256_ECB , filevault::core::AES_128_XTS , filevault::core::AES_256_XTS
  ,
  filevault::core::TRIPLE_DES_CBC , filevault::core::RSA_2048 , filevault::core::RSA_3072 , filevault::core::RSA_4096

,
filevault::core::ECC_P256 , filevault::core::ECC_P384 , filevault::core::ECC_P521 , filevault::core::CAESAR , filevault::core::VIGENERE , filevault::core::PLAYFAIR , filevault::core::SUBSTITUTION , filevault::core::HILL , filevault::core::KYBER_512 , filevault::core::KYBER_768 , filevault::core::KYBER_1024 , filevault::core::DILITHIUM_2 , filevault::core::DILITHIUM_3 , filevault::core::DILITHIUM_5 , filevault::core::KYBER_512_HYBRID , filevault::core::KYBER_768_HYBRID , filevault::core::KYBER_1024_HYBRID }

Enumeration of encryption algorithm types.

- enum class filevault::core::HashType { filevault::core::MD5 , filevault::core::SHA1 , filevault::core::SHA224 , filevault::core::SHA256 , filevault::core::SHA384 , filevault::core::SHA512 , filevault::core::SHA512_256 , filevault::core::SHA3_224 , filevault::core::SHA3_256 , filevault::core::SHA3_384 , filevault::core::SHA3_512 , filevault::core::BLAKE2B_256 , filevault::core::BLAKE2B_384 , filevault::core::BLAKE2B_512 , filevault::core::BLAKE2S_256 }

  Enumeration of hash algorithm types.
- enum class filevault::core::CompressionType { filevault::core::NONE , filevault::core::ZLIB , filevault::core::BZIP2 , filevault::core::LZMA }

  Compression algorithm types.
- enum class filevault::core::UserMode { filevault::core::STUDENT , filevault::core::PROFESSIONAL , filevault::core::ADVANCED }

  User mode/profile for algorithm selection.
- enum class filevault::core::KDFType { filevault::core::ARGON2ID , filevault::core::ARGON2I , filevault::core::PBKDF2_SHA256 , filevault::core::PBKDF2_SHA512 , filevault::core::SCRYPT }

  Enumeration of Key Derivation Function types.
- enum class filevault::core::SecurityLevel { filevault::core::WEAK , filevault::core::MEDIUM , filevault::core::STRONG , filevault::core::PARANOID }

  Security level determining algorithm parameters.
- enum class filevault::core::PasswordStrength { filevault::core::VERY_WEAK , filevault::core::WEAK , filevault::core::FAIR , filevault::core::STRONG , filevault::core::VERY_STRONG }

  Password strength levels.

10.100 types.hpp

Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_CORE_TYPES_HPP
00002 #define FILEVAULT_CORE_TYPES_HPP
00003
00004 #include <cstdint>
00005 #include <string>
00006 #include <vector>
00007 #include <optional>
00008
00009 namespace filevault {
00010 namespace core {
00011
00015 enum class AlgorithmType {
00016     // Symmetric ciphers (modern AEAD)
00017     AES_128_GCM,
00018     AES_192_GCM,
00019     AES_256_GCM,
00020     CHACHA20_POLY1305,
00021     SERPENT_256_GCM,
00022     TWOFISH_128_GCM,
00023     TWOFISH_192_GCM,
00024     TWOFISH_256_GCM,
00025
00026     // Camellia (Japanese standard - ISO/IEC 18033-3)
00027     CAMELLIA_128_GCM,
00028     CAMELLIA_192_GCM,
00029     CAMELLIA_256_GCM,
```

```
00030
00031     // ARIA (Korean standard - ISO/IEC 18033-3)
00032     ARIA_128_GCM,
00033     ARIA_192_GCM,
00034     ARIA_256_GCM,
00035
00036     // SM4 (Chinese standard - GB/T 32907-2016)
00037     SM4_GCM,
00038
00039     // Symmetric ciphers (CBC mode - legacy, not authenticated)
00040     AES_128_CBC,
00041     AES_192_CBC,
00042     AES_256_CBC,
00043
00044     // Symmetric ciphers (CTR mode - stream, not authenticated)
00045     AES_128_CTR,
00046     AES_192_CTR,
00047     AES_256_CTR,
00048
00049     // Symmetric ciphers (CFB mode - self-synchronizing stream)
00050     AES_128_CFB,
00051     AES_192_CFB,
00052     AES_256_CFB,
00053
00054     // Symmetric ciphers (OFB mode - output feedback stream)
00055     AES_128_OFB,
00056     AES_192_OFB,
00057     AES_256_OFB,
00058
00059     // Symmetric ciphers (ECB mode - INSECURE, educational only)
00060     AES_128_ECB,
00061     AES_192_ECB,
00062     AES_256_ECB,
00063
00064     // Symmetric ciphers (XTS mode - disk encryption)
00065     AES_128_XTS,
00066     AES_256_XTS,
00067
00068     // Legacy algorithms (for compatibility only)
00069     TRIPLE_DES_CBC,
00070
00071     // Asymmetric encryption (RSA)
00072     RSA_2048,
00073     RSA_3072,
00074     RSA_4096,
00075
00076     // Asymmetric encryption (ECC)
00077     ECC_P256,   // secp256r1, 128-bit security
00078     ECC_P384,   // secp384r1, 192-bit security
00079     ECC_P521,   // secp521r1, 256-bit security
00080
00081     // Classic (educational)
00082     CAESAR,
00083     VIGENERE,
00084     PLAYFAIR,
00085     SUBSTITUTION,
00086     HILL,
00087
00088     // Post-Quantum Key Encapsulation (NIST FIPS 203 - ML-KEM)
00089     KYBER_512,      // ~AES-128 equivalent security
00090     KYBER_768,      // ~AES-192 equivalent security
00091     KYBER_1024,     // ~AES-256 equivalent security
00092
00093     // Post-Quantum Digital Signatures (NIST FIPS 204 - ML-DSA)
00094     DILITHIUM_2,    // ~AES-128 equivalent security
00095     DILITHIUM_3,    // ~AES-192 equivalent security
00096     DILITHIUM_5,    // ~AES-256 equivalent security
00097
00098     // Hybrid Post-Quantum (Classic + PQC for transition period)
00099     KYBER_512_HYBRID,   // Kyber-512 + X25519
00100     KYBER_768_HYBRID,   // Kyber-768 + X25519
00101     KYBER_1024_HYBRID   // Kyber-1024 + X25519
00102 };
00103
00107 enum class HashType {
00108     // Legacy (insecure - for compatibility only)
00109     MD5,
00110     SHA1,
00111
00112     // SHA-2 family (secure)
00113     SHA224,
00114     SHA256,
00115     SHA384,
00116     SHA512,
00117     SHA512_256,  // SHA-512/256
00118
00119     // SHA-3 family (secure)
```

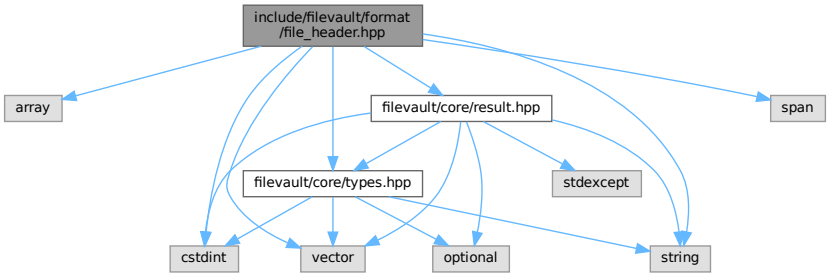
```
00120     SHA3_224,
00121     SHA3_256,
00122     SHA3_384,
00123     SHA3_512,
00124
00125     // BLAKE2 family (secure, fast)
00126     BLAKE2B_256,
00127     BLAKE2B_384,
00128     BLAKE2B_512,
00129     BLAKE2S_256
00130 };
00131
00135 enum class CompressionType {
00136     NONE,
00137     ZLIB,
00138     BZIP2,
00139     LZMA
00140 };
00141
00145 enum class UserMode {
00146     STUDENT,        // Educational - classical ciphers
00147     PROFESSIONAL,   // Standard - AES-256, Argon2
00148     ADVANCED        // Maximum security - custom params
00149 };
00150
00154 enum class KDFType {
00155     ARGON2ID,
00156     ARGON2I,
00157     PBKDF2_SHA256,
00158     PBKDF2_SHA512,
00159     SCRYPT
00160 };
00161
00165 enum class SecurityLevel {
00166     WEAK,       // Fast, for testing (KDF: 10k iterations, 16MB memory)
00167     MEDIUM,     // Balanced (KDF: 100k iterations, 64MB memory)
00168     STRONG,     // High security (KDF: 200k iterations, 128MB memory)
00169     PARANOID    // Maximum (KDF: 500k iterations, 256MB memory)
00170 };
00171
00175 struct EncryptionConfig {
00176     AlgorithmType algorithm = AlgorithmType::AES_256_GCM;
00177     KDFType kdf = KDFType::ARGON2ID;
00178     SecurityLevel level = SecurityLevel::MEDIUM;
00179     UserMode mode = UserMode::PROFESSIONAL;
00180
00181     // KDF parameters (auto-set based on SecurityLevel)
00182     uint32_t kdf_iterations = 100000;
00183     uint32_t kdf_memory_kb = 65536;  // 64MB default
00184     uint32_t kdf_parallelism = 4;
00185
00186     // Encryption parameters (generated automatically or provided)
00187     std::vector<uint8_t> salt;
00188     std::optional<std::vector<uint8_t>> nonce;
00189     std::optional<std::vector<uint8_t>> tag;
00190     std::optional<std::vector<uint8_t>> associated_data;
00191
00192     // Compression
00193     CompressionType compression = CompressionType::NONE;
00194     int compression_level = 6;  // 1-9 for zlib/bzip2/lzma
00195
00196     // Metadata
00197     bool include_metadata = true;
00198     std::string comment;
00199
00200     // Progress reporting
00201     bool show_progress = true;
00202     bool verbose = false;
00203
00207     void apply_security_level();
00208
00212     void apply_user_mode();
00213 };
00214
00218 struct HashConfig {
00219     HashType algorithm = HashType::SHA256;
00220     bool hmac_mode = false;
00221     std::vector<uint8_t> hmac_key;
00222
00223     // For file hashing
00224     bool verify_mode = false;
00225     std::string expected_hash;
00226
00227     // Output format
00228     bool uppercase = false;
00229     bool include_filename = true;
00230 };
```

```
00231
00235 enum class PasswordStrength {
00236     VERY_WEAK,
00237     WEAK,
00238     FAIR,
00239     STRONG,
00240     VERY_STRONG
00241 };
00242
00246 struct PasswordAnalysis {
00247     PasswordStrength strength;
00248     int score;  // 0-100
00249     std::vector<std::string> warnings;
00250     std::vector<std::string> suggestions;
00251
00252     // Detailed metrics
00253     size_t length;
00254     bool has_lowercase;
00255     bool has_uppercase;
00256     bool has_digits;
00257     bool has_special;
00258     bool has_repeated_chars;
00259     bool is_common_password;
00260
00261     // Estimated crack time
00262     std::string crack_time_online;   // "< 1 second" or "centuries"
00263     std::string crack_time_offline;  // "< 1 second" or "centuries"
00264 };
00265
00266 } // namespace core
00267 } // namespace filevault
00268
00269 #endif // FILEVAULT_CORE_TYPES_HPP
```

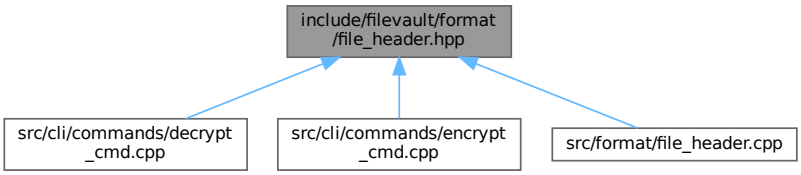
10.101 include/filevault/format/file_header.hpp File Reference

```
#include <array>
#include <cstdint>
#include <vector>
#include <string>
#include <span>
#include "filevault/core/types.hpp"
#include "filevault/core/result.hpp"
```

Include dependency graph for file_header.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::format::FileHeader

  FileVault encrypted file format.

Namespaces

- namespace filevault
- namespace filevault::format

10.102 file_header.hpp

Go to the documentation of this file.

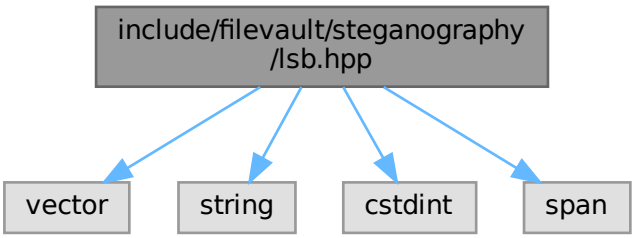
```
00001 #ifndef FILEVAULT_FORMAT_FILE_HEADER_HPP
00002 #define FILEVAULT_FORMAT_FILE_HEADER_HPP
00003
00004 #include <array>
00005 #include <cstdint>
00006 #include <vector>
00007 #include <string>
00008 #include <span>
00009 #include "filevault/core/types.hpp"
00010 #include "filevault/core/result.hpp"
00011
00012 namespace filevault {
00013 namespace format {
00014
00037 class FileHeader {
00038 public:
00039     static constexpr uint32_t MAGIC = 0x544C5646; // "FVLT" in little-endian
00040     static constexpr uint8_t VERSION_MAJOR = 1;
00041     static constexpr uint8_t VERSION_MINOR = 0;
00042     static constexpr size_t MIN_HEADER_SIZE = 64;
00043
00044     // Flags
00045     static constexpr uint32_t FLAG_COMPRESSED = 0x00000001;
00046     static constexpr uint32_t FLAG_METADATA   = 0x00000002;
00047
00048     FileHeader();
00049
00050     // Setters
00051     void set_algorithm(core::AlgorithmType algo) { algorithm_ = algo; }
00052     void set_kdf(core::KDFType kdf) { kdf_ = kdf; }
00053     void set_security_level(core::SecurityLevel level) { security_level_ = level; }
00054     void set_salt(const std::vector<uint8_t>& salt) { salt_ = salt; }
00055     void set_nonce(const std::vector<uint8_t>& nonce) { nonce_ = nonce; }
00056     void set_tag(const std::vector<uint8_t>& tag) { tag_ = tag; }
00057     void set_original_size(uint64_t size) { original_size_ = size; }
00058     void set_encrypted_size(uint64_t size) { encrypted_size_ = size; }
00059     void set_timestamp(uint64_t ts) { timestamp_ = ts; }
00060     void set_compressed(bool compressed);
00061
00062     // Getters
00063     core::AlgorithmType algorithm() const { return algorithm_; }
00064     core::KDFType kdf() const { return kdf_; }
00065     core::SecurityLevel security_level() const { return security_level_; }
00066     const std::vector<uint8_t>& salt() const { return salt_; }
00067     const std::vector<uint8_t>& nonce() const { return nonce_; }
00068     const std::vector<uint8_t>& tag() const { return tag_; }
```

```
00069     uint64_t original_size() const { return original_size_; }
00070     uint64_t encrypted_size() const { return encrypted_size_; }
00071     uint64_t timestamp() const { return timestamp_; }
00072     bool is_compressed() const { return (flags_ & FLAG_COMPRESSED) != 0; }
00073     uint32_t flags() const { return flags_; }
00074
00075     // Serialization
00076     std::vector<uint8_t> serialize() const;
00077     static core::Result<FileHeader> deserialize(std::span<const uint8_t> data);
00078
00079     // Validation
00080     bool validate() const;
00081     size_t total_size() const;
00082
00083 private:
00084     core::AlgorithmType algorithm_;
00085     core::KDFType kdf_;
00086     core::SecurityLevel security_level_;
00087     std::vector<uint8_t> salt_;
00088     std::vector<uint8_t> nonce_;
00089     std::vector<uint8_t> tag_;
00090     uint64_t original_size_;
00091     uint64_t encrypted_size_;
00092     uint64_t timestamp_;
00093     uint32_t flags_;
00094     std::array<uint8_t, 16> reserved_;
00095
00096     // Helper functions
00097     static void write_uint16(std::vector<uint8_t>& buf, uint16_t val);
00098     static void write_uint32(std::vector<uint8_t>& buf, uint32_t val);
00099     static void write_uint64(std::vector<uint8_t>& buf, uint64_t val);
00100     static uint16_t read_uint16(std::span<const uint8_t> data, size_t& offset);
00101     static uint32_t read_uint32(std::span<const uint8_t> data, size_t& offset);
00102     static uint64_t read_uint64(std::span<const uint8_t> data, size_t& offset);
00103 };
00104
00105 } // namespace format
00106 } // namespace filevault
00107
00108 #endif // FILEVAULT_FORMAT_FILE_HEADER_HPP
```

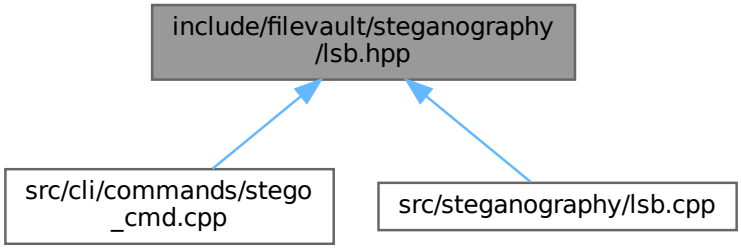
10.103 include/filevault/steganography/lsb.hpp File Reference

```
#include <vector>
#include <string>
#include <cstdint>
#include <span>
```

Include dependency graph for lsb.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::steganography::LSBSteganography

  LSB (Least Significant Bit) Steganography.

Namespaces

- namespace filevault
- namespace filevault::steganography

10.104 lsb.hpp

Go to the documentation of this file.

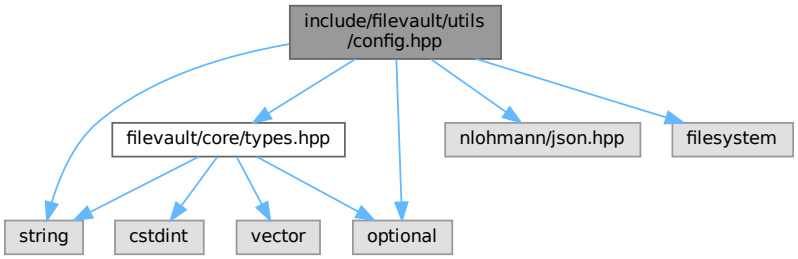
```
00001 #ifndef FILEVAULT_STEGANOGRAPHY_LSB_HPP
00002 #define FILEVAULT_STEGANOGRAPHY_LSB_HPP
00003
00004 #include <vector>
00005 #include <string>
00006 #include <cstdint>
00007 #include <span>
00008
00009 namespace filevault::steganography {
00010
00020 class LSBSteganography {
00021 public:
00031     static bool embed(
00032         const std::string& cover_image_path,
00033         std::span<const uint8_t> secret_data,
00034         const std::string& output_path,
00035         int bits_per_channel = 1
00036     );
00037
00045     static std::vector<uint8_t> extract(
00046         const std::string& stego_image_path,
00047         int bits_per_channel = 1
00048     );
00049
00057     static size_t calculate_capacity(
00058         const std::string& image_path,
00059         int bits_per_channel = 1
00060     );
00061
00062 private:
00063     // Embed length header (4 bytes) at the beginning
00064     static constexpr size_t LENGTH_HEADER_SIZE = 4;
00065
00066     // Helper to embed one byte into image data
00067     static void embed_byte(
00068         uint8_t* pixel_data,
00069         size_t pixel_count,
00070         size_t& bit_index,
00071         uint8_t byte,
00072         int bits_per_channel
```

```
00073     );
00074
00075     // Helper to extract one byte from image data
00076     static uint8_t extract_byte(
00077         const uint8_t* pixel_data,
00078         size_t pixel_count,
00079         size_t& bit_index,
00080         int bits_per_channel
00081     );
00082 };
00083
00084 } // namespace filevault::steganography
00085
00086 #endif // FILEVAULT_STEGANOGRAPHY_LSB_HPP
```

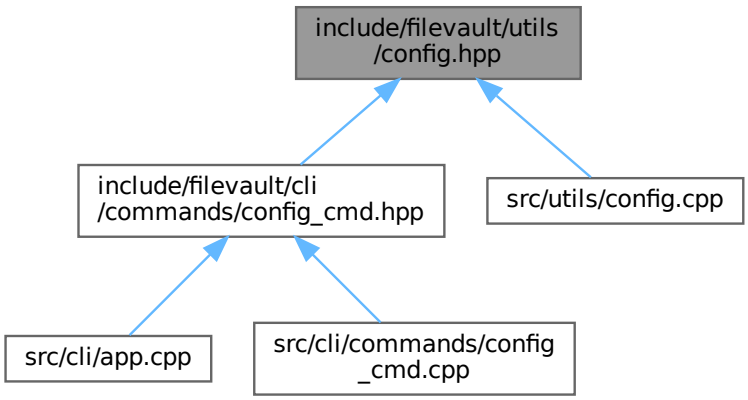
10.105 include/filevault/utils/config.hpp File Reference

```
#include "filevault/core/types.hpp"
#include <nlohmann/json.hpp>
#include <string>
#include <optional>
#include <filesystem>
```

Include dependency graph for config.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::utils::Config

  Configuration manager for FileVault.

Namespaces

- namespace filevault
- namespace filevault::utils

10.106 config.hpp

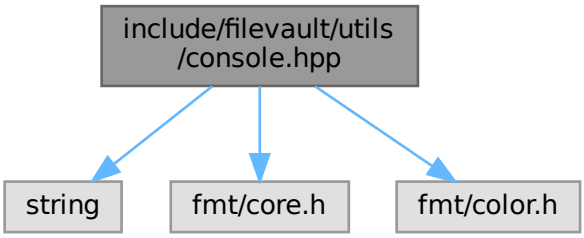
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_UTILS_CONFIG_HPP
00002 #define FILEVAULT_UTILS_CONFIG_HPP
00003
00004 #include "filevault/core/types.hpp"
00005 #include <nlohmann/json.hpp>
00006 #include <string>
00007 #include <optional>
00008 #include <filesystem>
00009
00010 namespace filevault {
00011 namespace utils {
00012
00018 class Config {
00019 public:
00023     static Config load();
00024
00028     bool save() const;
00029
00033     static std::filesystem::path get_config_path();
00034
00038     static Config get_default();
00039
00043     void reset();
00044
00045     // Getters
00046     std::string get_default_mode() const { return default_mode_; }
00047     std::string get_default_algorithm() const { return default_algorithm_; }
00048     std::string get_default_kdf() const { return default_kdf_; }
00049     std::string get_default_compression() const { return default_compression_; }
00050     int get_compression_level() const { return compression_level_; }
00051     bool get_show_progress() const { return show_progress_; }
00052     bool get_verbose() const { return verbose_; }
00053
00054     // Setters
00055     void set_default_mode(const std::string& mode) { default_mode_ = mode; }
00056     void set_default_algorithm(const std::string& algo) { default_algorithm_ = algo; }
00057     void set_default_kdf(const std::string& kdf) { default_kdf_ = kdf; }
00058     void set_default_compression(const std::string& comp) { default_compression_ = comp; }
00059     void set_compression_level(int level) { compression_level_ = level; }
00060     void set_show_progress(bool show) { show_progress_ = show; }
00061     void set_verbose(bool verbose) { verbose_ = verbose; }
00062
00066     std::optional<std::string> get(const std::string& key) const;
00067
00071     bool set(const std::string& key, const std::string& value);
00072
00076     nlohmann::json to_json() const;
00077
00081     static Config from_json(const nlohmann::json& j);
00082
00083 private:
00084     // Default settings
00085     std::string default_mode_ = "standard";
00086     std::string default_algorithm_ = "aes-256-gcm";
00087     std::string default_kdf_ = "argon2id";
00088     std::string default_compression_ = "none";
00089     int compression_level_ = 6;
00090
00091     // UI preferences
00092     bool show_progress_ = true;
00093     bool verbose_ = false;
00094 };
00095
00096 } // namespace utils
00097 } // namespace filevault
00098
00099 #endif // FILEVAULT_UTILS_CONFIG_HPP
```

10.107 include/filevault/utils/console.hpp File Reference

```
#include <string>
#include <fmt/core.h>
#include <fmt/color.h>
```

Include dependency graph for console.hpp:



This graph shows which files directly or indirectly include this file:

Classes

- class filevault::utils::Console

  Console utilities for colored output.

Namespaces

- namespace filevault
- namespace filevault::utils

10.108 console.hpp

Go to the documentation of this file.

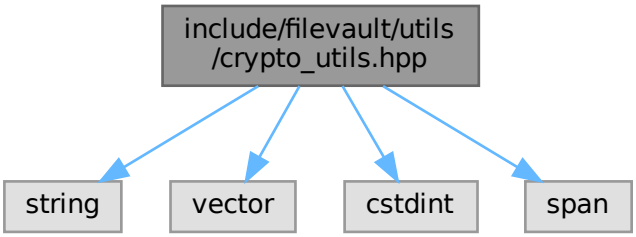
```
00001 #ifndef FILEVAULT_UTILS_CONSOLE_HPP
00002 #define FILEVAULT_UTILS_CONSOLE_HPP
00003
00004 #include <string>
00005 #include <fmt/core.h>
00006 #include <fmt/color.h>
00007
00008 namespace filevault {
00009 namespace utils {
00010
00014 class Console {
00015 public:
00016     static void success(const std::string& msg);
00017     static void error(const std::string& msg);
00018     static void warning(const std::string& msg);
00019     static void info(const std::string& msg);
00020     static void debug(const std::string& msg);
00021
00022     static void separator(char ch = '=', size_t width = 80);
00023     static void header(const std::string& title);
00024 };
00025
00026 } // namespace utils
00027 } // namespace filevault
```

```
00028
00029 #endif // FILEVAULT_UTILS_CONSOLE_HPP
```

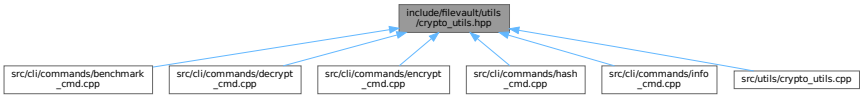
10.109 include/filevault/utils/crypto_utils.hpp File Reference

```
#include <string>
#include <vector>
#include <cstdint>
#include <span>
```

Include dependency graph for crypto_utils.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::utils::CryptoUtils

  Crypto utility functions.

Namespaces

- namespace filevault
- namespace filevault::utils

10.110 crypto_utils.hpp

Go to the documentation of this file.

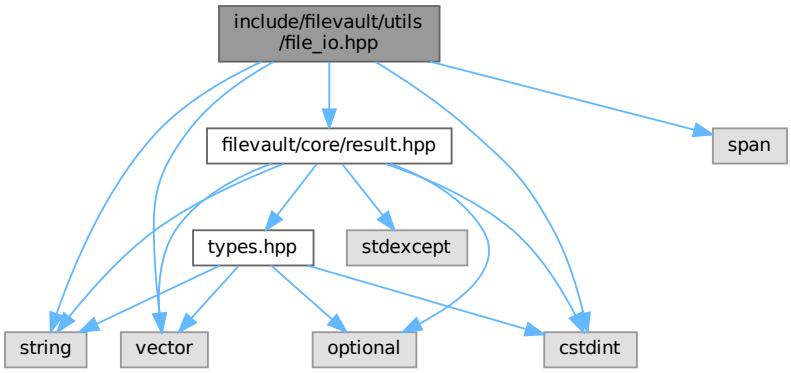
```
00001 #ifndef FILEVAULT_UTILS_CRYPTO_UTILS_HPP
00002 #define FILEVAULT_UTILS_CRYPTO_UTILS_HPP
00003
00004 #include <string>
00005 #include <vector>
00006 #include <cstdint>
00007 #include <span>
00008
00009 namespace filevault {
00010 namespace utils {
00011
00015 class CryptoUtils {
```

```
00016 public:
00020     static std::string hex_encode(std::span<const uint8_t> data);
00021
00025     static std::vector<uint8_t> hex_decode(const std::string& hex);
00026
00030     static std::string format_bytes(size_t bytes);
00031 };
00032
00033 } // namespace utils
00034 } // namespace filevault
00035
00036 #endif // FILEVAULT_UTILS_CRYPTO_UTILS_HPP
```

10.111 include/filevault/utils/file_io.hpp File Reference

```
#include <string>
#include <vector>
#include <cstdint>
#include <span>
#include "filevault/core/result.hpp"
```

Include dependency graph for file_io.hpp:



This graph shows which files directly or indirectly include this file:

Classes

- class filevault::utils::FileIO

  File I/O utilities.

Namespaces

- namespace filevault
- namespace filevault::utils

10.112 file_io.hpp

Go to the documentation of this file.

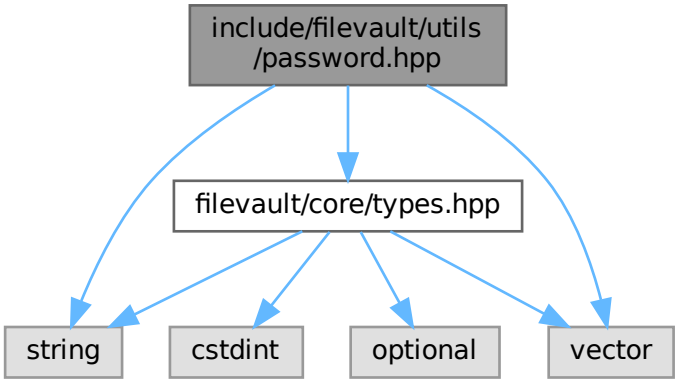
```
00001 #ifndef FILEVAULT_UTILS_FILE_IO_HPP
```

```
00002 #define FILEVAULT_UTILS_FILE_IO_HPP
00003
00004 #include <string>
00005 #include <vector>
00006 #include <cstdint>
00007 #include <span>
00008 #include "filevault/core/result.hpp"
00009
00010 namespace filevault {
00011 namespace utils {
00012
00016 class FileIO {
00017 public:
00021     static core::Result<std::vector<uint8_t>> read_file(const std::string& path);
00022
00026     static core::Result<void> write_file(const std::string& path, std::span<const uint8_t> data);
00027
00031     static bool file_exists(const std::string& path);
00032
00036     static size_t file_size(const std::string& path);
00037 };
00038
00039 } // namespace utils
00040 } // namespace filevault
00041
00042 #endif // FILEVAULT_UTILS_FILE_IO_HPP
```

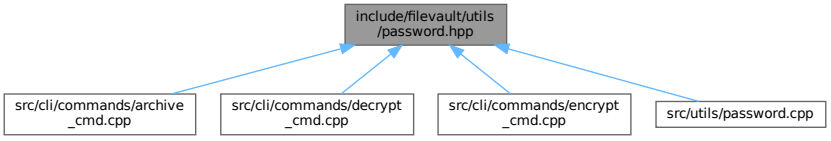
10.113 include/filevault/utils/password.hpp File Reference

```
#include "filevault/core/types.hpp"
#include <string>
#include <vector>
```

Include dependency graph for password.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::utils::Password

  Password utilities for secure input and strength analysis.

Namespaces

- namespace filevault
- namespace filevault::utils

10.114 password.hpp

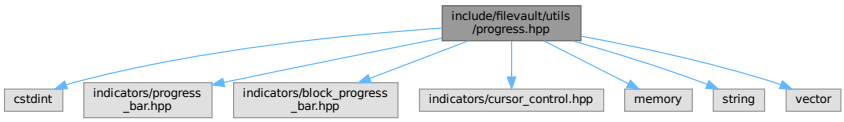
Go to the documentation of this file.

```
00001 #ifndef FILEVAULT_UTILS_PASSWORD_HPP
00002 #define FILEVAULT_UTILS_PASSWORD_HPP
00003
00004 #include "filevault/core/types.hpp"
00005 #include <string>
00006 #include <vector>
00007
00008 namespace filevault {
00009 namespace utils {
00010
00014 class Password {
00015 public:
00022     static std::string read_secure(const std::string& prompt = "Password: ",
00023                                    bool confirm = false);
00024
00030     static core::PasswordAnalysis analyze_strength(const std::string& password);
00031
00035     static std::string get_strength_color(core::PasswordStrength strength);
00036
00040     static std::string get_strength_label(core::PasswordStrength strength);
00041
00045     static void display_strength_meter(const core::PasswordAnalysis& analysis);
00046
00047 private:
00051     static void disable_echo();
00052
00056     static void enable_echo();
00057
00061     static bool is_common_password(const std::string& password);
00062
00066     static double calculate_entropy(const std::string& password);
00067
00071     static std::pair<std::string, std::string> estimate_crack_time(
00072         double entropy,
00073         const core::PasswordAnalysis& analysis
00074     );
00075
00076     // Top 100 most common passwords
00077     static const std::vector<std::string> common_passwords_;
00078 };
00079
00080 } // namespace utils
00081 } // namespace filevault
00082
00083 #endif // FILEVAULT_UTILS_PASSWORD_HPP
```

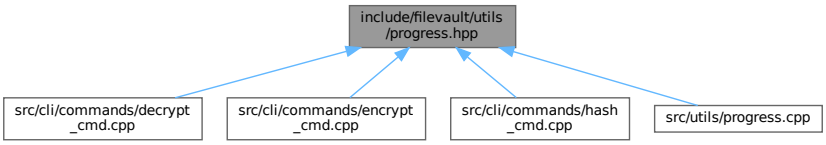
10.115 include/filevault/utils/progress.hpp File Reference

```
#include <cstdint>
#include <indicators/progress_bar.hpp>
#include <indicators/block_progress_bar.hpp>
#include <indicators/cursor_control.hpp>
#include <memory>
#include <string>
#include <vector>
```

Include dependency graph for progress.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::utils::ProgressBar

  Modern progress bar wrapper.
- class filevault::utils::BlockProgressBar

  Block-style progress bar.

Namespaces

- namespace filevault
- namespace filevault::utils

10.116 progress.hpp

Go to the documentation of this file.

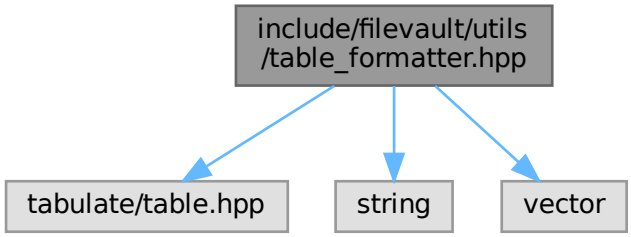
```
00001 #pragma once
00002
00003 #include <cstdint>
00004 #include <indicators/progress_bar.hpp>
00005 #include <indicators/block_progress_bar.hpp>
00006 #include <indicators/cursor_control.hpp>
00007 #include <memory>
00008 #include <string>
00009 #include <vector>
00010
00011 namespace filevault {
00012 namespace utils {
00013
00017 class ProgressBar {
00018 public:
00019     ProgressBar(const std::string& prefix, size_t max_progress = 100);
00020     ~ProgressBar();
00021
00022     void set_progress(size_t progress);
00023     void tick();
00024     void set_postfix(const std::string& text);
00025     void mark_as_completed();
00026
00027     void hide();
00028     void show();
00029
00030 private:
00031     std::unique_ptr<indicators::ProgressBar> bar_;
00032     size_t current_progress_;
00033     size_t max_progress_;
```

```
00034 };
00035
00039 class BlockProgressBar {
00040 public:
00041     BlockProgressBar(const std::string& prefix, size_t max_progress = 100);
00042
00043     void set_progress(size_t progress);
00044     void set_option_text(const std::string& text);
00045     void mark_as_completed();
00046
00047 private:
00048     std::unique_ptr<indicators::BlockProgressBar> bar_;
00049 };
00050
00051 // MultiProgress removed - not compatible with indicators API
00052
00053 // Spinner removed - use ProgressBar instead
00054
00055 } // namespace utils
00056 } // namespace filevault
```

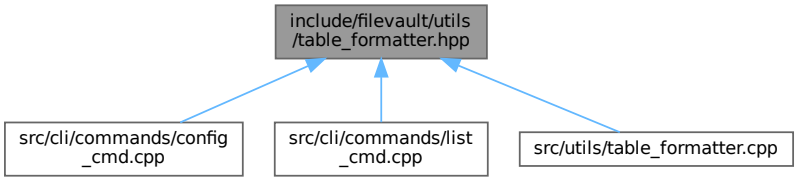
10.117 include/filevault/utils/table_formatter.hpp File Reference

```
#include <tabulate/table.hpp>
#include <string>
#include <vector>
```

Include dependency graph for table_formatter.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class filevault::utils::TableFormatter

  Beautiful table formatter.
- class filevault::utils::TableBuilder

  Quick table builders.

Namespaces

- namespace filevault
- namespace filevault::utils

10.118 table_formatter.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <tabulate/table.hpp>
00004 #include <string>
00005 #include <vector>
00006
00007 namespace filevault {
00008 namespace utils {
00009
00013 class TableFormatter {
00014 public:
00015     TableFormatter(const std::vector<std::string>& headers);
00016
00017     void add_row(const std::vector<std::string>& row);
00018     void print();
00019     std::string to_string();
00020
00021     // Styling
00022     void set_border_style(const std::string& style);  // "ascii", "unicode", "markdown"
00023     void set_column_alignment(size_t col_index, tabulate::FontAlign align);
00024     void set_column_format(size_t col_index, tabulate::Color color);
00025
00026     // Pre-built themes
00027     static TableFormatter algorithm_list_table();
00028     static TableFormatter benchmark_table();
00029     static TableFormatter file_info_table();
00030
00031 private:
00032     tabulate::Table table_;
00033 };
00034
00038 class TableBuilder {
00039 public:
00040     static void print_algorithm_list(
00041         const std::vector<std::tuple<std::string, std::string, std::string>& algorithms
00042     );
00043
00044     static void print_benchmark_results(
00045         const std::vector<std::tuple<std::string, double, double>& results
00046     );
00047
00048     static void print_file_summary(
00049         const std::string& input_file,
00050         const std::string& output_file,
00051         size_t input_size,
00052         size_t output_size,
00053         double processing_time_ms
00054     );
00055
00056     static void print_encryption_config(
00057         const std::string& algorithm,
00058         const std::string& kdf,
00059         const std::string& security_level,
00060         size_t kdf_iterations,
00061         size_t kdf_memory_kb
00062     );
00063 };
00064
00065 } // namespace utils
00066 } // namespace filevault
```

10.119 README.md File Reference

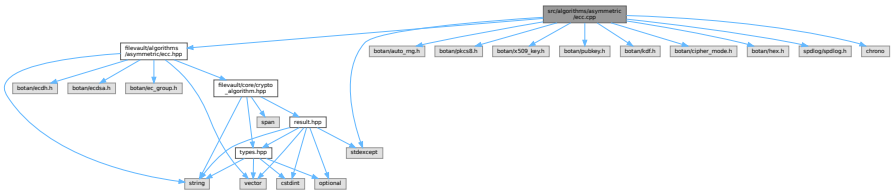
10.120 src/algorithms/asymmetric/ecc.cpp File Reference

Elliptic Curve Cryptography implementation.

```
#include "filevault/algorithms/asymmetric/ecc.hpp"
#include <botan/auto_rng.h>
```

```
#include <botan/pkcs8.h>
#include <botan/x509_key.h>
#include <botan/pubkey.h>
#include <botan/kdf.h>
#include <botan/cipher_mode.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
#include <stdexcept>
```

Include dependency graph for ecc.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::asymmetric

Functions

- static std::string filevault::algorithms::asymmetric::get_botan_curve_name (ECCurve curve)
- static size_t filevault::algorithms::asymmetric::get_curve_key_size (ECCurve curve)

10.120.1 Detailed Description

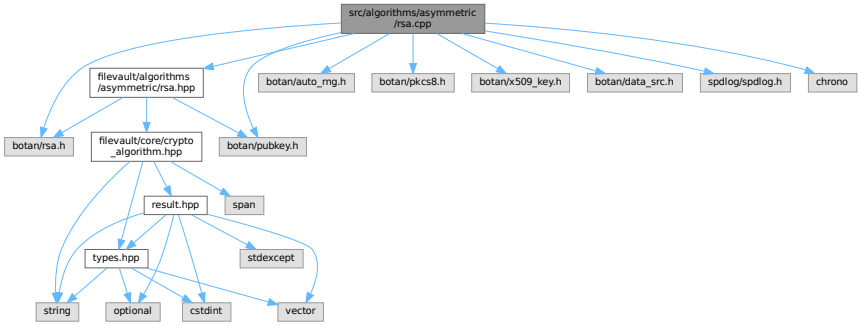
Elliptic Curve Cryptography implementation.

10.121 src/algorithms/asymmetric/rsa.cpp File Reference

RSA asymmetric encryption implementation.

```
#include "filevault/algorithms/asymmetric/rsa.hpp"
#include <botan/auto_rng.h>
#include <botan/rsa.h>
#include <botan/pubkey.h>
#include <botan/pkcs8.h>
#include <botan/x509_key.h>
#include <botan/data_src.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for rsa.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::asymmetric

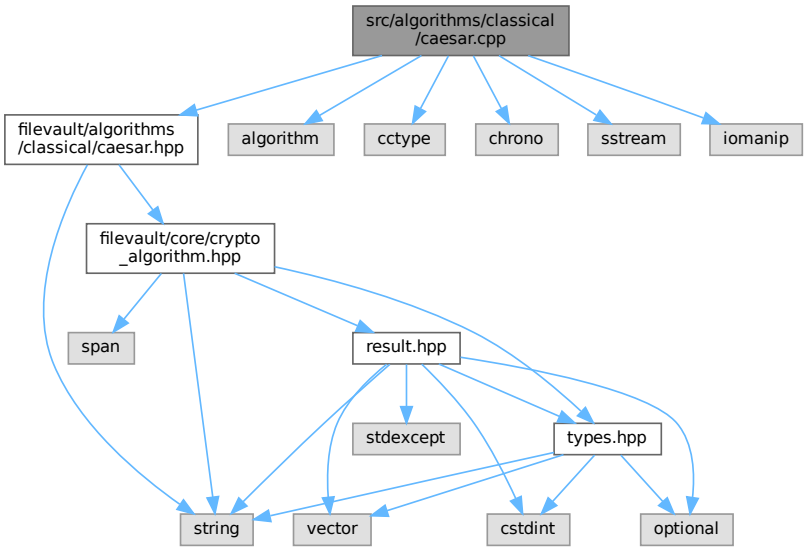
10.121.1 Detailed Description

RSA asymmetric encryption implementation.

10.122 src/algorithms/classical/caesar.cpp File Reference

```
#include "filevault/algorithms/classical/caesar.hpp"
#include <algorithm>
#include <cctype>
#include <chrono>
#include <sstream>
#include <iomanip>
```

Include dependency graph for caesar.cpp:



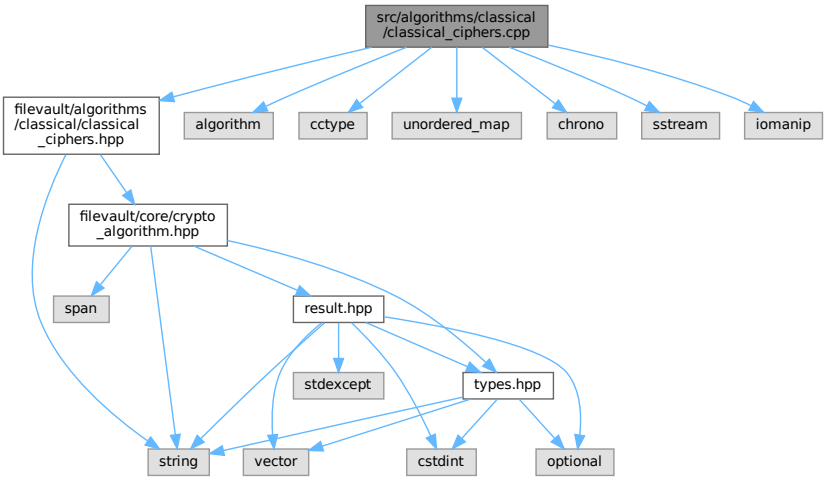
Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.123 src/algorithms/classical/classical_ciphers.cpp File Reference

```
#include "filevault/algorithms/classical/classical_ciphers.hpp"
#include <algorithm>
#include <cctype>
#include <unordered_map>
#include <chrono>
#include <sstream>
#include <iomanip>
```

Include dependency graph for classical_ciphers.cpp:



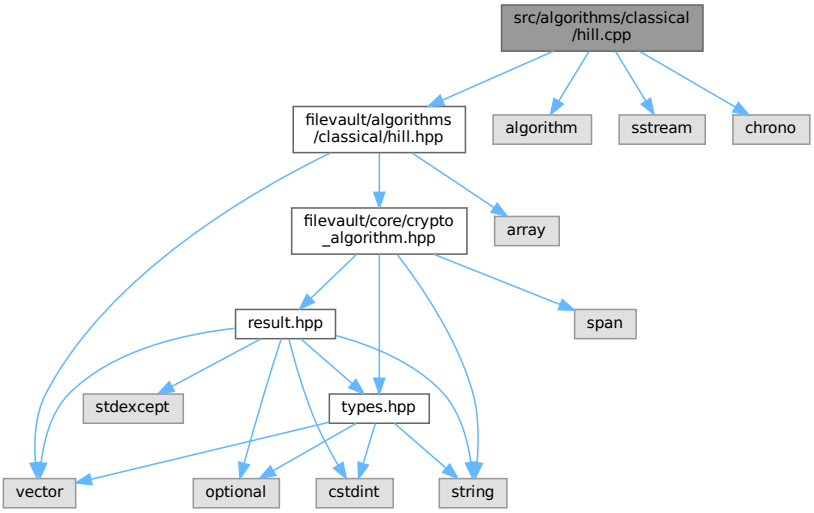
Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.124 src/algorithms/classical/hill.cpp File Reference

```
#include "filevault/algorithms/classical/hill.hpp"
#include <algorithm>
#include <sstream>
#include <chrono>
```

Include dependency graph for hill.cpp:



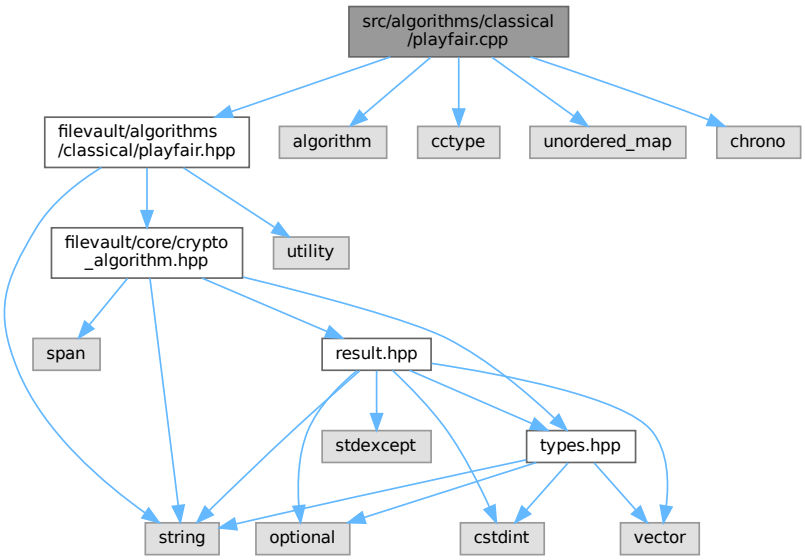
Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.125 src/algorithms/classical/playfair.cpp File Reference

```
#include "filevault/algorithms/classical/playfair.hpp"
#include <algorithm>
#include <cctype>
#include <unordered_map>
#include <chrono>
```

Include dependency graph for playfair.cpp:



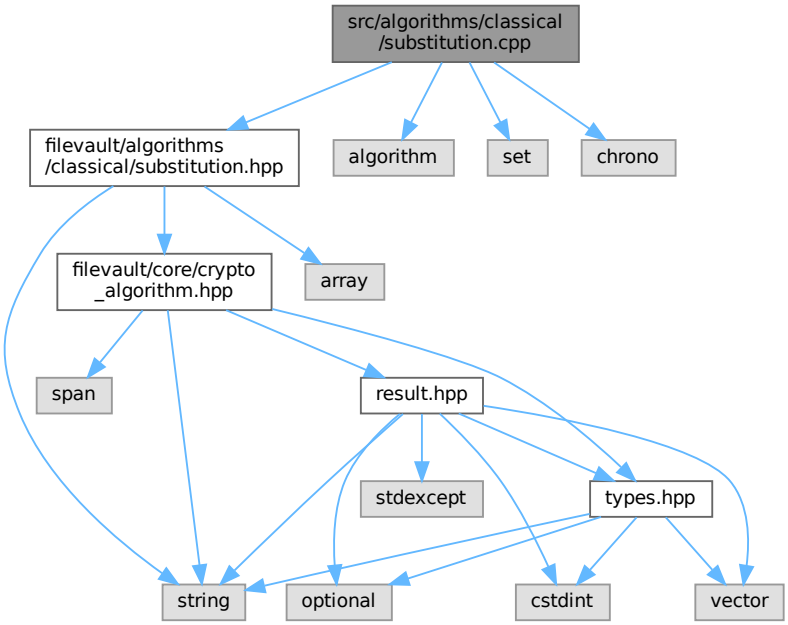
Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.126 src/algorithms/classical/substitution.cpp File Reference

```
#include "filevault/algorithms/classical/substitution.hpp"
#include <algorithm>
#include <set>
#include <chrono>
```

Include dependency graph for substitution.cpp:



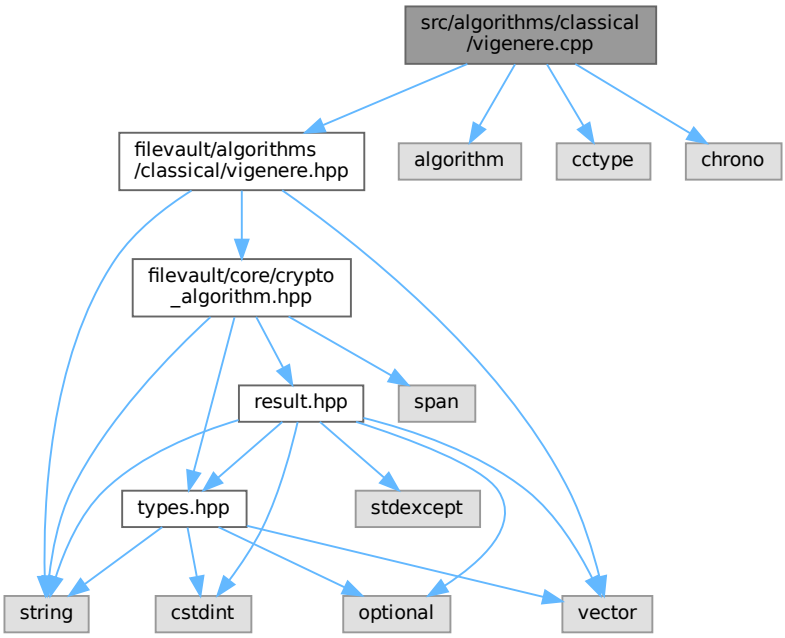
Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.127 src/algorithms/classical/vigenere.cpp File Reference

```
#include "filevault/algorithms/classical/vigenere.hpp"
#include <algorithm>
#include <cctype>
#include <chrono>
```

Include dependency graph for vigenere.cpp:



Namespaces

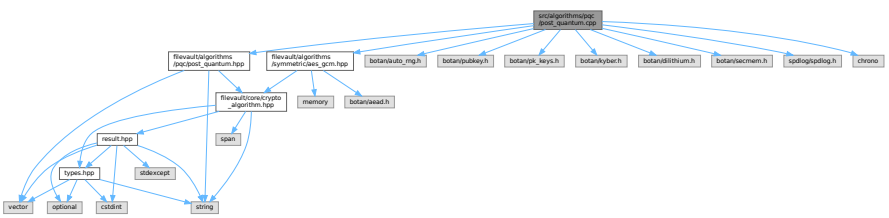
- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::classical

10.128 src/algorithms/pqc/post_quantum.cpp File Reference

Post-Quantum Cryptography implementations using Botan 3.x.

```
#include "filevault/algorithms/pqc/post_quantum.hpp"
#include "filevault/algorithms/symmetric/aes_gcm.hpp"
#include <botan/auto_rng.h>
#include <botan/pubkey.h>
#include <botan/pk_keys.h>
#include <botan/kyber.h>
#include <botan/dilithium.h>
#include <botan/secmem.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for post_quantum.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::pqc

Functions

- static Botan::KyberMode filevault::algorithms::pqc::get_kyber_mode (Kyber::Variant variant)
- static Botan::DilithiumMode filevault::algorithms::pqc::get_dilithium_mode (Dilithium::Variant variant)

10.128.1 Detailed Description

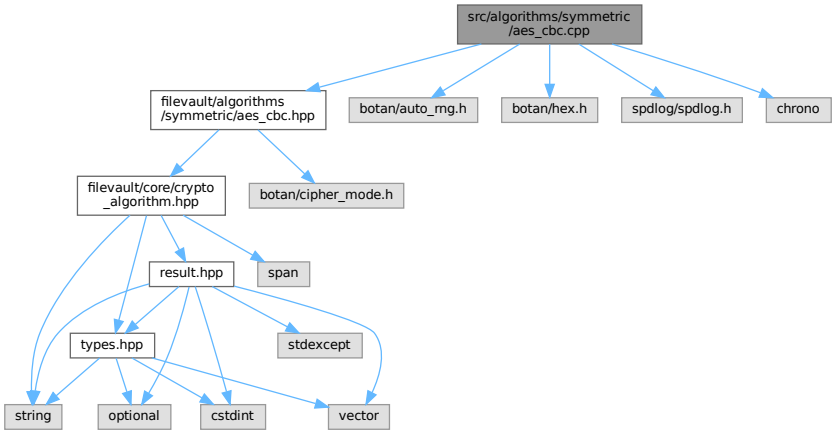
Post-Quantum Cryptography implementations using Botan 3.x.
Implements Kyber (ML-KEM) and Dilithium (ML-DSA) from NIST PQC standards.

10.129 src/algorithms/symmetric/aes_cbc.cpp File Reference

AES-CBC encryption implementation.

```
#include "filevault/algorithms/symmetric/aes_cbc.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for aes_cbc.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.129.1 Detailed Description

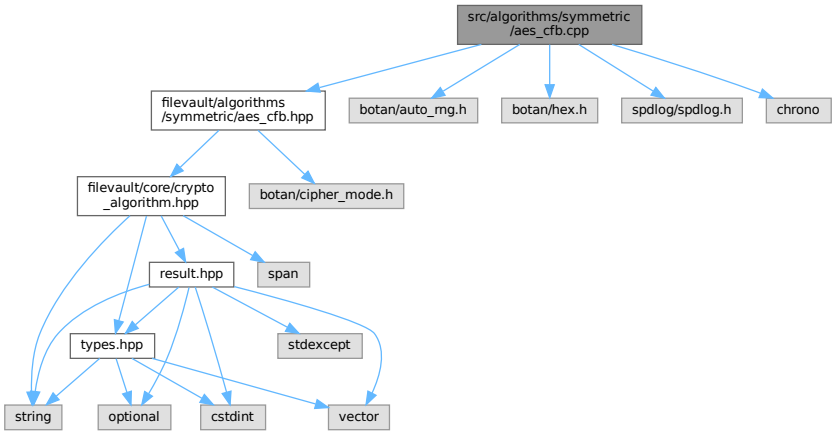
AES-CBC encryption implementation.

10.130 src/algorithms/symmetric/aes_cfb.cpp File Reference

AES-CFB encryption implementation.

```
#include "filevault/algorithms/symmetric/aes_cfb.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for aes_cfb.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.130.1 Detailed Description

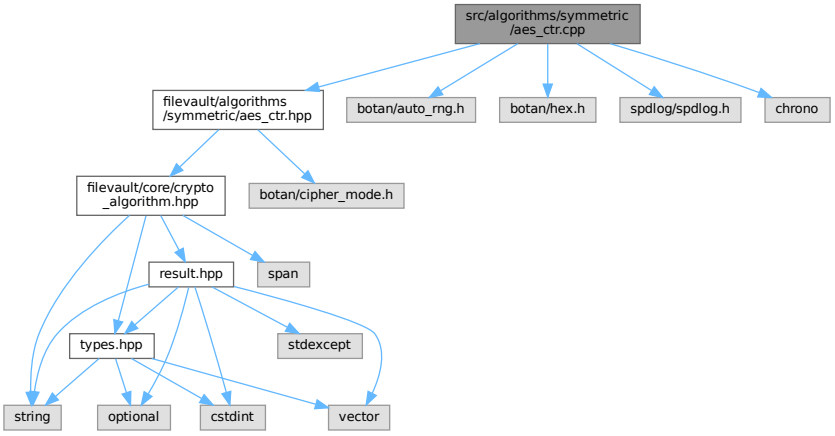
AES-CFB encryption implementation.

10.131 src/algorithms/symmetric/aes_ctr.cpp File Reference

AES-CTR encryption implementation.

```
#include "filevault/algorithms/symmetric/aes_ctr.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for aes_ctr.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.131.1 Detailed Description

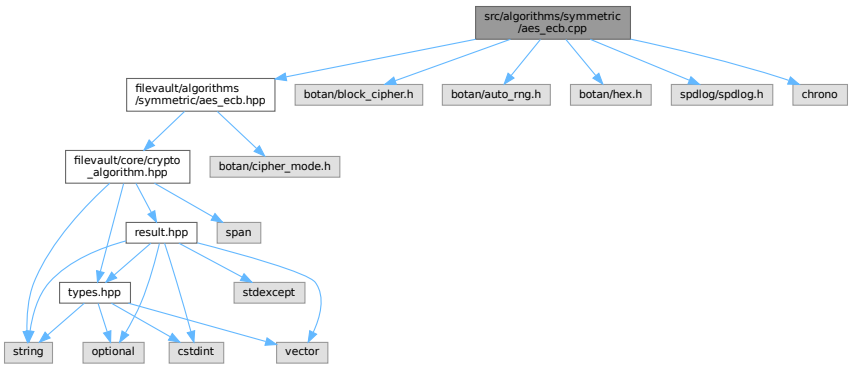
AES-CTR encryption implementation.

10.132 src/algorithms/symmetric/aes_ecb.cpp File Reference

AES-ECB encryption implementation.

```
#include "filevault/algorithms/symmetric/aes_ecb.hpp"
#include <botan/block_cipher.h>
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for aes_ecb.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

Functions

- static std::vector< uint8_t > filevault::algorithms::symmetric::add_pkcs7_padding (std::span< const uint8_t > data, size_t block_size)
- static std::vector< uint8_t > filevault::algorithms::symmetric::remove_pkcs7_padding (std::span< const uint8_t > data)

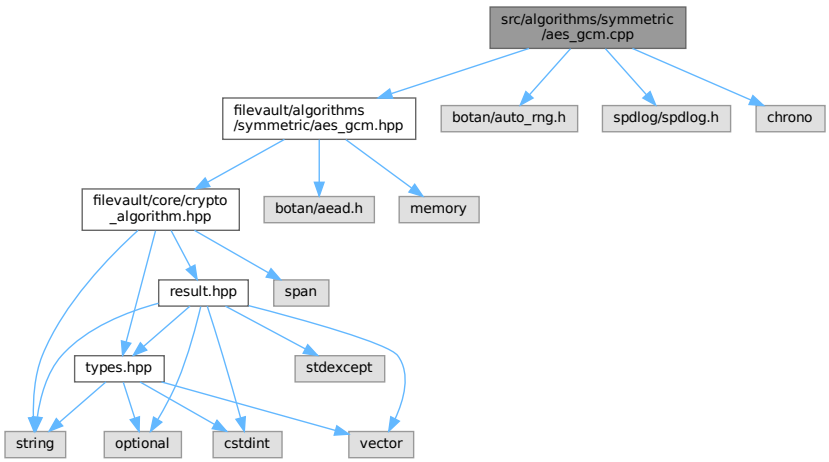
10.132.1 Detailed Description

AES-ECB encryption implementation.
WARNING: ECB mode is NOT secure for most uses!
Note: Botan 3.x doesn't expose ECB through Cipher_Mode, so we use BlockCipher directly with manual PKCS7 padding.

10.133 src/algorithms/symmetric/aes_gcm.cpp File Reference

```
#include "filevault/algorithms/symmetric/aes_gcm.hpp"
#include <botan/auto_rng.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for aes_gcm.cpp:



Namespaces

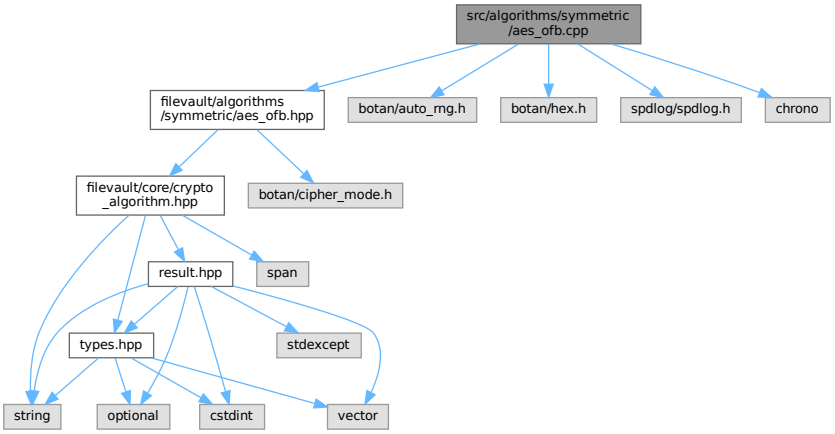
- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.134 src/algorithms/symmetric/aes_ofb.cpp File Reference

AES-OFB encryption implementation.

```
#include "filevault/algorithms/symmetric/aes_ofb.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for aes_ofb.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.134.1 Detailed Description

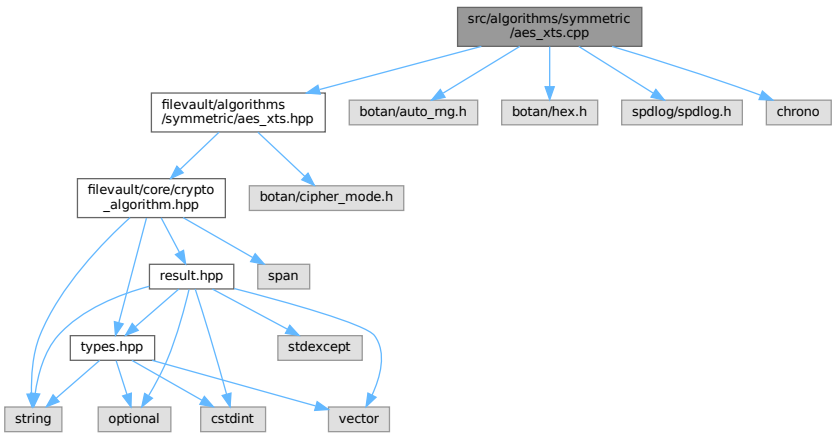
AES-OFB encryption implementation.

10.135 src/algorithms/symmetric/aes_xts.cpp File Reference

AES-XTS encryption implementation for disk encryption.

```
#include "filevault/algorithms/symmetric/aes_xts.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for aes_xts.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.135.1 Detailed Description

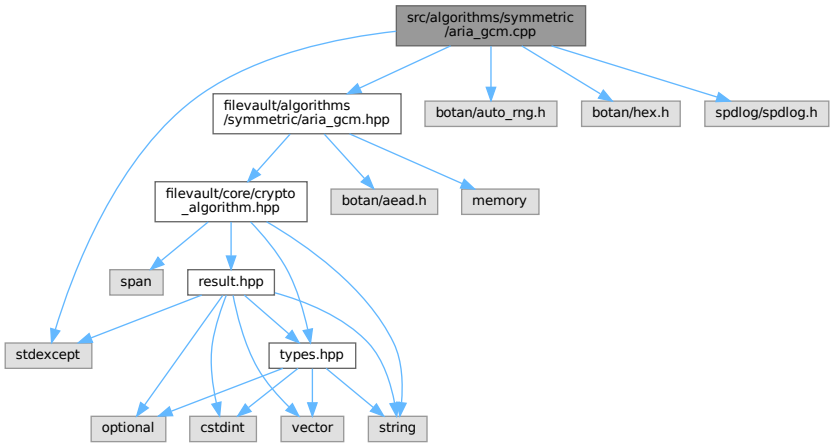
AES-XTS encryption implementation for disk encryption.

10.136 src/algorithms/symmetric/aria_gcm.cpp File Reference

Implementation of ARIA-GCM AEAD encryption.

```
#include "filevault/algorithms/symmetric/aria_gcm.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <stdexcept>
```

Include dependency graph for aria_gcm.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.136.1 Detailed Description

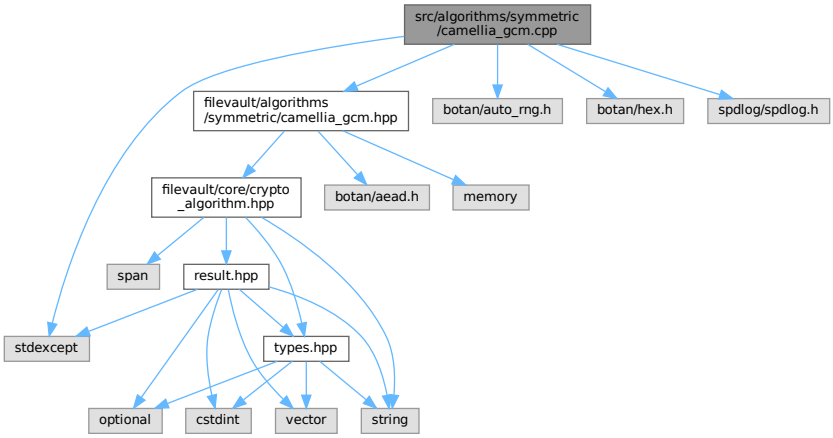
Implementation of ARIA-GCM AEAD encryption.

10.137 src/algorithms/symmetric/camellia_gcm.cpp File Reference

Implementation of Camellia-GCM AEAD encryption.

```
#include "filevault/algorithms/symmetric/camellia_gcm.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <stdexcept>
```

Include dependency graph for camellia_gcm.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

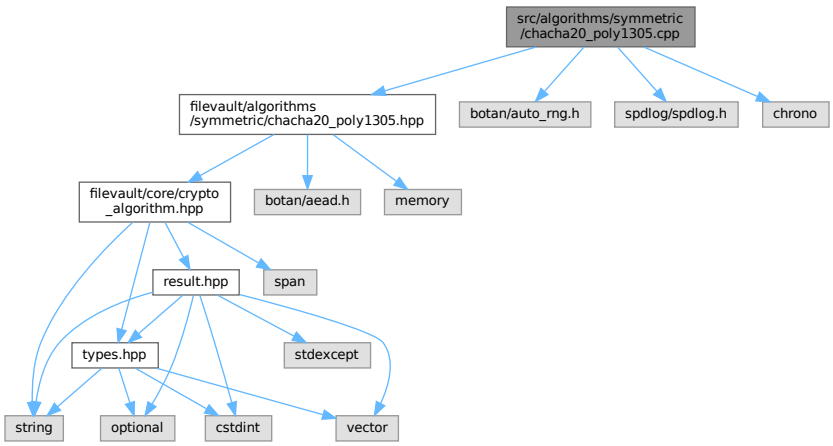
10.137.1 Detailed Description

Implementation of Camellia-GCM AEAD encryption.

10.138 src/algorithms/symmetric/chacha20_poly1305.cpp File Reference

```
#include "filevault/algorithms/symmetric/chacha20_poly1305.hpp"
#include <botan/auto_rng.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for chacha20_poly1305.cpp:



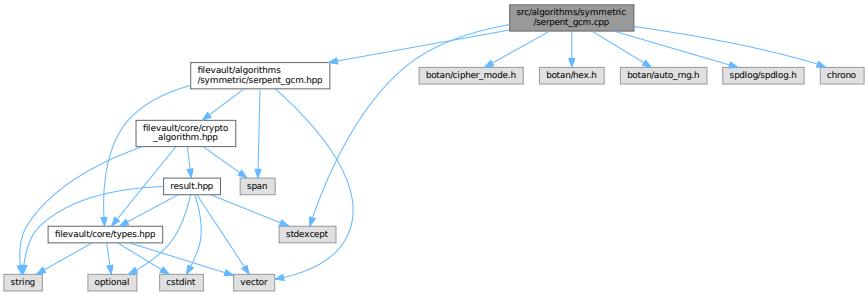
Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.139 src/algorithms/symmetric/serpent_gcm.cpp File Reference

```
#include "filevault/algorithms/symmetric/serpent_gcm.hpp"
#include <botan/cipher_mode.h>
#include <botan/hex.h>
#include <botan/auto_rng.h>
#include <spdlog/spdlog.h>
#include <stdexcept>
#include <chrono>
```

Include dependency graph for serpent_gcm.cpp:



Namespaces

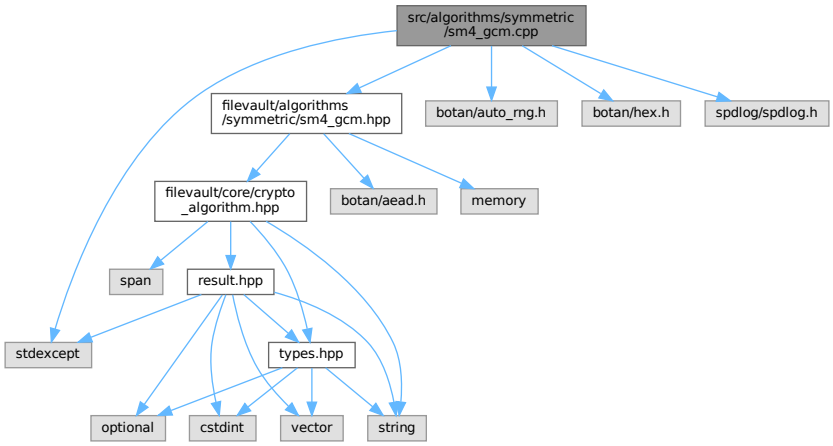
- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.140 src/algorithms/symmetric/sm4_gcm.cpp File Reference

Implementation of SM4-GCM AEAD encryption.

```
#include "filevault/algorithms/symmetric/sm4_gcm.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <stdexcept>
```

Include dependency graph for sm4_gcm.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.140.1 Detailed Description

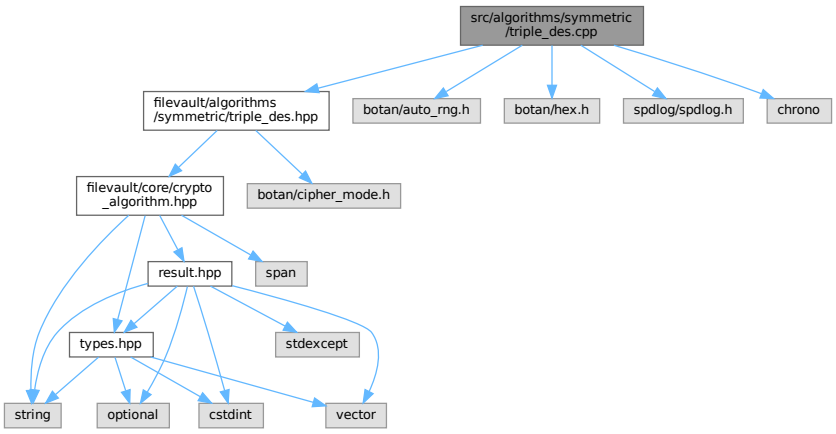
Implementation of SM4-GCM AEAD encryption.

10.141 src/algorithms/symmetric/triple_des.cpp File Reference

Triple-DES (3DES) encryption implementation.

```
#include "filevault/algorithms/symmetric/triple_des.hpp"
#include <botan/auto_rng.h>
#include <botan/hex.h>
#include <spdlog/spdlog.h>
#include <chrono>
```

Include dependency graph for triple_des.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.141.1 Detailed Description

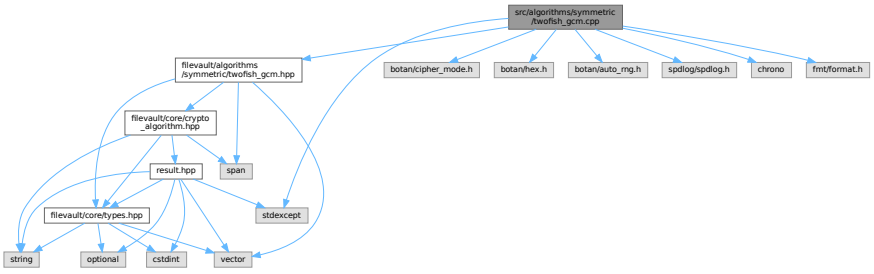
Triple-DES (3DES) encryption implementation.

10.142 src/algorithms/symmetric/twofish_gcm.cpp File Reference

Twofish-GCM encryption implementation.

```
#include "filevault/algorithms/symmetric/twofish_gcm.hpp"
#include <botan/cipher_mode.h>
#include <botan/hex.h>
#include <botan/auto_rng.h>
#include <spdlog/spdlog.h>
#include <stdexcept>
#include <chrono>
#include <fmt/format.h>
```

Include dependency graph for twofish_gcm.cpp:



Namespaces

- namespace filevault
- namespace filevault::algorithms
- namespace filevault::algorithms::symmetric

10.142.1 Detailed Description

Twofish-GCM encryption implementation.
Uses Botan's Twofish/GCM mode for authenticated encryption

Author

FileVault Team

Date

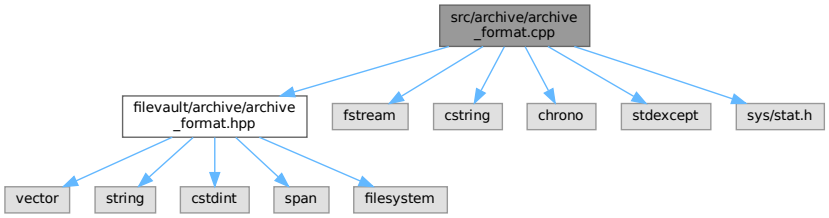
2024

10.143 src/archive/archive_format.cpp File Reference

```
#include "filevault/archive/archive_format.hpp"
#include <fstream>
```

```
#include <cstring>
#include <chrono>
#include <stdexcept>
#include <sys/stat.h>
```

Include dependency graph for archive_format.cpp:



Namespaces

- namespace filevault
- namespace filevault::archive

10.144 src/cli/app.cpp File Reference

```
#include "filevault/cli/app.hpp"
#include "filevault/cli/commands/encrypt_cmd.hpp"
#include "filevault/cli/commands/decrypt_cmd.hpp"
#include "filevault/cli/commands/hash_cmd.hpp"
#include "filevault/cli/commands/list_cmd.hpp"
#include "filevault/cli/commands/benchmark_cmd.hpp"
#include "filevault/cli/commands/config_cmd.hpp"
#include "filevault/cli/commands/info_cmd.hpp"
#include "filevault/cli/commands/compress_cmd.hpp"
#include "filevault/cli/commands/decompress_cmd.hpp"
#include "filevault/cli/commands/stego_cmd.hpp"
#include "filevault/cli/commands/archive_cmd.hpp"
#include "filevault/cli/commands/keygen_cmd.hpp"
#include "filevault/cli/commands/dump_cmd.hpp"
#include "filevault/cli/commands/sign_cmd.hpp"
#include "filevault/cli/commands/verify_cmd.hpp"
#include "filevault/cli/commands/keyinfo_cmd.hpp"
#include "filevault/utils/console.hpp"
#include <spdlog/spdlog.h>
#include <spdlog/sinks/stdout_color_sinks.h>
```

Include dependency graph for app.cpp:

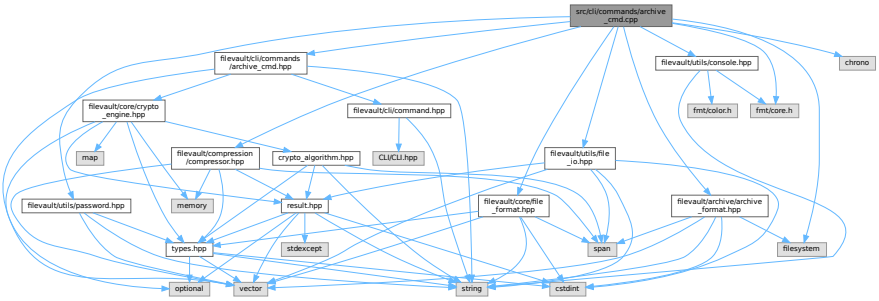
Namespaces

- namespace filevault
- namespace filevault::cli

10.145 src/cli/commands/archive_cmd.cpp File Reference

```
#include "filevault/cli/commands/archive_cmd.hpp"
#include "filevault/archive/archive_format.hpp"
#include "filevault/compression/compressor.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/password.hpp"
#include "filevault/core/file_format.hpp"
#include <fmt/core.h>
#include <filesystem>
#include <chrono>
```

Include dependency graph for archive_cmd.cpp:



Namespaces

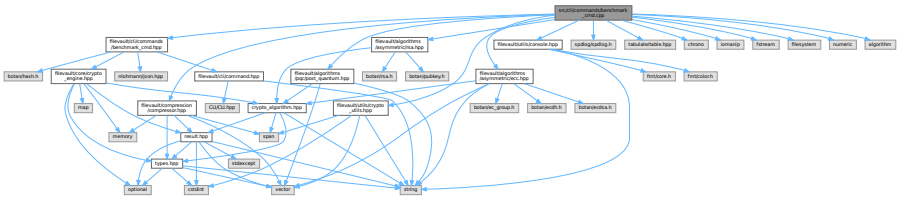
- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.146 src/cli/commands/benchmark_cmd.cpp File Reference

Enhanced benchmark command with all algorithms and tabulate tables.

```
#include "filevault/cli/commands/benchmark_cmd.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/crypto_utils.hpp"
#include "filevault/compression/compressor.hpp"
#include "filevault/algorithms/pqc/post_quantum.hpp"
#include "filevault/algorithms/asymmetric/rsa.hpp"
#include "filevault/algorithms/asymmetric/ecc.hpp"
#include <spdlog/spdlog.h>
#include <tabulate/table.hpp>
#include <chrono>
#include <iomanip>
#include <fstream>
#include <filesystem>
#include <numeric>
#include <algorithm>
```

Include dependency graph for benchmark_cmd.cpp:



Namespaces

- namespace filevault
- namespace filevault::cli

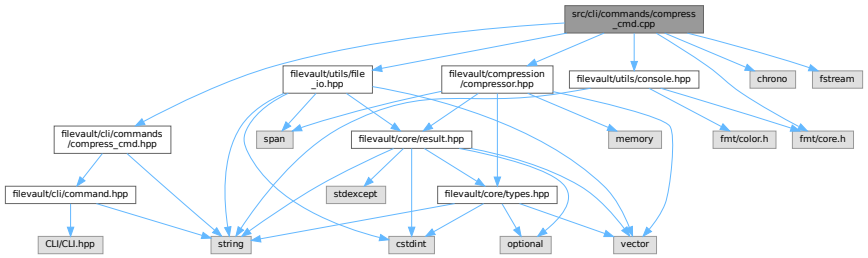
10.146.1 Detailed Description

Enhanced benchmark command with all algorithms and tabulate tables.

10.147 src/cli/commands/compress_cmd.cpp File Reference

```
#include "filevault/cli/commands/compress_cmd.hpp"
#include "filevault/compression/compressor.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/utils/console.hpp"
#include <fmt/core.h>
#include <chrono>
#include <fstream>
```

Include dependency graph for compress_cmd.cpp:



Namespaces

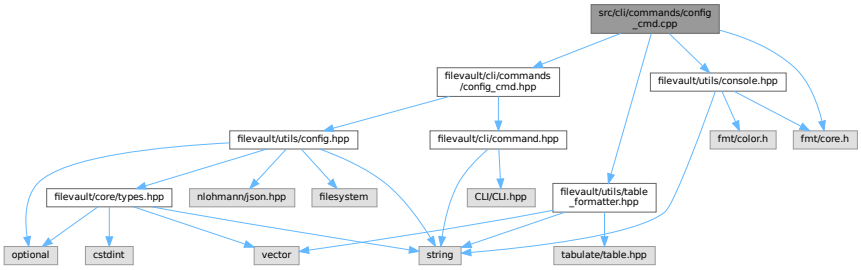
- namespace filevault
- namespace filevault::cli

10.148 src/cli/commands/config_cmd.cpp File Reference

```
#include "filevault/cli/commands/config_cmd.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/table_formatter.hpp"
```

```
#include <fmt/core.h>
```

Include dependency graph for config_cmd.cpp:



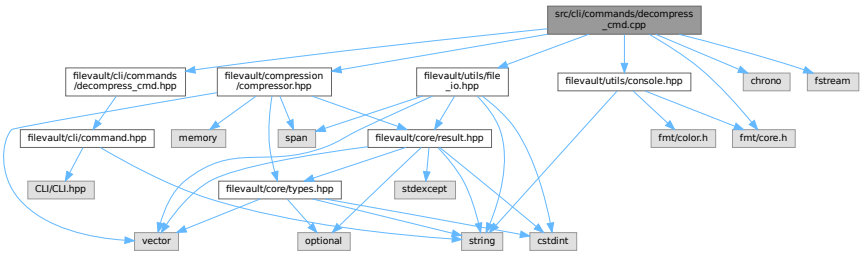
Namespaces

- namespace filevault
- namespace filevault::cli

10.149 src/cli/commands/decompress_cmd.cpp File Reference

```
#include "filevault/cli/commands/decompress_cmd.hpp"
#include "filevault/compression/compressor.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/utils/console.hpp"
#include <fmt/core.h>
#include <chrono>
#include <fstream>
```

Include dependency graph for decompress_cmd.cpp:



Namespaces

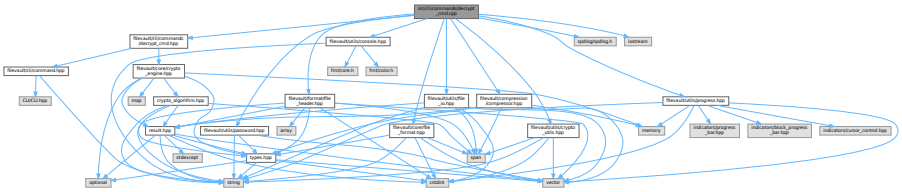
- namespace filevault
- namespace filevault::cli

10.150 src/cli/commands/decrypt_cmd.cpp File Reference

```
#include "filevault/cli/commands/decrypt_cmd.hpp"
#include "filevault/format/file_header.hpp"
#include "filevault/core/file_format.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/utils/crypto_utils.hpp"
```

```
#include "filevault/utils/password.hpp"
#include "filevault/utils/progress.hpp"
#include "filevault/compression/compressor.hpp"
#include <spdlog/spdlog.h>
#include <iostream>
```

Include dependency graph for decrypt_cmd.cpp:



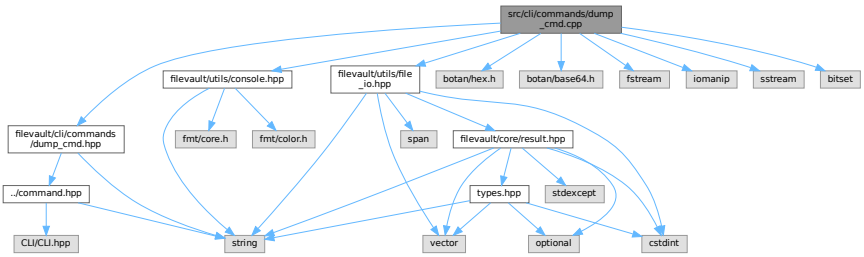
Namespaces

- namespace filevault
- namespace filevault::cli

10.151 src/cli/commands/dump_cmd.cpp File Reference

```
#include "filevault/cli/commands/dump_cmd.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include <botan/hex.h>
#include <botan/base64.h>
#include <fstream>
#include <iomanip>
#include <sstream>
#include <bitset>
```

Include dependency graph for dump_cmd.cpp:



Namespaces

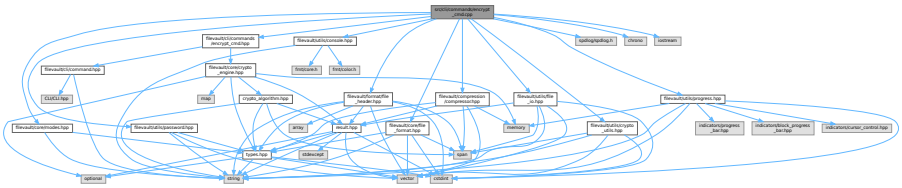
- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.152 src/cli/commands/encrypt_cmd.cpp File Reference

```
#include "filevault/cli/commands/encrypt_cmd.hpp"
#include "filevault/format/file_header.hpp"
#include "filevault/core/file_format.hpp"
```

```
#include "filevault/core/modes.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/utils/crypto_utils.hpp"
#include "filevault/utils/password.hpp"
#include "filevault/utils/progress.hpp"
#include "filevault/compression/compressor.hpp"
#include <spdlog/spdlog.h>
#include <chrono>
#include <iostream>
```

Include dependency graph for encrypt_cmd.cpp:



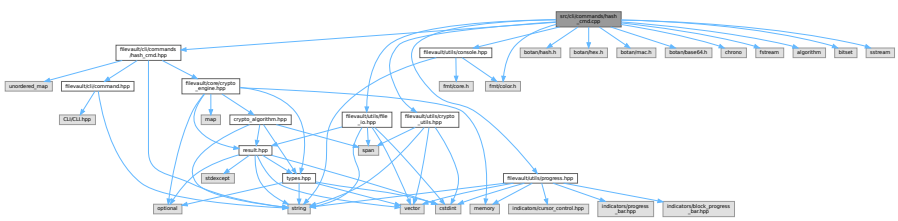
Namespaces

- namespace filevault
- namespace filevault::cli

10.153 src/cli/commands/hash_cmd.cpp File Reference

```
#include "filevault/cli/commands/hash_cmd.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/utils/crypto_utils.hpp"
#include "filevault/utils/progress.hpp"
#include <botan/hash.h>
#include <botan/hex.h>
#include <botan/mac.h>
#include <botan/base64.h>
#include <fmt/color.h>
#include <chrono>
#include <fstream>
#include <algorithm>
#include <bitset>
#include <sstream>
```

Include dependency graph for hash_cmd.cpp:



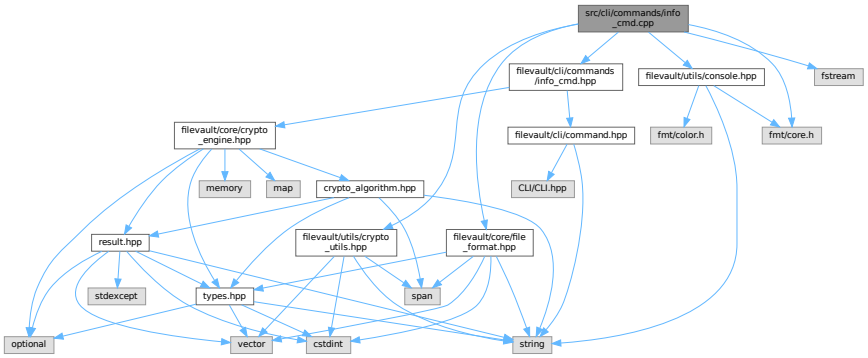
Namespaces

- namespace filevault
- namespace filevault::cli

10.154 src/cli/commands/info_cmd.cpp File Reference

```
#include "filevault/cli/commands/info_cmd.hpp"
#include "filevault/core/file_format.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/crypto_utils.hpp"
#include <fmt/core.h>
#include <fstream>
```

Include dependency graph for info_cmd.cpp:



Namespaces

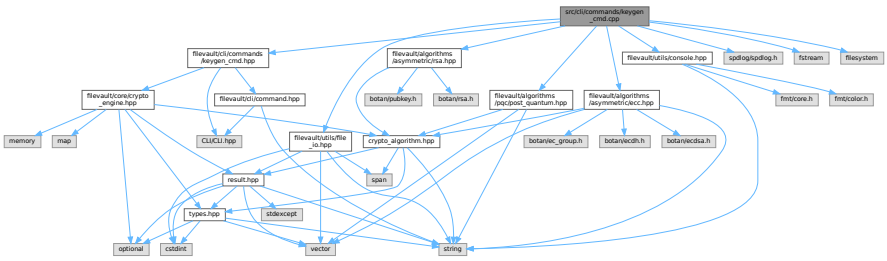
- namespace filevault
- namespace filevault::cli

10.155 src/cli/commands/keygen_cmd.cpp File Reference

Key generation command implementation.

```
#include "filevault/cli/commands/keygen_cmd.hpp"
#include "filevault/algorithms/asymmetric/rsa.hpp"
#include "filevault/algorithms/asymmetric/ecc.hpp"
#include "filevault/algorithms/pqc/post_quantum.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include <spdlog/spdlog.h>
#include <fstream>
#include <filesystem>
```

Include dependency graph for keygen_cmd.cpp:



Namespaces

- namespace filevault

- namespace filevault::cli

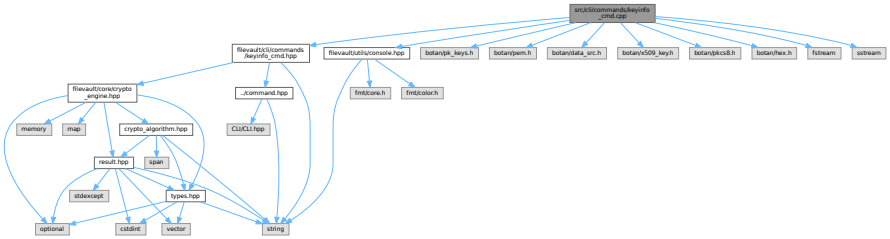
10.155.1 Detailed Description

Key generation command implementation.

10.156 src/cli/commands/keyinfo_cmd.cpp File Reference

```
#include "filevault/cli/commands/keyinfo_cmd.hpp"
#include "filevault/utils/console.hpp"
#include <botan/pk_keys.h>
#include <botan/pem.h>
#include <botan/data_src.h>
#include <botan/x509_key.h>
#include <botan/pkcs8.h>
#include <botan/hex.h>
#include <fstream>
#include <sstream>
```

Include dependency graph for keyinfo_cmd.cpp:



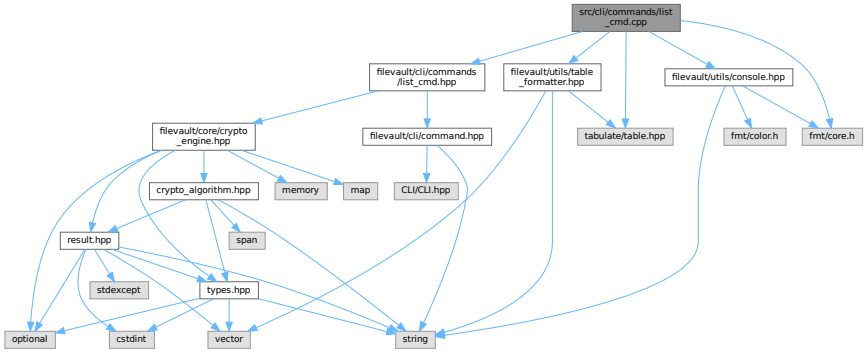
Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.157 src/cli/commands/list_cmd.cpp File Reference

```
#include "filevault/cli/commands/list_cmd.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/table_formatter.hpp"
#include <fmt/core.h>
#include <tabulate/table.hpp>
```

Include dependency graph for list_cmd.cpp:



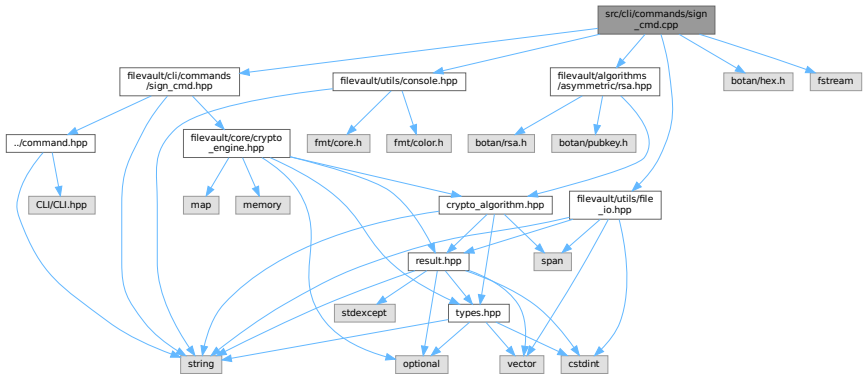
Namespaces

- namespace filevault
- namespace filevault::cli

10.158 src/cli/commands/sign_cmd.cpp File Reference

```
#include "filevault/cli/commands/sign_cmd.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/algorithms/asymmetric/rsa.hpp"
#include <botan/hex.h>
#include <fstream>
```

Include dependency graph for sign_cmd.cpp:



Namespaces

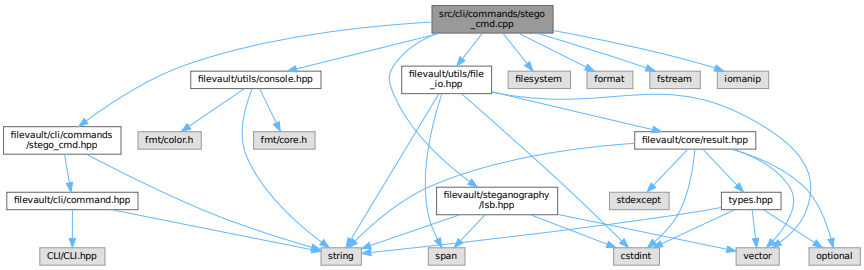
- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.159 src/cli/commands/stego_cmd.cpp File Reference

```
#include "filevault/cli/commands/stego_cmd.hpp"
#include "filevault/steganography/lsb.hpp"
```

```
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include <filesystem>
#include <format>
#include <fstream>
#include <iomanip>
```

Include dependency graph for stego_cmd.cpp:



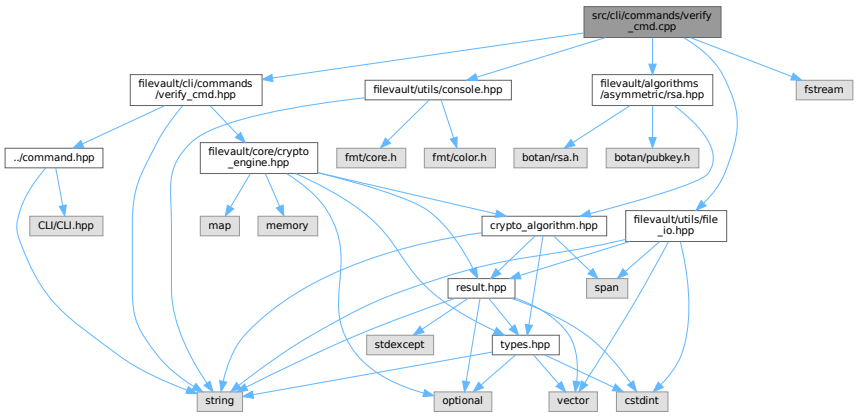
Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.160 src/cli/commands/verify_cmd.cpp File Reference

```
#include "filevault/cli/commands/verify_cmd.hpp"
#include "filevault/utils/console.hpp"
#include "filevault/utils/file_io.hpp"
#include "filevault/algorithms/asymmetric/rsa.hpp"
#include <fstream>
```

Include dependency graph for verify_cmd.cpp:



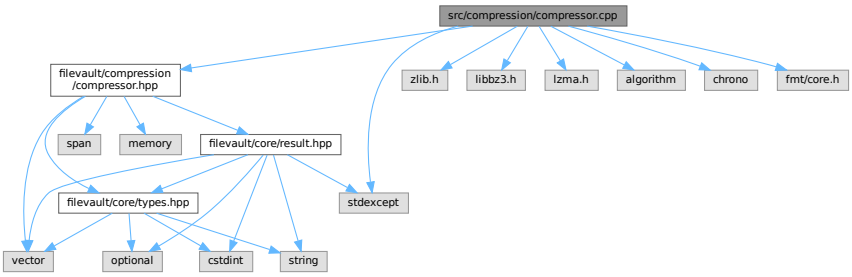
Namespaces

- namespace filevault
- namespace filevault::cli
- namespace filevault::cli::commands

10.161 src/compression/compressor.cpp File Reference

```
#include "filevault/compression/compressor.hpp"
#include <zlib.h>
#include <libbz3.h>
#include <lzma.h>
#include <algorithm>
#include <chrono>
#include <stdexcept>
#include <fmt/core.h>
```

Include dependency graph for compressor.cpp:



Namespaces

- namespace filevault
- namespace filevault::compression

10.162 src/core/crypto_engine.cpp File Reference

```
#include "filevault/core/crypto_engine.hpp"
#include "filevault/core/types.hpp"
#include "filevault/algorithms/symmetric/aes_gcm.hpp"
#include "filevault/algorithms/symmetric/aes_cbc.hpp"
#include "filevault/algorithms/symmetric/aes_ctr.hpp"
#include "filevault/algorithms/symmetric/aes_cfb.hpp"
#include "filevault/algorithms/symmetric/aes_ofb.hpp"
#include "filevault/algorithms/symmetric/aes_ecb.hpp"
#include "filevault/algorithms/symmetric/aes_xts.hpp"
#include "filevault/algorithms/symmetric/triple_des.hpp"
#include "filevault/algorithms/symmetric/chacha20_poly1305.hpp"
#include "filevault/algorithms/symmetric/serpent_gcm.hpp"
#include "filevault/algorithms/symmetric/twofish_gcm.hpp"
#include "filevault/algorithms/symmetric/camellia_gcm.hpp"
#include "filevault/algorithms/symmetric/aria_gcm.hpp"
#include "filevault/algorithms/symmetric/sm4_gcm.hpp"
#include "filevault/algorithms/asymmetric/rsa.hpp"
#include "filevault/algorithms/asymmetric/ecc.hpp"
#include "filevault/algorithms/pqc/post_quantum.hpp"
#include "filevault/algorithms/classical/caesar.hpp"
#include "filevault/algorithms/classical/vigenere.hpp"
#include "filevault/algorithms/classical/playfair.hpp"
#include "filevault/algorithms/classical/hill.hpp"
#include "filevault/algorithms/classical/substitution.hpp"
#include <botan/auto_rng.h>
#include <botan/argon2.h>
```

```
#include <botan/pwdhash.h>
#include <spdlog/spdlog.h>
#include <algorithm>
#include <cctype>
```

Include dependency graph for crypto_engine.cpp:

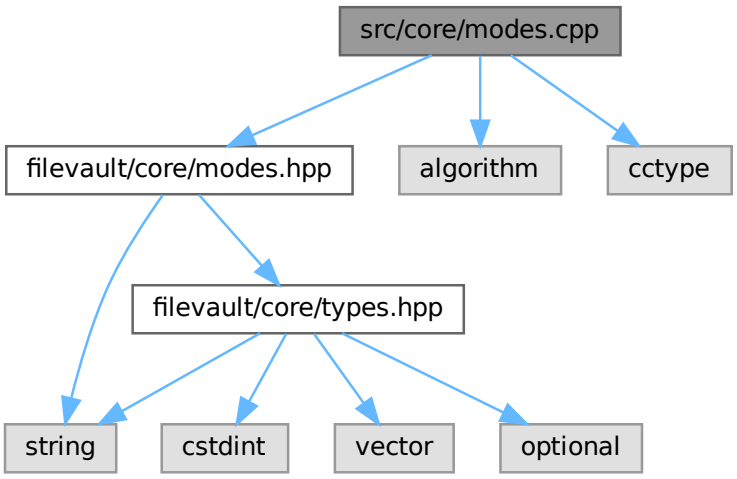
Namespaces

- namespace filevault
- namespace filevault::core

10.163 src/core/modes.cpp File Reference

```
#include "filevault/core/modes.hpp"
#include <algorithm>
#include <cctype>
```

Include dependency graph for modes.cpp:



Namespaces

- namespace filevault
- namespace filevault::core
- namespace filevault::core::presets

Variables

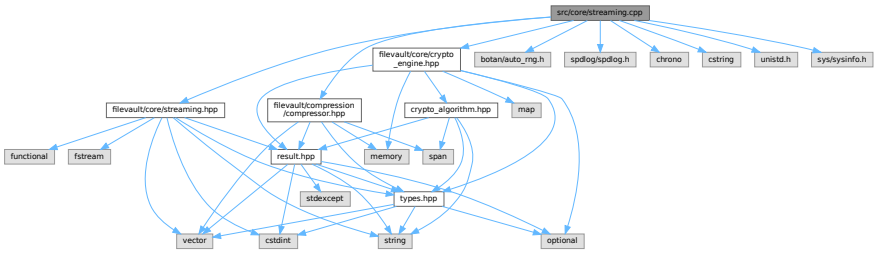
- const ModePreset filevault::core::presets::BASIC
- const ModePreset filevault::core::presets::STANDARD
- const ModePreset filevault::core::presets::ADVANCED

10.164 src/core/streaming.cpp File Reference

Streaming encryption implementation for large files.

```
#include "filevault/core/streaming.hpp"
#include "filevault/core/crypto_engine.hpp"
#include "filevault/compression/compressor.hpp"
#include <botan/auto_rng.h>
#include <spdlog/spdlog.h>
#include <chrono>
#include <cstring>
#include <unistd.h>
#include <sys/sysinfo.h>
```

Include dependency graph for streaming.cpp:



Namespaces

- namespace filevault
- namespace filevault::core

Variables

- static constexpr uint8_t filevault::core::STREAM_MAGIC [4] = {'F', 'V', 'S', 'T'}
- static constexpr uint8_t filevault::core::STREAM_VERSION = 1

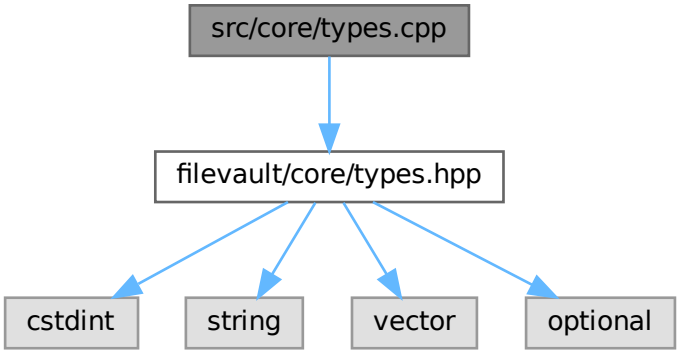
10.164.1 Detailed Description

Streaming encryption implementation for large files.

10.165 src/core/types.cpp File Reference

```
#include "filevault/core/types.hpp"
```

Include dependency graph for types.cpp:



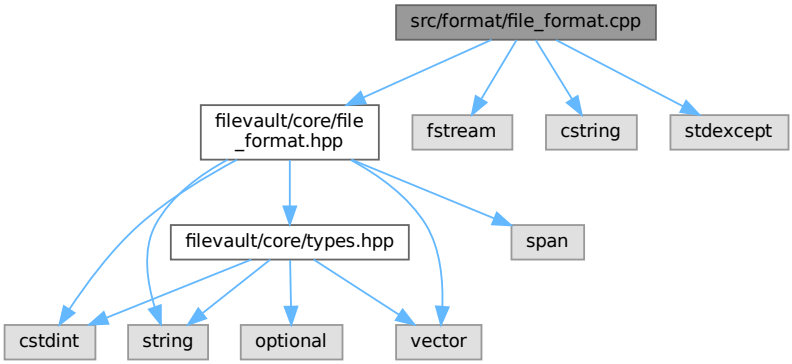
Namespaces

- namespace filevault
- namespace filevault::core

10.166 src/format/file_format.cpp File Reference

```
#include "filevault/core/file_format.hpp"
#include <fstream>
#include <cstring>
#include <stdexcept>
```

Include dependency graph for file_format.cpp:



Namespaces

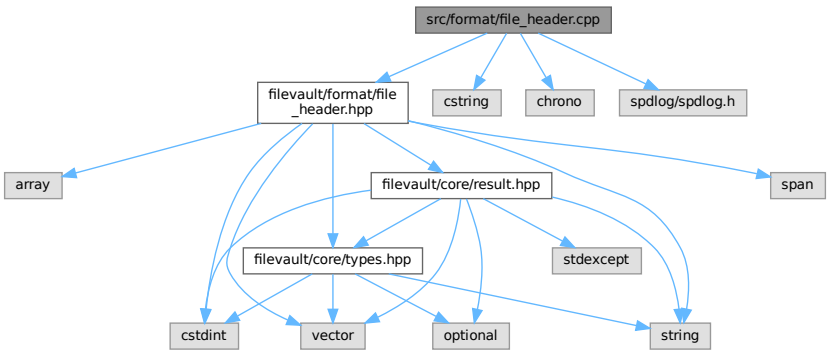
- namespace filevault

- namespace filevault::core

10.167 src/format/file_header.cpp File Reference

```
#include "filevault/format/file_header.hpp"
#include <cstring>
#include <chrono>
#include <spdlog/spdlog.h>
```

Include dependency graph for file_header.cpp:



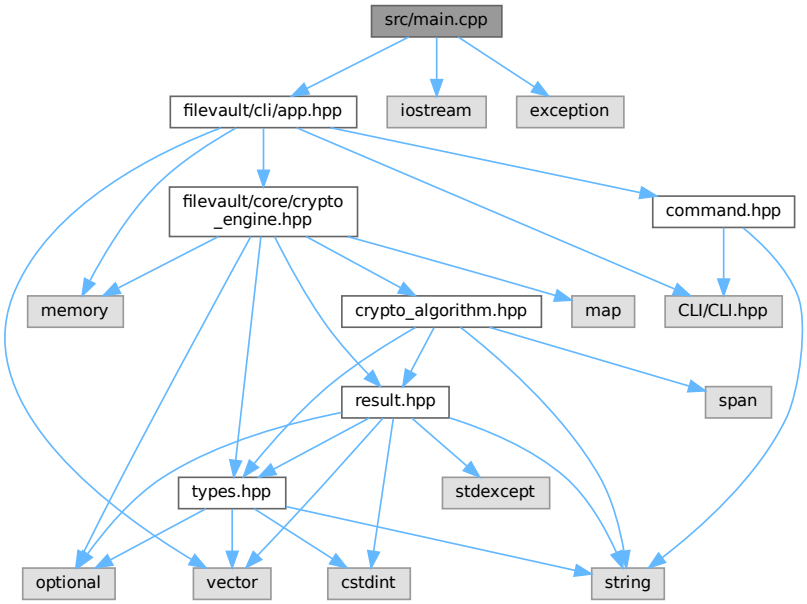
Namespaces

- namespace filevault
- namespace filevault::format

10.168 src/main.cpp File Reference

```
#include "filevault/cli/app.hpp"
#include <iostream>
#include <exception>
```

Include dependency graph for main.cpp:



Functions

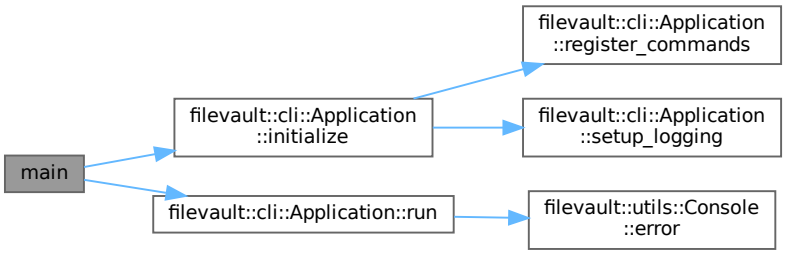
- int main (int argc, char **argv)

10.168.1 Function Documentation

10.168.1.1 main()

```
int main (
            int argc,
            char ** argv)
```

Here is the call graph for this function:

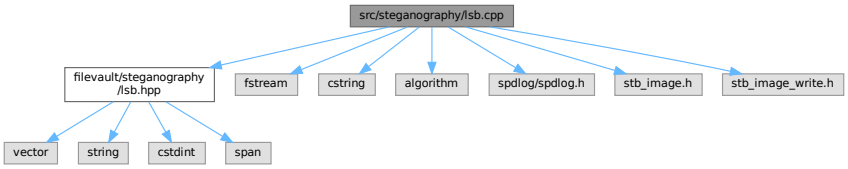


10.169 src/steganography/lsb.cpp File Reference

```
#include "filevault/steganography/lsb.hpp"
#include <fstream>
```

```
#include <cstring>
#include <algorithm>
#include <spdlog/spdlog.h>
#include <stb_image.h>
#include <stb_image_write.h>
```

Include dependency graph for lsb.cpp:



Namespaces

- namespace filevault
- namespace filevault::steganography

Macros

- #define STB_IMAGE_IMPLEMENTATION
- #define STB_IMAGE_WRITE_IMPLEMENTATION
- #define STBI_FAILURE_USERMSG

10.169.1 Macro Definition Documentation

10.169.1.1 STB_IMAGE_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

10.169.1.2 STB_IMAGE_WRITE_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```

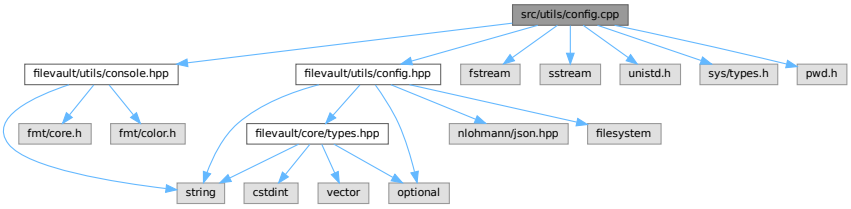
10.169.1.3 STBI_FAILURE_USERMSG

```
#define STBI_FAILURE_USERMSG
```

10.170 src/utils/config.cpp File Reference

```
#include "filevault/utils/config.hpp"
#include "filevault/utils/console.hpp"
#include <fstream>
#include <sstream>
#include <unistd.h>
#include <sys/types.h>
#include <pwd.h>
```

Include dependency graph for config.cpp:



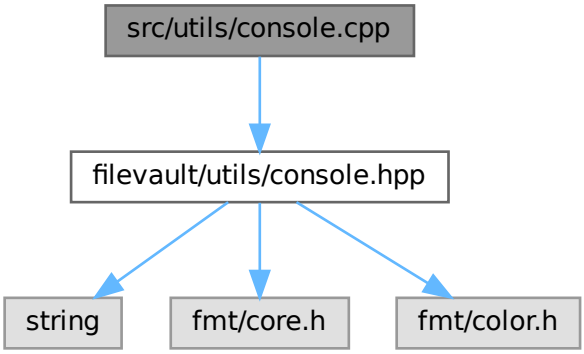
Namespaces

- namespace filevault
- namespace filevault::utils

10.171 src/utils/console.cpp File Reference

```
#include "filevault/utils/console.hpp"
```

Include dependency graph for console.cpp:



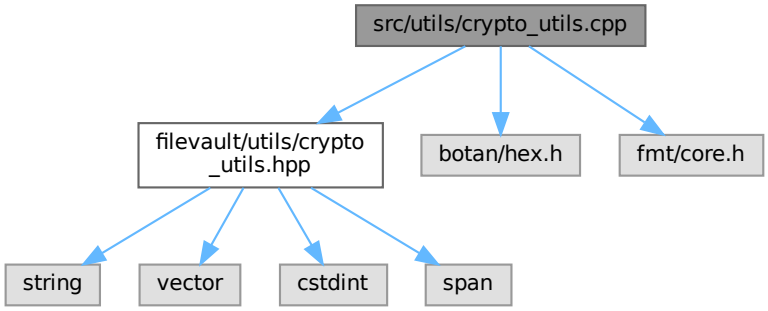
Namespaces

- namespace filevault
- namespace filevault::utils

10.172 src/utils/crypto_utils.cpp File Reference

```
#include "filevault/utils/crypto_utils.hpp"
#include <botan/hex.h>
#include <fmt/core.h>
```

Include dependency graph for crypto_utils.cpp:



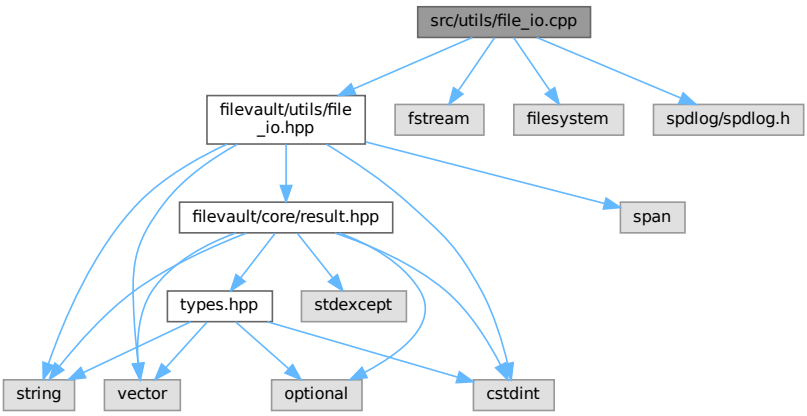
Namespaces

- namespace filevault
- namespace filevault::utils

10.173 src/utils/file_io.cpp File Reference

```
#include "filevault/utils/file_io.hpp"
#include <fstream>
#include <filesystem>
#include <spdlog/spdlog.h>
```

Include dependency graph for file_io.cpp:



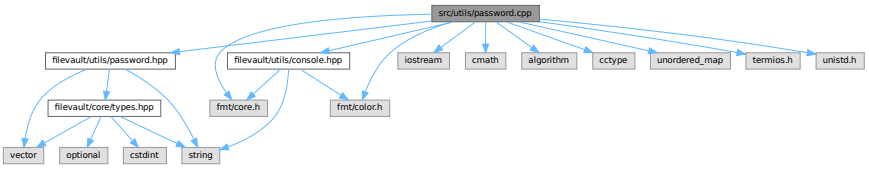
Namespaces

- namespace filevault
- namespace filevault::utils

10.174 src/utils/password.cpp File Reference

```
#include "filevault/utils/password.hpp"
#include "filevault/utils/console.hpp"
#include <fmt/core.h>
#include <fmt/color.h>
#include <iostream>
#include <cmath>
#include <algorithm>
#include <cctype>
#include <unordered_map>
#include <termios.h>
#include <unistd.h>
```

Include dependency graph for password.cpp:



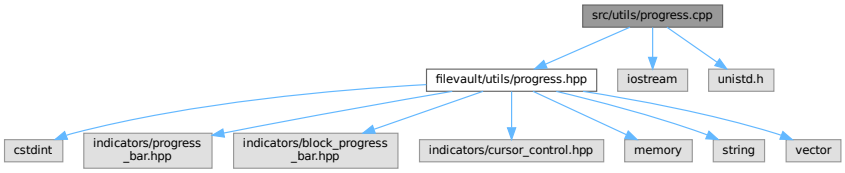
Namespaces

- namespace filevault
- namespace filevault::utils

10.175 src/utils/progress.cpp File Reference

```
#include "filevault/utils/progress.hpp"
#include <iostream>
#include <unistd.h>
```

Include dependency graph for progress.cpp:



Namespaces

- namespace filevault
- namespace filevault::utils

Macros

- #define ISATTY isatty
- #define FILENO fileno

Functions

- static bool filevault::utils::is_terminal ()

10.175.1 Macro Definition Documentation

10.175.1.1 FILENO

```
#define FILENO fileno
```

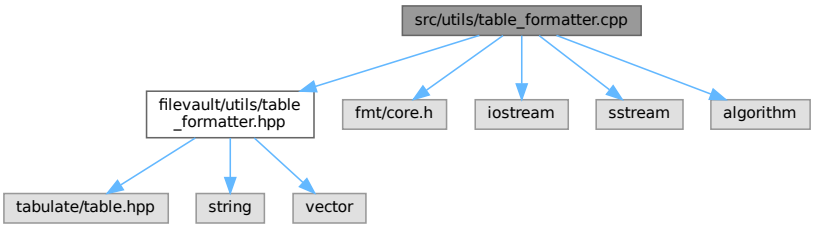
10.175.1.2 ISATTY

```
#define ISATTY isatty
```

10.176 src/utils/table_formatter.cpp File Reference

```
#include "filevault/utils/table_formatter.hpp"
#include <fmt/core.h>
#include <iostream>
#include <sstream>
#include <algorithm>
```

Include dependency graph for table_formatter.cpp:



Namespaces

- namespace filevault
- namespace filevault::utils

Variables

- static std::vector< std::vector< std::string > > filevault::utils::mingw_data

Index

~AES_CBC
filevault::algorithms::symmetric::AES_CBC, 66
~AES_CFB
filevault::algorithms::symmetric::AES_CFB, 72
~AES_CTR
filevault::algorithms::symmetric::AES_CTR, 77
~AES_ECB
filevault::algorithms::symmetric::AES_ECB, 82
~AES_GCM
filevault::algorithms::symmetric::AES_GCM, 88
~AES_OFB
filevault::algorithms::symmetric::AES_OFB, 93
~AES_XTS
filevault::algorithms::symmetric::AES_XTS, 98
~ARIA_GCM
filevault::algorithms::symmetric::ARIA_GCM, 119
~Application
filevault::cli::Application, 103
~Camellia_GCM
filevault::algorithms::symmetric::Camellia_GCM, 146
~ChaCha20Poly1305
filevault::algorithms::symmetric::ChaCha20Poly1305, 151
~CompressCommand
filevault::cli::CompressCommand, 157
~CryptoEngine
filevault::core::CryptoEngine, 183
~ECCHybrid
filevault::algorithms::asymmetric::ECCHybrid, 208
~ECDH
filevault::algorithms::asymmetric::ECDH, 212
~ECDSA
filevault::algorithms::asymmetric::ECDSA, 216
~ICommand
filevault::cli::ICommand, 266
~ICompressor
filevault::compression::ICompressor, 267
~ICryptoAlgorithm
filevault::core::ICryptoAlgorithm, 270
~Kyber
filevault::algorithms::pqc::Kyber, 286
~KyberHybrid
filevault::algorithms::pqc::KyberHybrid, 291
~ProgressBar
filevault::utils::ProgressBar, 322
~RSA
filevault::algorithms::asymmetric::RSA, 328
~SM4_GCM
filevault::algorithms::symmetric::SM4_GCM, 345
~TripleDES
filevault::algorithms::symmetric::TripleDES, 372

add_pkcs7_padding
filevault::algorithms::symmetric, 51
add_row
filevault::utils::TableFormatter, 368
ADVANCED
filevault::core, 60
filevault::core::presets, 61
AES_128_CBC
filevault::core, 56, 57
AES_128_CFB
filevault::core, 56, 57
AES_128_CTR
filevault::core, 56, 57
AES_128_ECB
filevault::core, 56, 57
AES_128_GCM
filevault::core, 56, 57
AES_128_OFB
filevault::core, 56, 57
AES_128_XTS
filevault::core, 56, 57
AES_192_CBC
filevault::core, 56, 57
AES_192_CFB
filevault::core, 56, 57
AES_192_CTR
filevault::core, 56, 57
AES_192_ECB
filevault::core, 56, 57
AES_192_GCM
filevault::core, 56, 57
AES_192_OFB
filevault::core, 56, 57
AES_256_CBC
filevault::core, 56, 57
AES_256_CFB
filevault::core, 56, 57
AES_256_CTR
filevault::core, 56, 57
AES_256_ECB
filevault::core, 56, 57
AES_256_GCM
filevault::core, 56, 57
AES_256_OFB
filevault::core, 56, 57
AES_256_XTS

filevault::core, 56, 58
AES_CBC
filevault::algorithms::symmetric::AES_CBC, 66
AES_CFB
filevault::algorithms::symmetric::AES_CFB, 72
AES_CTR
filevault::algorithms::symmetric::AES_CTR, 77
AES_ECB
filevault::algorithms::symmetric::AES_ECB, 82
AES_GCM
filevault::algorithms::symmetric::AES_GCM, 88
AES_OFB
filevault::algorithms::symmetric::AES_OFB, 93
AES_XTS
filevault::algorithms::symmetric::AES_XTS, 98
algorithm
filevault::algorithms::pqc::PQKeyPair, 321
filevault::cli::AsymmetricBenchmarkResult, 123
filevault::cli::BenchmarkResult, 135
filevault::cli::InfoCommand::FileInfo, 246
filevault::cli::PQCBenchmarkResult, 320
filevault::cli::SignatureBenchmarkResult, 339
filevault::core::EncryptionConfig, 225
filevault::core::FileHeader, 234
filevault::core::HashConfig, 256
filevault::core::ModePreset, 304
filevault::core::StreamingConfig, 355
filevault::format::FileHeader, 237
algorithm_
filevault::cli::BenchmarkCommand, 134
filevault::cli::commands::ArchiveCommand, 111
filevault::cli::commands::SignCommand, 343
filevault::cli::commands::VerifyCommand, 385
filevault::cli::CompressCommand, 161
filevault::cli::DecompressCommand, 194
filevault::cli::EncryptCommand, 223
filevault::cli::HashCommand, 255
filevault::cli::KeygenCommand, 280
filevault::format::FileHeader, 245
algorithm_list_table
filevault::utils::TableFormatter, 368
algorithm_name
filevault::core::CryptoEngine, 183
algorithm_used
filevault::core::CryptoResult, 188
AlgorithmID
filevault::core, 55
algorithms_
filevault::core::CryptoEngine, 187
AlgorithmType
filevault::core, 57
all_
filevault::cli::BenchmarkCommand, 134
analyze_strength
filevault::utils::Password, 305
app_
filevault::cli::Application, 105
Application
filevault::cli::Application, 103
apply_security_level
filevault::core::EncryptionConfig, 224
apply_substitution
filevault::algorithms::classical::SubstitutionCipher, 362
apply_to
filevault::core::ModePreset, 302
apply_user_mode
filevault::core::EncryptionConfig, 225
ArchiveCommand
filevault::cli::commands::ArchiveCommand, 107
ARGON2I
filevault::core, 59, 60
ARGON2ID
filevault::core, 59, 60
ARIA_128_GCM
filevault::core, 56, 57
ARIA_192_GCM
filevault::core, 56, 57
ARIA_256_GCM
filevault::core, 56, 57
ARIA_GCM
filevault::algorithms::symmetric::ARIA_GCM, 119
associated_data
filevault::core::EncryptionConfig, 225
asymmetric_only_
filevault::cli::BenchmarkCommand, 134
auto_detect_
filevault::cli::CompressCommand, 161
filevault::cli::DecompressCommand, 194

bar_
filevault::utils::BlockProgressBar, 136
filevault::utils::ProgressBar, 323
BASIC
filevault::core::presets, 61
benchmark_
filevault::cli::CompressCommand, 161
filevault::cli::DecompressCommand, 194
filevault::cli::HashCommand, 255
benchmark_algorithm
filevault::cli::BenchmarkCommand, 125
benchmark_asymmetric
filevault::cli::BenchmarkCommand, 125
benchmark_asymmetric_algorithm
filevault::cli::BenchmarkCommand, 126
benchmark_compression
filevault::cli::BenchmarkCommand, 127
benchmark_hash
filevault::cli::BenchmarkCommand, 128
benchmark_hybrid_algorithm
filevault::cli::BenchmarkCommand, 128
benchmark_kdf
filevault::cli::BenchmarkCommand, 129
benchmark_pqc
filevault::cli::BenchmarkCommand, 129
benchmark_pqc_algorithm
filevault::cli::BenchmarkCommand, 130

benchmark_signature_algorithm
filevault::cli::BenchmarkCommand, 130
benchmark_symmetric
filevault::cli::BenchmarkCommand, 131
benchmark_table
filevault::utils::TableFormatter, 369
BenchmarkCommand
filevault::cli::BenchmarkCommand, 125
bits
filevault::algorithms::asymmetric::RSAKeyPair, 332
bits_per_channel_
filevault::cli::commands::StegoCommand, 354
BLAKE2B_256
filevault::core, 59
BLAKE2B_384
filevault::core, 59
BLAKE2B_512
filevault::core, 59
BLAKE2S_256
filevault::core, 59
block_size
filevault::algorithms::symmetric::AES_CBC, 67
filevault::algorithms::symmetric::AES_CFB, 72
filevault::algorithms::symmetric::AES_ECB, 82
filevault::algorithms::symmetric::AES_OFB, 94
filevault::algorithms::symmetric::AES_XTS, 98
filevault::algorithms::symmetric::TripleDES, 372
BlockProgressBar
filevault::utils::BlockProgressBar, 136
botan_curve_name_
filevault::algorithms::asymmetric::ECCHybrid, 211
filevault::algorithms::asymmetric::ECDH, 214
filevault::algorithms::asymmetric::ECDSA, 218
botan_name_
filevault::algorithms::pqc::Dilithium, 203
filevault::algorithms::pqc::Kyber, 289
filevault::algorithms::symmetric::AES_CBC, 70
filevault::algorithms::symmetric::AES_CFB, 75
filevault::algorithms::symmetric::AES_CTR, 80
filevault::algorithms::symmetric::AES_ECB, 86
filevault::algorithms::symmetric::AES_GCM, 91
filevault::algorithms::symmetric::AES_OFB, 96
filevault::algorithms::symmetric::AES_XTS, 103
filevault::algorithms::symmetric::ARIA_GCM, 123
filevault::algorithms::symmetric::Camellia_GCM, 150
filevault::algorithms::symmetric::Twofish_GCM, 382
brute_force
filevault::algorithms::classical::Caesar, 141
build_matrix
filevault::algorithms::classical::Playfair, 315, 316
bytes_processed
filevault::core::ChunkInfo, 155
filevault::core::StreamingResult, 360
BZIP2
filevault::core, 58, 59

CAESAR
filevault::core, 56, 58
Caesar
filevault::algorithms::classical::Caesar, 140
calculate_capacity
filevault::steganography::LSBSteganography, 297
calculate_entropy
filevault::utils::Password, 306
calculate_file_hash
filevault::cli::HashCommand, 251
calculate_file_hmac
filevault::cli::HashCommand, 251
CAMELLIA_128_GCM
filevault::core, 56, 57
CAMELLIA_192_GCM
filevault::core, 56, 57
CAMELLIA_256_GCM
filevault::core, 56, 57
Camellia_GCM
filevault::algorithms::symmetric::Camellia_GCM, 146
CHACHA20_POLY1305
filevault::core, 56, 57
ChaCha20Poly1305
filevault::algorithms::symmetric::ChaCha20Poly1305, 151
check_pair_
filevault::cli::commands::KeyInfoCommand, 283
chunk_index
filevault::core::ChunkInfo, 155
chunk_size
filevault::core::ChunkInfo, 155
filevault::core::StreamingConfig, 355
chunks_processed
filevault::core::StreamingResult, 360
ciphertext_size
filevault::algorithms::pqc::Kyber, 286
commands_
filevault::cli::Application, 105
comment
filevault::core::EncryptionConfig, 225
common_passwords_
filevault::utils::Password, 310
compress
filevault::compression::Bzip2Compressor, 138
filevault::compression::ICompressor, 267
filevault::compression::LzmaCompressor, 301
filevault::compression::ZlibCompressor, 392
CompressCommand
filevault::cli::CompressCommand, 157
compressed
filevault::cli::InfoCommand::FileInfo, 246
filevault::core::FileHeader, 234
compressed_size
filevault::compression::CompressionResult, 162
compression
filevault::cli::InfoCommand::FileInfo, 246
filevault::core::EncryptionConfig, 225
filevault::core::FileHeader, 235

filevault::core::ModePreset, 304
filevault::core::StreamingConfig, 355
compression_
filevault::cli::commands::ArchiveCommand, 111
compression_level
filevault::core::EncryptionConfig, 225
filevault::core::ModePreset, 304
filevault::core::StreamingConfig, 355
compression_level_
filevault::cli::EncryptCommand, 223
filevault::utils::Config, 170
compression_only_
filevault::cli::BenchmarkCommand, 134
compression_ratio
filevault::compression::CompressionResult, 162
compression_type_
filevault::cli::EncryptCommand, 223
CompressionID
filevault::core, 58
CompressionType
filevault::core, 58
ConfigCommand
filevault::cli::ConfigCommand, 172
cover_image_
filevault::cli::commands::StegoCommand, 354
crack_time_offline
filevault::core::PasswordAnalysis, 311
crack_time_online
filevault::core::PasswordAnalysis, 311
create
filevault::compression::CompressionService, 163
create_archive
filevault::archive::ArchiveFormat, 113
create_header
filevault::core::FileFormatHandler, 228
create_reverse_map
filevault::algorithms::classical::SubstitutionCipher, 362
CryptoEngine
filevault::core::CryptoEngine, 183
current_progress_
filevault::utils::ProgressBar, 323
curve
filevault::algorithms::asymmetric::ECCKeyPair, 211
curve_
filevault::algorithms::asymmetric::ECCHybrid, 211
filevault::algorithms::asymmetric::ECDH, 214
filevault::algorithms::asymmetric::ECDSA, 218
curve_name
filevault::algorithms::asymmetric::ECCKeyPair, 211
filevault::algorithms::asymmetric::ECDH, 213
filevault::algorithms::asymmetric::ECDSA, 216

data
filevault::compression::CompressionResult, 162
filevault::core::CryptoResult, 188
data_size
filevault::cli::InfoCommand::FileInfo, 246
data_size_
filevault::cli::BenchmarkCommand, 134
debug
filevault::utils::Console, 176
decaps_ms
filevault::cli::PQCBenchmarkResult, 320
decompress
filevault::compression::Bzip2Compressor, 138
filevault::compression::ICompressor, 268
filevault::compression::LzmaCompressor, 301
filevault::compression::ZlibCompressor, 392
decompress_
filevault::cli::CompressCommand, 161
decrypt
filevault::algorithms::asymmetric::ECCHybrid, 209
filevault::algorithms::asymmetric::RSA, 329
filevault::algorithms::classical::Caesar, 141, 142
filevault::algorithms::classical::HillCipher, 258
filevault::algorithms::classical::Playfair, 316, 317
filevault::algorithms::classical::SubstitutionCipher, 363
filevault::algorithms::classical::Vigenere, 388
filevault::algorithms::pqc::Kyber, 286
filevault::algorithms::pqc::KyberHybrid, 291
filevault::algorithms::symmetric::AES_CBC, 67
filevault::algorithms::symmetric::AES_CFB, 72
filevault::algorithms::symmetric::AES_CTR, 77
filevault::algorithms::symmetric::AES_ECB, 83
filevault::algorithms::symmetric::AES_GCM, 88
filevault::algorithms::symmetric::AES_OFB, 94
filevault::algorithms::symmetric::AES_XTS, 99
filevault::algorithms::symmetric::ARIA_GCM, 119
filevault::algorithms::symmetric::Camellia_GCM, 146
filevault::algorithms::symmetric::ChaCha20Poly1305, 152
filevault::algorithms::symmetric::Serpent_GCM, 336
filevault::algorithms::symmetric::SM4_GCM, 345
filevault::algorithms::symmetric::TripleDES, 372
filevault::algorithms::symmetric::Twofish_GCM, 378
filevault::core::ICryptoAlgorithm, 270
decrypt_block
filevault::algorithms::classical::HillCipher, 259
decrypt_file
filevault::core::StreamingCrypto, 356
decrypt_mbps
filevault::cli::BenchmarkResult, 135
decrypt_ms
filevault::cli::AsymmetricBenchmarkResult, 123
filevault::cli::BenchmarkResult, 135
DecryptCommand
filevault::cli::DecryptCommand, 196
default_algorithm_
filevault::utils::Config, 170
default_compression_
filevault::utils::Config, 170
default_kdf_

filevault::utils::Config, 170
default_mode_
filevault::utils::Config, 170
derive_chunk_nonce
filevault::core::StreamingCrypto, 357
derive_key
filevault::core::CryptoEngine, 183
derive_shared_secret
filevault::algorithms::asymmetric::ECDH, 213
description
filevault::cli::BenchmarkCommand, 131
filevault::cli::commands::ArchiveCommand, 107
filevault::cli::commands::DumpCommand, 205
filevault::cli::commands::KeyInfoCommand, 282
filevault::cli::commands::SignCommand, 341
filevault::cli::commands::StegoCommand, 350
filevault::cli::commands::VerifyCommand, 384
filevault::cli::CompressCommand, 157
filevault::cli::ConfigCommand, 172
filevault::cli::DecompressCommand, 191
filevault::cli::DecryptCommand, 196
filevault::cli::EncryptCommand, 220
filevault::cli::HashCommand, 252
filevault::cli::ICommand, 266
filevault::cli::InfoCommand, 274
filevault::cli::KeygenCommand, 278
filevault::cli::ListCommand, 295
filevault::core::ModePreset, 303
description_
filevault::cli::commands::DumpCommand, 206
filevault::cli::commands::KeyInfoCommand, 283
filevault::cli::commands::SignCommand, 343
filevault::cli::commands::VerifyCommand, 385
deserialize
filevault::archive::FileEntry, 227
filevault::core::Argon2Params, 116
filevault::core::FileHeader, 233
filevault::core::PBKDF2Params, 312
filevault::core::ScryptParams, 333
filevault::format::FileHeader, 237
detect_algorithm
filevault::cli::CompressCommand, 157
filevault::cli::DecompressCommand, 191
determinant
filevault::algorithms::classical::HillCipher, 260
Dilithium
filevault::algorithms::pqc::Dilithium, 200
Dilithium2
filevault::algorithms::pqc::Dilithium, 200
Dilithium3
filevault::algorithms::pqc::Dilithium, 200
Dilithium5
filevault::algorithms::pqc::Dilithium, 200
DILITHIUM_2
filevault::core, 58
DILITHIUM_3
filevault::core, 58
DILITHIUM_5
filevault::core, 58
disable_echo
filevault::utils::Password, 306
display_info
filevault::cli::InfoCommand, 274
display_strength_meter
filevault::utils::Password, 307
do_capacity
filevault::cli::commands::StegoCommand, 350
do_compress
filevault::cli::CompressCommand, 158
do_create
filevault::cli::commands::ArchiveCommand, 107
do_decompress
filevault::cli::CompressCommand, 158
do_embed
filevault::cli::commands::StegoCommand, 351
do_extract
filevault::cli::commands::ArchiveCommand, 108
filevault::cli::commands::StegoCommand, 352
DumpCommand
filevault::cli::commands::DumpCommand, 204

ECC_P256
filevault::core, 57, 58
ECC_P384
filevault::core, 57, 58
ECC_P521
filevault::core, 57, 58
ECCHybrid
filevault::algorithms::asymmetric::ECCHybrid, 208
ECCurve
filevault::algorithms::asymmetric, 48
ECDH
filevault::algorithms::asymmetric::ECDH, 212
ecdh_
filevault::algorithms::asymmetric::ECCHybrid, 211
ECDSA
filevault::algorithms::asymmetric::ECDSA, 215
embed
filevault::steganography::LSBSteganography, 298
embed_byte
filevault::steganography::LSBSteganography, 298
enable_echo
filevault::utils::Password, 307
encaps_ms
filevault::cli::PQCBenchmarkResult, 320
encrypt
filevault::algorithms::asymmetric::ECCHybrid, 209
filevault::algorithms::asymmetric::RSA, 329
filevault::algorithms::classical::Caesar, 142
filevault::algorithms::classical::HillCipher, 260
filevault::algorithms::classical::Playfair, 317, 318
filevault::algorithms::classical::SubstitutionCipher, 363
filevault::algorithms::classical::Vigenere, 389
filevault::algorithms::pqc::Kyber, 287
filevault::algorithms::pqc::KyberHybrid, 292
filevault::algorithms::symmetric::AES_CBC, 68

filevault::algorithms::symmetric::AES_CFB, 73
filevault::algorithms::symmetric::AES_CTR, 78
filevault::algorithms::symmetric::AES_ECB, 84
filevault::algorithms::symmetric::AES_GCM, 89
filevault::algorithms::symmetric::AES_OFB, 94
filevault::algorithms::symmetric::AES_XTS, 100
filevault::algorithms::symmetric::ARIA_GCM, 120
filevault::algorithms::symmetric::Camellia_GCM, 147
filevault::algorithms::symmetric::ChaCha20Poly1305, 152
filevault::algorithms::symmetric::Serpent_GCM, 336
filevault::algorithms::symmetric::SM4_GCM, 346
filevault::algorithms::symmetric::TripleDES, 373
filevault::algorithms::symmetric::Twofish_GCM, 378
filevault::core::ICryptoAlgorithm, 270
encrypt_block
filevault::algorithms::classical::HillCipher, 260
encrypt_file
filevault::core::StreamingCrypto, 357
encrypt_mbps
filevault::cli::BenchmarkResult, 135
encrypt_ms
filevault::cli::AsymmetricBenchmarkResult, 123
filevault::cli::BenchmarkResult, 135
EncryptCommand
filevault::cli::EncryptCommand, 220
encrypted_size
filevault::format::FileHeader, 238
encrypted_size_
filevault::format::FileHeader, 245
engine_
filevault::cli::Application, 105
filevault::cli::BenchmarkCommand, 134
filevault::cli::commands::ArchiveCommand, 111
filevault::cli::commands::KeyInfoCommand, 283
filevault::cli::commands::SignCommand, 343
filevault::cli::commands::VerifyCommand, 385
filevault::cli::DecryptCommand, 198
filevault::cli::EncryptCommand, 223
filevault::cli::HashCommand, 255
filevault::cli::InfoCommand, 276
filevault::cli::KeygenCommand, 280
filevault::cli::ListCommand, 297
error
filevault::core::Result< T >, 324
filevault::core::Result< void >, 326
filevault::utils::Console, 176
error_message
filevault::algorithms::asymmetric::ECDHResult, 214
filevault::algorithms::asymmetric::ECDSASignResult, 218
filevault::compression::CompressionResult, 162
filevault::core::CryptoResult, 188
filevault::core::Result< T >, 325
filevault::core::Result< void >, 326
filevault::core::StreamingResult, 360
estimate_crack_time
filevault::utils::Password, 308
estimate_key_length
filevault::algorithms::classical::Vigenere, 390
execute
filevault::cli::BenchmarkCommand, 132
filevault::cli::commands::ArchiveCommand, 109
filevault::cli::commands::DumpCommand, 205
filevault::cli::commands::KeyInfoCommand, 282
filevault::cli::commands::SignCommand, 341
filevault::cli::commands::StegoCommand, 352
filevault::cli::commands::VerifyCommand, 384
filevault::cli::CompressCommand, 159
filevault::cli::ConfigCommand, 172
filevault::cli::DecompressCommand, 192
filevault::cli::DecryptCommand, 196
filevault::cli::EncryptCommand, 220
filevault::cli::HashCommand, 252
filevault::cli::ICommand, 266
filevault::cli::InfoCommand, 274
filevault::cli::KeygenCommand, 278
filevault::cli::ListCommand, 295
expected_hash
filevault::core::HashConfig, 256
extract
filevault::steganography::LSBSteganography, 299
extract_
filevault::cli::commands::ArchiveCommand, 112
extract_archive
filevault::archive::ArchiveFormat, 113
extract_byte
filevault::steganography::LSBSteganography, 299
extract_dir_
filevault::cli::commands::ArchiveCommand, 112

FAIR
filevault::core, 60
file_exists
filevault::utils::FileIO, 247
FILE_FORMAT_MAGIC
filevault::core, 61
FILE_FORMAT_VERSION_MAJOR
filevault::core, 61
FILE_FORMAT_VERSION_MINOR
filevault::core, 61
file_info_table
filevault::utils::TableFormatter, 369
file_path_
filevault::cli::commands::DumpCommand, 206
filevault::cli::commands::SignCommand, 343
filevault::cli::commands::VerifyCommand, 385
file_size
filevault::archive::FileEntry, 227
filevault::cli::InfoCommand::FileInfo, 246
filevault::utils::FileIO, 248
FileHeader
filevault::format::FileHeader, 237

filename
filevault::archive::FileEntry, 227
FILENO
progress.cpp, 545
FileVault, 1
filevault, 47
filevault::algorithms, 47
filevault::algorithms::asymmetric, 47
ECCurve, 48
get_botan_curve_name, 48
get_curve_key_size, 48
SECP256R1, 48
SECP384R1, 48
SECP521R1, 48
X25519, 48
filevault::algorithms::asymmetric::ECCHybrid, 206
∼ECCHybrid, 208
botan_curve_name_, 211
curve_, 211
decrypt, 209
ECCHybrid, 208
ecdh_, 211
encrypt, 209
generate_key_pair, 209
is_suitable_for, 210
key_size, 210
name, 210
type, 210
type_, 211
filevault::algorithms::asymmetric::ECCKeyPair, 211
curve, 211
curve_name, 211
private_key, 211
public_key, 211
filevault::algorithms::asymmetric::ECDH, 212
∼ECDH, 212
botan_curve_name_, 214
curve_, 214
curve_name, 213
derive_shared_secret, 213
ECDH, 212
generate_key_pair, 213
key_size, 213
name, 214
filevault::algorithms::asymmetric::ECDHResult, 214
error_message, 214
shared_secret, 214
success, 214
filevault::algorithms::asymmetric::ECDSA, 215
∼ECDSA, 216
botan_curve_name_, 218
curve_, 218
curve_name, 216
ECDSA, 215
generate_key_pair, 216
key_size, 216
name, 216
sign, 217
signature_size, 217
verify, 217
filevault::algorithms::asymmetric::ECDSASignResult, 218
error_message, 218
signature, 218
success, 218
filevault::algorithms::asymmetric::RSA, 326
∼RSA, 328
decrypt, 329
encrypt, 329
generate_key_pair, 330
is_authenticated, 330
is_suitable_for, 330
key_bits_, 332
key_size, 330
max_plaintext_size, 330
name, 330
requires_padding, 331
RSA, 328
sign, 331
type, 331
type_, 332
verify, 331
filevault::algorithms::asymmetric::RSAKeyPair, 332
bits, 332
private_key, 332
public_key, 332
filevault::algorithms::classical, 49
filevault::algorithms::classical::Caesar, 138
brute_force, 141
Caesar, 140
decrypt, 141, 142
encrypt, 142
frequency_analysis, 143
is_suitable_for, 143
key_size, 143
name, 143
shift_, 144
shift_char, 144
type, 144
filevault::algorithms::classical::HillCipher, 257
decrypt, 258
decrypt_block, 259
determinant, 260
encrypt, 260
encrypt_block, 260
invert_matrix, 261
is_suitable_for, 261
is_valid_key, 262
key_size, 262
Matrix2x2, 258
mod_inverse, 262
name, 263
parse_key, 263
type, 263
filevault::algorithms::classical::Playfair, 313
build_matrix, 315, 316

decrypt, 316, 317
encrypt, 317, 318
find_position, 318, 319
is_suitable_for, 319
key_size, 319
matrix_, 320
name, 319
Playfair, 315
type, 320
filevault::algorithms::classical::SubstitutionCipher, 360
apply_substitution, 362
create_reverse_map, 362
decrypt, 363
encrypt, 363
is_suitable_for, 364
is_valid_key, 364
key_size, 365
name, 365
parse_key, 365
SubstitutionMap, 362
type, 365
filevault::algorithms::classical::Vigenere, 385
decrypt, 388
encrypt, 389
estimate_key_length, 390
is_suitable_for, 390
kasiski_examination, 390
key_size, 390
keyword_, 391
name, 390
type, 390, 391
Vigenere, 387, 388
filevault::algorithms::pqc, 49
get_dilithium_mode, 50
get_kyber_mode, 50
filevault::algorithms::pqc::Dilithium, 199
botan_name_, 203
Dilithium, 200
Dilithium2, 200
Dilithium3, 200
Dilithium5, 200
generate_keypair, 200
name, 201
private_key_size, 201
public_key_size, 201
sign, 201
signature_size, 202
Variant, 200
variant_, 203
verify, 202
filevault::algorithms::pqc::Kyber, 284
~Kyber, 286
botan_name_, 289
ciphertext_size, 286
decrypt, 286
encrypt, 287
generate_keypair, 288
is_suitable_for, 288
key_size, 288
Kyber, 286
Kyber1024, 286
Kyber512, 286
Kyber768, 286
name, 288
private_key_size, 289
public_key_size, 289
shared_secret_size, 289
type, 289
type_, 289
Variant, 286
variant_, 289
filevault::algorithms::pqc::KyberHybrid, 289
~KyberHybrid, 291
decrypt, 291
encrypt, 292
generate_keypair, 293
is_suitable_for, 293
key_size, 293
kyber_, 294
KyberHybrid, 291
name, 293
type, 293
filevault::algorithms::pqc::PQKeyPair, 321
algorithm, 321
private_key, 321
public_key, 321
filevault::algorithms::symmetric, 50
add_pkcs7_padding, 51
remove_pkcs7_padding, 51
filevault::algorithms::symmetric::AES_CBC, 65
~AES_CBC, 66
AES_CBC, 66
block_size, 67
botan_name_, 70
decrypt, 67
encrypt, 68
is_suitable_for, 69
iv_size, 69
key_bits_, 70
key_size, 69
name, 70
type, 70
type_, 70
filevault::algorithms::symmetric::AES_CFB, 70
~AES_CFB, 72
AES_CFB, 72
block_size, 72
botan_name_, 75
decrypt, 72
encrypt, 73
is_authenticated, 74
is_suitable_for, 74
iv_size, 74
key_bits_, 75
key_size, 74
name, 75

requires_padding, 75
type, 75
type_, 75
filevault::algorithms::symmetric::AES_CTR, 76
~AES_CTR, 77
AES_CTR, 77
botan_name_, 80
decrypt, 77
encrypt, 78
is_suitable_for, 79
key_bits_, 80
key_size, 79
name, 79
nonce_size, 80
type, 80
type_, 80
filevault::algorithms::symmetric::AES_ECB, 80
~AES_ECB, 82
AES_ECB, 82
block_size, 82
botan_name_, 86
decrypt, 83
encrypt, 84
is_authenticated, 85
is_suitable_for, 85
key_bits_, 86
key_size, 85
name, 86
requires_padding, 86
type, 86
type_, 86
filevault::algorithms::symmetric::AES_GCM, 86
~AES_GCM, 88
AES_GCM, 88
botan_name_, 91
decrypt, 88
encrypt, 89
is_suitable_for, 90
key_bits_, 91
key_size, 90
name, 90
nonce_size, 91
tag_size, 91
type, 91
type_, 91
filevault::algorithms::symmetric::AES_OFB, 92
~AES_OFB, 93
AES_OFB, 93
block_size, 94
botan_name_, 96
decrypt, 94
encrypt, 94
is_authenticated, 95
is_suitable_for, 95
iv_size, 95
key_bits_, 96
key_size, 95
name, 96
requires_padding, 96
type, 96
type_, 96
filevault::algorithms::symmetric::AES_XTS, 97
~AES_XTS, 98
AES_XTS, 98
block_size, 98
botan_name_, 103
decrypt, 99
encrypt, 100
is_authenticated, 101
is_suitable_for, 101
key_bits_, 103
key_size, 101
name, 102
requires_padding, 102
tweak_size, 102
type, 102
type_, 103
filevault::algorithms::symmetric::ARIA_GCM, 117
~ARIA_GCM, 119
ARIA_GCM, 119
botan_name_, 123
decrypt, 119
encrypt, 120
is_suitable_for, 121
key_bits_, 123
key_size, 121
name, 121
nonce_size, 122
tag_size, 122
type, 122
type_, 123
filevault::algorithms::symmetric::Camellia_GCM, 145
~Camellia_GCM, 146
botan_name_, 150
Camellia_GCM, 146
decrypt, 146
encrypt, 147
is_suitable_for, 148
key_bits_, 150
key_size, 148
name, 148
nonce_size, 149
tag_size, 149
type, 149
type_, 150
filevault::algorithms::symmetric::ChaCha20Poly1305, 150
~ChaCha20Poly1305, 151
ChaCha20Poly1305, 151
decrypt, 152
encrypt, 152
is_suitable_for, 153
key_size, 153
name, 154
nonce_size, 154
tag_size, 154

type, 155
filevault::algorithms::symmetric::Serpent_GCM, 334
decrypt, 336
encrypt, 336
is_suitable_for, 337
key_size, 337
name, 338
process_gcm, 338
Serpent_GCM, 335
type, 339
filevault::algorithms::symmetric::SM4_GCM, 343
~SM4_GCM, 345
decrypt, 345
encrypt, 346
is_suitable_for, 347
key_size, 347
name, 347
nonce_size, 348
SM4_GCM, 345
tag_size, 348
type, 348
filevault::algorithms::symmetric::TripleDES, 371
~TripleDES, 372
block_size, 372
decrypt, 372
encrypt, 373
is_suitable_for, 374
iv_size, 374
key_size, 374
name, 375
TripleDES, 372
type, 375
filevault::algorithms::symmetric::Twofish_GCM, 375
botan_name_, 382
decrypt, 378
encrypt, 378
is_suitable_for, 379
key_bits_, 382
key_size, 380
name, 380
nonce_size, 380
process_gcm, 380
tag_size, 381
Twofish_GCM, 377
type, 381
type_, 382
filevault::archive, 51
filevault::archive::ArchiveFormat, 112
create_archive, 113
extract_archive, 113
list_files, 114
MAGIC, 116
read_string, 114
read_uint32, 115
read_uint64, 115
VERSION, 116
write_uint32, 115
write_uint64, 115
filevault::archive::FileEntry, 226
deserialize, 227
file_size, 227
filename, 227
modified_time, 227
offset, 227
permissions, 227
serialize, 227
filevault::cli, 52
filevault::cli::Application, 103
~Application, 103
app_, 105
Application, 103
commands_, 105
engine_, 105
initialize, 104
log_level_, 105
register_commands, 104
run, 104
setup_logging, 105
verbose_, 105
filevault::cli::AsymmetricBenchmarkResult, 123
algorithm, 123
decrypt_ms, 123
encrypt_ms, 123
keygen_ms, 123
success, 123
filevault::cli::BenchmarkCommand, 124
algorithm_, 134
all_, 134
asymmetric_only_, 134
benchmark_algorithm, 125
benchmark_asymmetric, 125
benchmark_asymmetric_algorithm, 126
benchmark_compression, 127
benchmark_hash, 128
benchmark_hybrid_algorithm, 128
benchmark_kdf, 129
benchmark_pqc, 129
benchmark_pqc_algorithm, 130
benchmark_signature_algorithm, 130
benchmark_symmetric, 131
BenchmarkCommand, 125
compression_only_, 134
data_size_, 134
description, 131
engine_, 134
execute, 132
get_platform_info, 132
hash_only_, 134
iterations_, 134
json_output_, 134
kdf_only_, 135
name, 133
output_file_, 135
pqc_only_, 135
save_json_output, 133
save_log_output, 133

setup, 133
symmetric_only_, 135
filevault::cli::BenchmarkResult, 135
algorithm, 135
decrypt_mbps, 135
decrypt_ms, 135
encrypt_mbps, 135
encrypt_ms, 135
success, 135
filevault::cli::commands, 52
filevault::cli::commands::ArchiveCommand, 106
algorithm_, 111
ArchiveCommand, 107
compression_, 111
description, 107
do_create, 107
do_extract, 108
engine_, 111
execute, 109
extract_, 112
extract_dir_, 112
input_files_, 112
kdf_, 112
name, 110
output_file_, 112
password_, 112
security_level_, 112
setup, 111
verbose_, 112
filevault::cli::commands::DumpCommand, 203
description, 205
description_, 206
DumpCommand, 204
execute, 205
file_path_, 206
format_, 206
max_bytes_, 206
name, 205
name_, 206
setup, 205
show_ascii_, 206
show_offset_, 206
filevault::cli::commands::KeyInfoCommand, 281
check_pair_, 283
description, 282
description_, 283
engine_, 283
execute, 282
key_path_, 283
KeyInfoCommand, 282
name, 283
name_, 283
pair_key_path_, 284
setup, 283
show_public_, 284
filevault::cli::commands::SignCommand, 340
algorithm_, 343
description, 341
description_, 343
engine_, 343
execute, 341
file_path_, 343
name, 342
name_, 343
output_path_, 343
private_key_path_, 343
setup, 342
SignCommand, 341
filevault::cli::commands::StegoCommand, 349
bits_per_channel_, 354
cover_image_, 354
description, 350
do_capacity, 350
do_embed, 351
do_extract, 352
execute, 352
input_file_, 354
name, 353
operation_, 354
output_file_, 354
setup, 353
verbose_, 354
filevault::cli::commands::VerifyCommand, 382
algorithm_, 385
description, 384
description_, 385
engine_, 385
execute, 384
file_path_, 385
name, 384
name_, 385
public_key_path_, 385
setup, 384
signature_path_, 385
VerifyCommand, 383
filevault::cli::CompressCommand, 156
~CompressCommand, 157
algorithm_, 161
auto_detect_, 161
benchmark_, 161
CompressCommand, 157
decompress_, 161
description, 157
detect_algorithm, 157
do_compress, 158
do_decompress, 158
execute, 159
generate_output_path, 160
input_file_, 161
level_, 162
name, 160
output_file_, 162
setup, 161
subcommand_, 162
verbose_, 162
filevault::cli::ConfigCommand, 170

ConfigCommand, 172
description, 172
execute, 172
key_, 175
name, 173
reset_config, 173
set_value, 174
setup, 174
show_config, 174
subcommand_, 175
value_, 175
filevault::cli::DecompressCommand, 190
algorithm_, 194
auto_detect_, 194
benchmark_, 194
description, 191
detect_algorithm, 191
execute, 192
generate_output_path, 192
input_file_, 194
name, 193
output_file_, 194
setup, 193
subcommand_, 194
verbose_, 194
filevault::cli::DecryptCommand, 195
DecryptCommand, 196
description, 196
engine_, 198
execute, 196
input_file_, 198
name, 197
no_progress_, 199
output_file_, 199
password_, 199
setup, 198
verbose_, 199
filevault::cli::EncryptCommand, 218
algorithm_, 223
compression_level_, 223
compression_type_, 223
description, 220
EncryptCommand, 220
engine_, 223
execute, 220
force_weak_password_, 223
input_file_, 223
kdf_, 223
mode_, 223
name, 221
no_progress_, 223
output_file_, 223
password_, 223
security_level_, 223
setup, 222
verbose_, 223
filevault::cli::HashCommand, 250
algorithm_, 255
benchmark_, 255
calculate_file_hash, 251
calculate_file_hmac, 251
description, 252
engine_, 255
execute, 252
get_botan_algorithm_name, 253
HashCommand, 251
hmac_key_, 255
input_file_, 255
is_secure_algorithm, 253
name, 254
no_filename_, 255
output_file_, 255
output_format_, 255
setup, 254
uppercase_, 256
verbose_, 256
verify_hash_, 256
verify_mode, 254
filevault::cli::ICommand, 264
~ICommand, 266
description, 266
execute, 266
name, 266
setup, 266
filevault::cli::InfoCommand, 272
description, 274
display_info, 274
engine_, 276
execute, 274
InfoCommand, 273
input_file_, 276
name, 275
parse_file, 275
setup, 276
verbose_, 276
filevault::cli::InfoCommand::FileInfo, 246
algorithm, 246
compressed, 246
compression, 246
data_size, 246
file_size, 246
has_header, 246
header_size, 247
kdf, 247
nonce_size, 247
salt_size, 247
tag_size, 247
version, 247
filevault::cli::KeygenCommand, 277
algorithm_, 280
description, 278
engine_, 280
execute, 278
force_, 280
KeygenCommand, 278
name, 279

output_prefix_, 280
setup, 279
verbose_, 280
filevault::cli::ListCommand, 294
description, 295
engine_, 297
execute, 295
ListCommand, 295
name, 296
setup, 296
filevault::cli::PQCBenchmarkResult, 320
algorithm, 320
decaps_ms, 320
encaps_ms, 320
keygen_ms, 320
success, 321
filevault::cli::SignatureBenchmarkResult, 339
algorithm, 339
keygen_ms, 339
sign_ms, 340
success, 340
verify_ms, 340
filevault::compression, 53
filevault::compression::Bzip2Compressor, 137
compress, 138
decompress, 138
name, 138
filevault::compression::CompressionResult, 162
compressed_size, 162
compression_ratio, 162
data, 162
error_message, 162
original_size, 162
processing_time_ms, 163
success, 163
filevault::compression::CompressionService, 163
create, 163
get_algorithm_name, 164
parse_algorithm, 164
filevault::compression::ICompressor, 267
~ICompressor, 267
compress, 267
decompress, 268
name, 268
filevault::compression::LzmaCompressor, 300
compress, 301
decompress, 301
name, 301
filevault::compression::ZlibCompressor, 391
compress, 392
decompress, 392
name, 393
filevault::core, 53
ADVANCED, 60
AES_128_CBC, 56, 57
AES_128_CFB, 56, 57
AES_128_CTR, 56, 57
AES_128_ECB, 56, 57
AES_128_GCM, 56, 57
AES_128_OFB, 56, 57
AES_128_XTS, 56, 57
AES_192_CBC, 56, 57
AES_192_CFB, 56, 57
AES_192_CTR, 56, 57
AES_192_ECB, 56, 57
AES_192_GCM, 56, 57
AES_192_OFB, 56, 57
AES_256_CBC, 56, 57
AES_256_CFB, 56, 57
AES_256_CTR, 56, 57
AES_256_ECB, 56, 57
AES_256_GCM, 56, 57
AES_256_OFB, 56, 57
AES_256_XTS, 56, 58
AlgorithmID, 55
AlgorithmType, 57
ARGON2I, 59, 60
ARGON2ID, 59, 60
ARIA_128_GCM, 56, 57
ARIA_192_GCM, 56, 57
ARIA_256_GCM, 56, 57
BLAKE2B_256, 59
BLAKE2B_384, 59
BLAKE2B_512, 59
BLAKE2S_256, 59
BZIP2, 58, 59
CAESAR, 56, 58
CAMELLIA_128_GCM, 56, 57
CAMELLIA_192_GCM, 56, 57
CAMELLIA_256_GCM, 56, 57
CHACHA20_POLY1305, 56, 57
CompressionID, 58
CompressionType, 58
DILITHIUM_2, 58
DILITHIUM_3, 58
DILITHIUM_5, 58
ECC_P256, 57, 58
ECC_P384, 57, 58
ECC_P521, 57, 58
FAIR, 60
FILE_FORMAT_MAGIC, 61
FILE_FORMAT_VERSION_MAJOR, 61
FILE_FORMAT_VERSION_MINOR, 61
HashType, 59
HILL, 56, 58
KDFID, 59
KDFType, 59
KYBER_1024, 58
KYBER_1024_HYBRID, 58
KYBER_512, 58
KYBER_512_HYBRID, 58
KYBER_768, 58
KYBER_768_HYBRID, 58
LZMA, 58, 59
MD5, 59
MEDIUM, 60

NONE, 58, 59
PARANOID, 60
PasswordStrength, 60
PBKDF2_SHA256, 59, 60
PBKDF2_SHA512, 59, 60
PLAYFAIR, 56, 58
PROFESSIONAL, 60
RSA_2048, 56, 58
RSA_3072, 56, 58
RSA_4096, 57, 58
SCRYPT, 59, 60
SecurityLevel, 60
SERPENT_256_GCM, 56, 57
SHA1, 59
SHA224, 59
SHA256, 59
SHA384, 59
SHA3_224, 59
SHA3_256, 59
SHA3_384, 59
SHA3_512, 59
SHA512, 59
SHA512_256, 59
SM4_GCM, 56, 57
STREAM_MAGIC, 61
STREAM_VERSION, 61
StreamProgressCallback, 55
STRONG, 60
STUDENT, 60
SUBSTITUTION, 56, 58
TRIPLE_DES_CBC, 56, 58
TWOFISH_128_GCM, 56, 57
TWOFISH_192_GCM, 56, 57
TWOFISH_256_GCM, 56, 57
UNKNOWN, 56
UserMode, 60
VERY_STRONG, 60
VERY_WEAK, 60
VIGENERE, 56, 58
WEAK, 60
ZLIB, 58, 59
filevault::core::Argon2Params, 116
deserialize, 116
iterations, 117
memory_kb, 117
parallelism, 117
serialize, 116
filevault::core::ChunkInfo, 155
bytes_processed, 155
chunk_index, 155
chunk_size, 155
total_bytes, 155
total_chunks, 155
filevault::core::CryptoEngine, 182
~CryptoEngine, 183
algorithm_name, 183
algorithms_, 187
CryptoEngine, 183
derive_key, 183
generate_nonce, 184
generate_salt, 185
get_algorithm, 185
initialize, 185
kdf_name, 186
operator=, 186
parse_algorithm, 186
parse_kdf, 187
parse_security_level, 187
register_algorithm, 187
security_level_name, 187
filevault::core::CryptoResult, 188
algorithm_used, 188
data, 188
error_message, 188
final_size, 188
nonce, 188
original_size, 188
processing_time_ms, 188
salt, 188
success, 188
tag, 189
filevault::core::EncryptionConfig, 224
algorithm, 225
apply_security_level, 224
apply_user_mode, 225
associated_data, 225
comment, 225
compression, 225
compression_level, 225
include_metadata, 225
kdf, 226
kdf_iterations, 226
kdf_memory_kb, 226
kdf_parallelism, 226
level, 226
mode, 226
nonce, 226
salt, 226
show_progress, 226
tag, 226
verbose, 226
filevault::core::FileFormatHandler, 228
create_header, 228
from_algorithm_id, 229
from_compression_id, 229
from_kdf_id, 230
is_legacy_format, 230
read_file, 230
read_legacy_file, 231
to_algorithm_id, 231
to_compression_id, 231
to_kdf_id, 231
write_file, 232
filevault::core::FileHeader, 232
algorithm, 234
compressed, 234

compression, 235
deserialize, 233
is_valid, 233
kdf, 235
kdf_params, 235
magic, 235
nonce, 235
reserved, 235
salt, 235
serialize, 234
size, 234
version_major, 235
version_minor, 235
filevault::core::HashConfig, 256
algorithm, 256
expected_hash, 256
hmac_key, 256
hmac_mode, 256
include_filename, 256
uppercase, 256
verify_mode, 257
filevault::core::ICryptoAlgorithm, 268
~ICryptoAlgorithm, 270
decrypt, 270
encrypt, 270
is_suitable_for, 271
key_size, 271
name, 271
type, 272
filevault::core::ModePreset, 302
algorithm, 304
apply_to, 302
compression, 304
compression_level, 304
description, 303
get_all_presets, 303
get_preset, 303
kdf, 304
kdf_iterations, 304
kdf_memory_kb, 304
kdf_parallelism, 304
mode, 304
name, 303
parse_mode, 303
security_level, 304
filevault::core::PasswordAnalysis, 311
crack_time_offline, 311
crack_time_online, 311
has_digits, 311
has_lowercase, 311
has_repeated_chars, 311
has_special, 311
has_uppercase, 311
is_common_password, 311
length, 311
score, 312
strength, 312
suggestions, 312
warnings, 312
filevault::core::PBKDF2Params, 312
deserialize, 312
iterations, 313
serialize, 312
filevault::core::presets, 61
ADVANCED, 61
BASIC, 61
STANDARD, 62
filevault::core::Result< T >, 323
error, 324
error_message, 325
ok, 324
operator bool, 325
success, 325
unwrap, 325
value, 325
filevault::core::Result< void >, 325
error, 326
error_message, 326
ok, 326
operator bool, 326
success, 326
unwrap, 326
value, 326
filevault::core::ScryptParams, 333
deserialize, 333
n, 333
p, 333
r, 333
serialize, 333
filevault::core::StreamingConfig, 354
algorithm, 355
chunk_size, 355
compression, 355
compression_level, 355
kdf, 355
level, 355
progress_callback, 355
filevault::core::StreamingCrypto, 355
decrypt_file, 356
derive_chunk_nonce, 357
encrypt_file, 357
get_recommended_chunk_size, 358
read_stream_header, 358
should_use_streaming, 359
write_stream_header, 359
filevault::core::StreamingResult, 360
bytes_processed, 360
chunks_processed, 360
error_message, 360
processing_time_ms, 360
success, 360
throughput_mbps, 360
filevault::format, 62
filevault::format::FileHeader, 235
algorithm, 237
algorithm_, 245

deserialize, 237
encrypted_size, 238
encrypted_size_, 245
FileHeader, 237
FLAG_COMPRESSED, 245
FLAG_METADATA, 245
flags, 238
flags_, 245
is_compressed, 238
kdf, 238
kdf_, 245
MAGIC, 245
MIN_HEADER_SIZE, 245
nonce, 239
nonce_, 245
original_size, 239
original_size_, 245
read_uint16, 239
read_uint32, 239
read_uint64, 240
reserved_, 245
salt, 240
salt_, 245
security_level, 240
security_level_, 245
serialize, 240
set_algorithm, 241
set_compressed, 241
set_encrypted_size, 241
set_kdf, 241
set_nonce, 241
set_original_size, 242
set_salt, 242
set_security_level, 242
set_tag, 242
set_timestamp, 243
tag, 243
tag_, 245
timestamp, 243
timestamp_, 245
total_size, 243
validate, 243
VERSION_MAJOR, 245
VERSION_MINOR, 246
write_uint16, 243
write_uint32, 244
write_uint64, 244
filevault::steganography, 62
filevault::steganography::LSBSteganography, 297
calculate_capacity, 297
embed, 298
embed_byte, 298
extract, 299
extract_byte, 299
LENGTH_HEADER_SIZE, 300
filevault::utils, 62
is_terminal, 63
mingw_data, 63
filevault::utils::BlockProgressBar, 136
bar_, 136
BlockProgressBar, 136
mark_as_completed, 136
set_option_text, 136
set_progress, 136
filevault::utils::Config, 165
compression_level_, 170
default_algorithm_, 170
default_compression_, 170
default_kdf_, 170
default_mode_, 170
from_json, 166
get, 166
get_compression_level, 166
get_config_path, 166
get_default, 167
get_default_algorithm, 167
get_default_compression, 167
get_default_kdf, 167
get_default_mode, 167
get_show_progress, 167
get_verbose, 167
load, 167
reset, 168
save, 168
set, 169
set_compression_level, 169
set_default_algorithm, 169
set_default_compression, 169
set_default_kdf, 169
set_default_mode, 169
set_show_progress, 169
set_verbose, 169
show_progress_, 170
to_json, 169
verbose_, 170
filevault::utils::Console, 175
debug, 176
error, 176
header, 177
info, 178
separator, 179
success, 180
warning, 181
filevault::utils::CryptoUtils, 189
format_bytes, 189
hex_decode, 189
hex_encode, 189
filevault::utils::FileIO, 247
file_exists, 247
file_size, 248
read_file, 248
write_file, 249
filevault::utils::Password, 305
analyze_strength, 305
calculate_entropy, 306
common_passwords_, 310

disable_echo, 306
display_strength_meter, 307
enable_echo, 307
estimate_crack_time, 308
get_strength_color, 308
get_strength_label, 308
is_common_password, 309
read_secure, 309
filevault::utils::ProgressBar, 321
~ProgressBar, 322
bar_, 323
current_progress_, 323
hide, 322
mark_as_completed, 322
max_progress_, 323
ProgressBar, 322
set_postfix, 322
set_progress, 322
show, 323
tick, 323
filevault::utils::TableBuilder, 366
print_algorithm_list, 366
print_benchmark_results, 366
print_encryption_config, 367
print_file_summary, 367
filevault::utils::TableFormatter, 367
add_row, 368
algorithm_list_table, 368
benchmark_table, 369
file_info_table, 369
print, 370
set_border_style, 370
set_column_alignment, 370
set_column_format, 370
table_, 370
TableFormatter, 368
to_string, 370
final_size
filevault::core::CryptoResult, 188
find_position
filevault::algorithms::classical::Playfair, 318, 319
FLAG_COMPRESSED
filevault::format::FileHeader, 245
FLAG_METADATA
filevault::format::FileHeader, 245
flags
filevault::format::FileHeader, 238
flags_
filevault::format::FileHeader, 245
force_
filevault::cli::KeygenCommand, 280
force_weak_password_
filevault::cli::EncryptCommand, 223
format_
filevault::cli::commands::DumpCommand, 206
format_bytes
filevault::utils::CryptoUtils, 189
frequency_analysis
filevault::algorithms::classical::Caesar, 143
from_algorithm_id
filevault::core::FileFormatHandler, 229
from_compression_id
filevault::core::FileFormatHandler, 229
from_json
filevault::utils::Config, 166
from_kdf_id
filevault::core::FileFormatHandler, 230

generate_key_pair
filevault::algorithms::asymmetric::ECCHybrid, 209
filevault::algorithms::asymmetric::ECDH, 213
filevault::algorithms::asymmetric::ECDSA, 216
filevault::algorithms::asymmetric::RSA, 330
generate_keypair
filevault::algorithms::pqc::Dilithium, 200
filevault::algorithms::pqc::Kyber, 288
filevault::algorithms::pqc::KyberHybrid, 293
generate_nonce
filevault::core::CryptoEngine, 184
generate_output_path
filevault::cli::CompressCommand, 160
filevault::cli::DecompressCommand, 192
generate_salt
filevault::core::CryptoEngine, 185
get
filevault::utils::Config, 166
get_algorithm
filevault::core::CryptoEngine, 185
get_algorithm_name
filevault::compression::CompressionService, 164
get_all_presets
filevault::core::ModePreset, 303
get_botan_algorithm_name
filevault::cli::HashCommand, 253
get_botan_curve_name
filevault::algorithms::asymmetric, 48
get_compression_level
filevault::utils::Config, 166
get_config_path
filevault::utils::Config, 166
get_curve_key_size
filevault::algorithms::asymmetric, 48
get_default
filevault::utils::Config, 167
get_default_algorithm
filevault::utils::Config, 167
get_default_compression
filevault::utils::Config, 167
get_default_kdf
filevault::utils::Config, 167
get_default_mode
filevault::utils::Config, 167
get_dilithium_mode
filevault::algorithms::pqc, 50
get_kyber_mode
filevault::algorithms::pqc, 50
get_platform_info

filevault::cli::BenchmarkCommand, 132
get_preset
filevault::core::ModePreset, 303
get_recommended_chunk_size
filevault::core::StreamingCrypto, 358
get_show_progress
filevault::utils::Config, 167
get_strength_color
filevault::utils::Password, 308
get_strength_label
filevault::utils::Password, 308
get_verbose
filevault::utils::Config, 167

has_digits
filevault::core::PasswordAnalysis, 311
has_header
filevault::cli::InfoCommand::FileInfo, 246
has_lowercase
filevault::core::PasswordAnalysis, 311
has_repeated_chars
filevault::core::PasswordAnalysis, 311
has_special
filevault::core::PasswordAnalysis, 311
has_uppercase
filevault::core::PasswordAnalysis, 311
hash_only_
filevault::cli::BenchmarkCommand, 134
HashCommand
filevault::cli::HashCommand, 251
HashType
filevault::core, 59
header
filevault::utils::Console, 177
header_size
filevault::cli::InfoCommand::FileInfo, 247
hex_decode
filevault::utils::CryptoUtils, 189
hex_encode
filevault::utils::CryptoUtils, 189
hide
filevault::utils::ProgressBar, 322
HILL
filevault::core, 56, 58
hmac_key
filevault::core::HashConfig, 256
hmac_key_
filevault::cli::HashCommand, 255
hmac_mode
filevault::core::HashConfig, 256

include Directory Reference, 39
include/filevault Directory Reference, 38
include/filevault/algorithms Directory Reference, 29
include/filevault/algorithms/asymmetric Directory Reference, 31
include/filevault/algorithms/asymmetric/ecc.hpp, 395, 396
include/filevault/algorithms/asymmetric/rsa.hpp, 398, 399
include/filevault/algorithms/classical Directory Reference, 32
include/filevault/algorithms/classical/caesar.hpp, 400
include/filevault/algorithms/classical/classical_ciphers.hpp, 401, 402
include/filevault/algorithms/classical/hill.hpp, 403, 404
include/filevault/algorithms/classical/playfair.hpp, 405, 406
include/filevault/algorithms/classical/substitution.hpp, 407, 408
include/filevault/algorithms/classical/vigenere.hpp, 409, 411
include/filevault/algorithms/pqc Directory Reference, 40
include/filevault/algorithms/pqc/post_quantum.hpp, 411, 413
include/filevault/algorithms/symmetric Directory Reference, 42
include/filevault/algorithms/symmetric/aes_cbc.hpp, 414, 416
include/filevault/algorithms/symmetric/aes_cfb.hpp, 416, 418
include/filevault/algorithms/symmetric/aes_ctr.hpp, 418, 420
include/filevault/algorithms/symmetric/aes_ecb.hpp, 420, 422
include/filevault/algorithms/symmetric/aes_gcm.hpp, 422, 423
include/filevault/algorithms/symmetric/aes_ofb.hpp, 424, 426
include/filevault/algorithms/symmetric/aes_xts.hpp, 426, 428
include/filevault/algorithms/symmetric/aria_gcm.hpp, 428, 430
include/filevault/algorithms/symmetric/camellia_gcm.hpp, 430, 432
include/filevault/algorithms/symmetric/chacha20_poly1305.hpp, 432, 434
include/filevault/algorithms/symmetric/serpent_gcm.hpp, 434, 435
include/filevault/algorithms/symmetric/sm4_gcm.hpp, 436, 438
include/filevault/algorithms/symmetric/triple_des.hpp, 438, 440
include/filevault/algorithms/symmetric/twofish_gcm.hpp, 440, 442
include/filevault/archive Directory Reference, 30
include/filevault/archive/archive_format.hpp, 442, 443
include/filevault/cli Directory Reference, 33
include/filevault/cli/app.hpp, 444, 445
include/filevault/cli/command.hpp, 446, 447
include/filevault/cli/commands Directory Reference, 34
include/filevault/cli/commands/archive_cmd.hpp, 447, 448
include/filevault/cli/commands/benchmark_cmd.hpp, 449, 450
include/filevault/cli/commands/compress_cmd.hpp, 451,

452
include/filevault/cli/commands/config_cmd.hpp, 453
include/filevault/cli/commands/decompress_cmd.hpp, 454, 455
include/filevault/cli/commands/decrypt_cmd.hpp, 456
include/filevault/cli/commands/dump_cmd.hpp, 457, 458
include/filevault/cli/commands/encrypt_cmd.hpp, 459
include/filevault/cli/commands/hash_cmd.hpp, 460, 461
include/filevault/cli/commands/info_cmd.hpp, 462, 463
include/filevault/cli/commands/keygen_cmd.hpp, 464, 465
include/filevault/cli/commands/keyinfo_cmd.hpp, 466, 467
include/filevault/cli/commands/list_cmd.hpp, 467, 468
include/filevault/cli/commands/sign_cmd.hpp, 468, 469
include/filevault/cli/commands/stego_cmd.hpp, 470, 471
include/filevault/cli/commands/verify_cmd.hpp, 472
include/filevault/compression Directory Reference, 36
include/filevault/compression/compressor.hpp, 473, 474
include/filevault/core Directory Reference, 37
include/filevault/core/crypto_algorithm.hpp, 475, 476
include/filevault/core/crypto_engine.hpp, 477, 478
include/filevault/core/file_format.hpp, 478, 480
include/filevault/core/modes.hpp, 482, 483
include/filevault/core/result.hpp, 484, 485
include/filevault/core/streaming.hpp, 486, 487
include/filevault/core/types.hpp, 488, 490
include/filevault/format Directory Reference, 38
include/filevault/format/file_header.hpp, 493, 494
include/filevault/steganography Directory Reference, 41
include/filevault/steganography/lsb.hpp, 495, 496
include/filevault/utils Directory Reference, 44
include/filevault/utils/config.hpp, 497, 498
include/filevault/utils/console.hpp, 499
include/filevault/utils/crypto_utils.hpp, 500
include/filevault/utils/file_io.hpp, 501
include/filevault/utils/password.hpp, 502, 503
include/filevault/utils/progress.hpp, 503, 504
include/filevault/utils/table_formatter.hpp, 505, 506
include_filename
    filevault::core::HashConfig, 256
include_metadata
    filevault::core::EncryptionConfig, 225
info
    filevault::utils::Console, 178
InfoCommand
    filevault::cli::InfoCommand, 273
initialize
    filevault::cli::Application, 104
    filevault::core::CryptoEngine, 185
input_file_
    filevault::cli::commands::StegoCommand, 354
    filevault::cli::CompressCommand, 161
    filevault::cli::DecompressCommand, 194
    filevault::cli::DecryptCommand, 198
    filevault::cli::EncryptCommand, 223
    filevault::cli::HashCommand, 255
    filevault::cli::InfoCommand, 276
input_files_
    filevault::cli::commands::ArchiveCommand, 112
invert_matrix
    filevault::algorithms::classical::HillCipher, 261
is_authenticated
    filevault::algorithms::asymmetric::RSA, 330
    filevault::algorithms::symmetric::AES_CFB, 74
    filevault::algorithms::symmetric::AES_ECB, 85
    filevault::algorithms::symmetric::AES_OFB, 95
    filevault::algorithms::symmetric::AES_XTS, 101
is_common_password
    filevault::core::PasswordAnalysis, 311
    filevault::utils::Password, 309
is_compressed
    filevault::format::FileHeader, 238
is_legacy_format
    filevault::core::FileFormatHandler, 230
is_secure_algorithm
    filevault::cli::HashCommand, 253
is_suitable_for
    filevault::algorithms::asymmetric::ECCHybrid, 210
    filevault::algorithms::asymmetric::RSA, 330
    filevault::algorithms::classical::Caesar, 143
    filevault::algorithms::classical::HillCipher, 261
    filevault::algorithms::classical::Playfair, 319
    filevault::algorithms::classical::SubstitutionCipher, 364
    filevault::algorithms::classical::Vigenere, 390
    filevault::algorithms::pqc::Kyber, 288
    filevault::algorithms::pqc::KyberHybrid, 293
    filevault::algorithms::symmetric::AES_CBC, 69
    filevault::algorithms::symmetric::AES_CFB, 74
    filevault::algorithms::symmetric::AES_CTR, 79
    filevault::algorithms::symmetric::AES_ECB, 85
    filevault::algorithms::symmetric::AES_GCM, 90
    filevault::algorithms::symmetric::AES_OFB, 95
    filevault::algorithms::symmetric::AES_XTS, 101
    filevault::algorithms::symmetric::ARIA_GCM, 121
    filevault::algorithms::symmetric::Camellia_GCM, 148
    filevault::algorithms::symmetric::ChaCha20Poly1305, 153
    filevault::algorithms::symmetric::Serpent_GCM, 337
    filevault::algorithms::symmetric::SM4_GCM, 347
    filevault::algorithms::symmetric::TripleDES, 374
    filevault::algorithms::symmetric::Twofish_GCM, 379
    filevault::core::ICryptoAlgorithm, 271
is_terminal
    filevault::utils, 63
is_valid
    filevault::core::FileHeader, 233
is_valid_key
    filevault::algorithms::classical::HillCipher, 262
    filevault::algorithms::classical::SubstitutionCipher, 364
ISATTY

progress.cpp, 545
iterations
filevault::core::Argon2Params, 117
filevault::core::PBKDF2Params, 313
iterations_
filevault::cli::BenchmarkCommand, 134
iv_size
filevault::algorithms::symmetric::AES_CBC, 69
filevault::algorithms::symmetric::AES_CFB, 74
filevault::algorithms::symmetric::AES_OFB, 95
filevault::algorithms::symmetric::TripleDES, 374

json_output_
filevault::cli::BenchmarkCommand, 134

kasiski_examination
filevault::algorithms::classical::Vigenere, 390
kdf
filevault::cli::InfoCommand::FileInfo, 247
filevault::core::EncryptionConfig, 226
filevault::core::FileHeader, 235
filevault::core::ModePreset, 304
filevault::core::StreamingConfig, 355
filevault::format::FileHeader, 238
kdf_
filevault::cli::commands::ArchiveCommand, 112
filevault::cli::EncryptCommand, 223
filevault::format::FileHeader, 245
kdf_iterations
filevault::core::EncryptionConfig, 226
filevault::core::ModePreset, 304
kdf_memory_kb
filevault::core::EncryptionConfig, 226
filevault::core::ModePreset, 304
kdf_name
filevault::core::CryptoEngine, 186
kdf_only_
filevault::cli::BenchmarkCommand, 135
kdf_parallelism
filevault::core::EncryptionConfig, 226
filevault::core::ModePreset, 304
kdf_params
filevault::core::FileHeader, 235
KDFID
filevault::core, 59
KDFType
filevault::core, 59
key_
filevault::cli::ConfigCommand, 175
key_bits_
filevault::algorithms::asymmetric::RSA, 332
filevault::algorithms::symmetric::AES_CBC, 70
filevault::algorithms::symmetric::AES_CFB, 75
filevault::algorithms::symmetric::AES_CTR, 80
filevault::algorithms::symmetric::AES_ECB, 86
filevault::algorithms::symmetric::AES_GCM, 91
filevault::algorithms::symmetric::AES_OFB, 96
filevault::algorithms::symmetric::AES_XTS, 103
filevault::algorithms::symmetric::ARIA_GCM, 123
filevault::algorithms::symmetric::Camellia_GCM, 150
filevault::algorithms::symmetric::Twofish_GCM, 382
key_path_
filevault::cli::commands::KeyInfoCommand, 283
key_size
filevault::algorithms::asymmetric::ECCHybrid, 210
filevault::algorithms::asymmetric::ECDH, 213
filevault::algorithms::asymmetric::ECDSA, 216
filevault::algorithms::asymmetric::RSA, 330
filevault::algorithms::classical::Caesar, 143
filevault::algorithms::classical::HillCipher, 262
filevault::algorithms::classical::Playfair, 319
filevault::algorithms::classical::SubstitutionCipher, 365
filevault::algorithms::classical::Vigenere, 390
filevault::algorithms::pqc::Kyber, 288
filevault::algorithms::pqc::KyberHybrid, 293
filevault::algorithms::symmetric::AES_CBC, 69
filevault::algorithms::symmetric::AES_CFB, 74
filevault::algorithms::symmetric::AES_CTR, 79
filevault::algorithms::symmetric::AES_ECB, 85
filevault::algorithms::symmetric::AES_GCM, 90
filevault::algorithms::symmetric::AES_OFB, 95
filevault::algorithms::symmetric::AES_XTS, 101
filevault::algorithms::symmetric::ARIA_GCM, 121
filevault::algorithms::symmetric::Camellia_GCM, 148
filevault::algorithms::symmetric::ChaCha20Poly1305, 153
filevault::algorithms::symmetric::Serpent_GCM, 337
filevault::algorithms::symmetric::SM4_GCM, 347
filevault::algorithms::symmetric::TripleDES, 374
filevault::algorithms::symmetric::Twofish_GCM, 380
filevault::core::ICryptoAlgorithm, 271
keygen_ms
filevault::cli::AsymmetricBenchmarkResult, 123
filevault::cli::PQCBenchmarkResult, 320
filevault::cli::SignatureBenchmarkResult, 339
KeygenCommand
filevault::cli::KeygenCommand, 278
KeyInfoCommand
filevault::cli::commands::KeyInfoCommand, 282
keyword_
filevault::algorithms::classical::Vigenere, 391
Kyber
filevault::algorithms::pqc::Kyber, 286
Kyber1024
filevault::algorithms::pqc::Kyber, 286
Kyber512
filevault::algorithms::pqc::Kyber, 286
Kyber768
filevault::algorithms::pqc::Kyber, 286
kyber_
filevault::algorithms::pqc::KyberHybrid, 294

KYBER_1024
filevault::core, 58
KYBER_1024_HYBRID
filevault::core, 58
KYBER_512
filevault::core, 58
KYBER_512_HYBRID
filevault::core, 58
KYBER_768
filevault::core, 58
KYBER_768_HYBRID
filevault::core, 58
KyberHybrid
filevault::algorithms::pqc::KyberHybrid, 291

length
filevault::core::PasswordAnalysis, 311
LENGTH_HEADER_SIZE
filevault::steganography::LSBSteganography, 300
level
filevault::core::EncryptionConfig, 226
filevault::core::StreamingConfig, 355
level_
filevault::cli::CompressCommand, 162
list_files
filevault::archive::ArchiveFormat, 114
ListCommand
filevault::cli::ListCommand, 295
load
filevault::utils::Config, 167
log_level_
filevault::cli::Application, 105
lsb.cpp
STB_IMAGE_IMPLEMENTATION, 541
STB_IMAGE_WRITE_IMPLEMENTATION, 541
STBI_FAILURE_USERMSG, 541
LZMA
filevault::core, 58, 59

MAGIC
filevault::archive::ArchiveFormat, 116
filevault::format::FileHeader, 245
magic
filevault::core::FileHeader, 235
main
main.cpp, 540
main.cpp
main, 540
mark_as_completed
filevault::utils::BlockProgressBar, 136
filevault::utils::ProgressBar, 322
Matrix2x2
filevault::algorithms::classical::HillCipher, 258
matrix_
filevault::algorithms::classical::Playfair, 320
max_bytes_
filevault::cli::commands::DumpCommand, 206
max_plaintext_size
filevault::algorithms::asymmetric::RSA, 330
max_progress_
filevault::utils::ProgressBar, 323
MD5
filevault::core, 59
MEDIUM
filevault::core, 60
memory_kb
filevault::core::Argon2Params, 117
MIN_HEADER_SIZE
filevault::format::FileHeader, 245
mingw_data
filevault::utils, 63
mod_inverse
filevault::algorithms::classical::HillCipher, 262
mode
filevault::core::EncryptionConfig, 226
filevault::core::ModePreset, 304
mode_
filevault::cli::EncryptCommand, 223
modified_time
filevault::archive::FileEntry, 227

n
filevault::core::ScryptParams, 333
name
filevault::algorithms::asymmetric::ECCHybrid, 210
filevault::algorithms::asymmetric::ECDH, 214
filevault::algorithms::asymmetric::ECDSA, 216
filevault::algorithms::asymmetric::RSA, 330
filevault::algorithms::classical::Caesar, 143
filevault::algorithms::classical::HillCipher, 263
filevault::algorithms::classical::Playfair, 319
filevault::algorithms::classical::SubstitutionCipher, 365
filevault::algorithms::classical::Vigenere, 390
filevault::algorithms::pqc::Dilithium, 201
filevault::algorithms::pqc::Kyber, 288
filevault::algorithms::pqc::KyberHybrid, 293
filevault::algorithms::symmetric::AES_CBC, 70
filevault::algorithms::symmetric::AES_CFB, 75
filevault::algorithms::symmetric::AES_CTR, 79
filevault::algorithms::symmetric::AES_ECB, 86
filevault::algorithms::symmetric::AES_GCM, 90
filevault::algorithms::symmetric::AES_OFB, 96
filevault::algorithms::symmetric::AES_XTS, 102
filevault::algorithms::symmetric::ARIA_GCM, 121
filevault::algorithms::symmetric::Camellia_GCM, 148
filevault::algorithms::symmetric::ChaCha20Poly1305, 154
filevault::algorithms::symmetric::Serpent_GCM, 338
filevault::algorithms::symmetric::SM4_GCM, 347
filevault::algorithms::symmetric::TripleDES, 375
filevault::algorithms::symmetric::Twofish_GCM, 380
filevault::cli::BenchmarkCommand, 133
filevault::cli::commands::ArchiveCommand, 110
filevault::cli::commands::DumpCommand, 205

filevault::cli::commands::KeyInfoCommand, 283
filevault::cli::commands::SignCommand, 342
filevault::cli::commands::StegoCommand, 353
filevault::cli::commands::VerifyCommand, 384
filevault::cli::CompressCommand, 160
filevault::cli::ConfigCommand, 173
filevault::cli::DecompressCommand, 193
filevault::cli::DecryptCommand, 197
filevault::cli::EncryptCommand, 221
filevault::cli::HashCommand, 254
filevault::cli::ICommand, 266
filevault::cli::InfoCommand, 275
filevault::cli::KeygenCommand, 279
filevault::cli::ListCommand, 296
filevault::compression::Bzip2Compressor, 138
filevault::compression::ICompressor, 268
filevault::compression::LzmaCompressor, 301
filevault::compression::ZlibCompressor, 393
filevault::core::ICryptoAlgorithm, 271
filevault::core::ModePreset, 303
name_
filevault::cli::commands::DumpCommand, 206
filevault::cli::commands::KeyInfoCommand, 283
filevault::cli::commands::SignCommand, 343
filevault::cli::commands::VerifyCommand, 385
no_filename_
filevault::cli::HashCommand, 255
no_progress_
filevault::cli::DecryptCommand, 199
filevault::cli::EncryptCommand, 223
nonce
filevault::core::CryptoResult, 188
filevault::core::EncryptionConfig, 226
filevault::core::FileHeader, 235
filevault::format::FileHeader, 239
nonce_
filevault::format::FileHeader, 245
nonce_size
filevault::algorithms::symmetric::AES_CTR, 80
filevault::algorithms::symmetric::AES_GCM, 91
filevault::algorithms::symmetric::ARIA_GCM, 122
filevault::algorithms::symmetric::Camellia_GCM, 149
filevault::algorithms::symmetric::ChaCha20Poly1305, 154
filevault::algorithms::symmetric::SM4_GCM, 348
filevault::algorithms::symmetric::Twofish_GCM, 380
filevault::cli::InfoCommand::FileInfo, 247
NONE
filevault::core, 58, 59

offset
filevault::archive::FileEntry, 227
ok
filevault::core::Result< T >, 324
filevault::core::Result< void >, 326
operation_
filevault::cli::commands::StegoCommand, 354
operator bool
filevault::core::Result< T >, 325
filevault::core::Result< void >, 326
operator=
filevault::core::CryptoEngine, 186
original_size
filevault::compression::CompressionResult, 162
filevault::core::CryptoResult, 188
filevault::format::FileHeader, 239
original_size_
filevault::format::FileHeader, 245
output_file_
filevault::cli::BenchmarkCommand, 135
filevault::cli::commands::ArchiveCommand, 112
filevault::cli::commands::StegoCommand, 354
filevault::cli::CompressCommand, 162
filevault::cli::DecompressCommand, 194
filevault::cli::DecryptCommand, 199
filevault::cli::EncryptCommand, 223
filevault::cli::HashCommand, 255
output_format_
filevault::cli::HashCommand, 255
output_path_
filevault::cli::commands::SignCommand, 343
output_prefix_
filevault::cli::KeygenCommand, 280

p
filevault::core::ScryptParams, 333
pair_key_path_
filevault::cli::commands::KeyInfoCommand, 284
parallelism
filevault::core::Argon2Params, 117
PARANOID
filevault::core, 60
parse_algorithm
filevault::compression::CompressionService, 164
filevault::core::CryptoEngine, 186
parse_file
filevault::cli::InfoCommand, 275
parse_kdf
filevault::core::CryptoEngine, 187
parse_key
filevault::algorithms::classical::HillCipher, 263
filevault::algorithms::classical::SubstitutionCipher, 365
parse_mode
filevault::core::ModePreset, 303
parse_security_level
filevault::core::CryptoEngine, 187
password_
filevault::cli::commands::ArchiveCommand, 112
filevault::cli::DecryptCommand, 199
filevault::cli::EncryptCommand, 223
PasswordStrength
filevault::core, 60
PBKDF2_SHA256
filevault::core, 59, 60
PBKDF2_SHA512

filevault::core, 59, 60
permissions
filevault::archive::FileEntry, 227
PLAYFAIR
filevault::core, 56, 58
Playfair
filevault::algorithms::classical::Playfair, 315
pqc_only_
filevault::cli::BenchmarkCommand, 135
print
filevault::utils::TableFormatter, 370
print_algorithm_list
filevault::utils::TableBuilder, 366
print_benchmark_results
filevault::utils::TableBuilder, 366
print_encryption_config
filevault::utils::TableBuilder, 367
print_file_summary
filevault::utils::TableBuilder, 367
private_key
filevault::algorithms::asymmetric::ECCKeyPair, 211
filevault::algorithms::asymmetric::RSAKeyPair, 332
filevault::algorithms::pqc::PQKeyPair, 321
private_key_path_
filevault::cli::commands::SignCommand, 343
private_key_size
filevault::algorithms::pqc::Dilithium, 201
filevault::algorithms::pqc::Kyber, 289
process_gcm
filevault::algorithms::symmetric::Serpent_GCM, 338
filevault::algorithms::symmetric::Twofish_GCM, 380
processing_time_ms
filevault::compression::CompressionResult, 163
filevault::core::CryptoResult, 188
filevault::core::StreamingResult, 360
PROFESSIONAL
filevault::core, 60
progress.cpp
FILENO, 545
ISATTY, 545
progress_callback
filevault::core::StreamingConfig, 355
ProgressBar
filevault::utils::ProgressBar, 322
public_key
filevault::algorithms::asymmetric::ECCKeyPair, 211
filevault::algorithms::asymmetric::RSAKeyPair, 332
filevault::algorithms::pqc::PQKeyPair, 321
public_key_path_
filevault::cli::commands::VerifyCommand, 385
public_key_size
filevault::algorithms::pqc::Dilithium, 201
filevault::algorithms::pqc::Kyber, 289

r
filevault::core::ScryptParams, 333
read_file
filevault::core::FileFormatHandler, 230
filevault::utils::FileIO, 248
read_legacy_file
filevault::core::FileFormatHandler, 231
read_secure
filevault::utils::Password, 309
read_stream_header
filevault::core::StreamingCrypto, 358
read_string
filevault::archive::ArchiveFormat, 114
read_uint16
filevault::format::FileHeader, 239
read_uint32
filevault::archive::ArchiveFormat, 115
filevault::format::FileHeader, 239
read_uint64
filevault::archive::ArchiveFormat, 115
filevault::format::FileHeader, 240
README.md, 506
register_algorithm
filevault::core::CryptoEngine, 187
register_commands
filevault::cli::Application, 104
remove_pkcs7_padding
filevault::algorithms::symmetric, 51
requires_padding
filevault::algorithms::asymmetric::RSA, 331
filevault::algorithms::symmetric::AES_CFB, 75
filevault::algorithms::symmetric::AES_ECB, 86
filevault::algorithms::symmetric::AES_OFB, 96
filevault::algorithms::symmetric::AES_XTS, 102
reserved
filevault::core::FileHeader, 235
reserved_
filevault::format::FileHeader, 245
reset
filevault::utils::Config, 168
reset_config
filevault::cli::ConfigCommand, 173
RSA
filevault::algorithms::asymmetric::RSA, 328
RSA_2048
filevault::core, 56, 58
RSA_3072
filevault::core, 56, 58
RSA_4096
filevault::core, 57, 58
run
filevault::cli::Application, 104

salt
filevault::core::CryptoResult, 188
filevault::core::EncryptionConfig, 226
filevault::core::FileHeader, 235
filevault::format::FileHeader, 240
salt_
filevault::format::FileHeader, 245
salt_size
filevault::cli::InfoCommand::FileInfo, 247

save
filevault::utils::Config, 168
save_json_output
filevault::cli::BenchmarkCommand, 133
save_log_output
filevault::cli::BenchmarkCommand, 133
score
filevault::core::PasswordAnalysis, 312
SCRYPT
filevault::core, 59, 60
SECP256R1
filevault::algorithms::asymmetric, 48
SECP384R1
filevault::algorithms::asymmetric, 48
SECP521R1
filevault::algorithms::asymmetric, 48
security_level
filevault::core::ModePreset, 304
filevault::format::FileHeader, 240
security_level_
filevault::cli::commands::ArchiveCommand, 112
filevault::cli::EncryptCommand, 223
filevault::format::FileHeader, 245
security_level_name
filevault::core::CryptoEngine, 187
SecurityLevel
filevault::core, 60
separator
filevault::utils::Console, 179
serialize
filevault::archive::FileEntry, 227
filevault::core::Argon2Params, 116
filevault::core::FileHeader, 234
filevault::core::PBKDF2Params, 312
filevault::core::ScryptParams, 333
filevault::format::FileHeader, 240
SERPENT_256_GCM
filevault::core, 56, 57
Serpent_GCM
filevault::algorithms::symmetric::Serpent_GCM, 335
set
filevault::utils::Config, 169
set_algorithm
filevault::format::FileHeader, 241
set_border_style
filevault::utils::TableFormatter, 370
set_column_alignment
filevault::utils::TableFormatter, 370
set_column_format
filevault::utils::TableFormatter, 370
set_compressed
filevault::format::FileHeader, 241
set_compression_level
filevault::utils::Config, 169
set_default_algorithm
filevault::utils::Config, 169
set_default_compression
filevault::utils::Config, 169
set_default_kdf
filevault::utils::Config, 169
set_default_mode
filevault::utils::Config, 169
set_encrypted_size
filevault::format::FileHeader, 241
set_kdf
filevault::format::FileHeader, 241
set_nonce
filevault::format::FileHeader, 241
set_option_text
filevault::utils::BlockProgressBar, 136
set_original_size
filevault::format::FileHeader, 242
set_postfix
filevault::utils::ProgressBar, 322
set_progress
filevault::utils::BlockProgressBar, 136
filevault::utils::ProgressBar, 322
set_salt
filevault::format::FileHeader, 242
set_security_level
filevault::format::FileHeader, 242
set_show_progress
filevault::utils::Config, 169
set_tag
filevault::format::FileHeader, 242
set_timestamp
filevault::format::FileHeader, 243
set_value
filevault::cli::ConfigCommand, 174
set_verbose
filevault::utils::Config, 169
setup
filevault::cli::BenchmarkCommand, 133
filevault::cli::commands::ArchiveCommand, 111
filevault::cli::commands::DumpCommand, 205
filevault::cli::commands::KeyInfoCommand, 283
filevault::cli::commands::SignCommand, 342
filevault::cli::commands::StegoCommand, 353
filevault::cli::commands::VerifyCommand, 384
filevault::cli::CompressCommand, 161
filevault::cli::ConfigCommand, 174
filevault::cli::DecompressCommand, 193
filevault::cli::DecryptCommand, 198
filevault::cli::EncryptCommand, 222
filevault::cli::HashCommand, 254
filevault::cli::ICommand, 266
filevault::cli::InfoCommand, 276
filevault::cli::KeygenCommand, 279
filevault::cli::ListCommand, 296
setup_logging
filevault::cli::Application, 105
SHA1
filevault::core, 59
SHA224
filevault::core, 59

SHA256
filevault::core, 59
SHA384
filevault::core, 59
SHA3_224
filevault::core, 59
SHA3_256
filevault::core, 59
SHA3_384
filevault::core, 59
SHA3_512
filevault::core, 59
SHA512
filevault::core, 59
SHA512_256
filevault::core, 59
shared_secret
filevault::algorithms::asymmetric::ECDHResult, 214
shared_secret_size
filevault::algorithms::pqc::Kyber, 289
shift_
filevault::algorithms::classical::Caesar, 144
shift_char
filevault::algorithms::classical::Caesar, 144
should_use_streaming
filevault::core::StreamingCrypto, 359
show
filevault::utils::ProgressBar, 323
show_ascii_
filevault::cli::commands::DumpCommand, 206
show_config
filevault::cli::ConfigCommand, 174
show_offset_
filevault::cli::commands::DumpCommand, 206
show_progress
filevault::core::EncryptionConfig, 226
show_progress_
filevault::utils::Config, 170
show_public_
filevault::cli::commands::KeyInfoCommand, 284
sign
filevault::algorithms::asymmetric::ECDSA, 217
filevault::algorithms::asymmetric::RSA, 331
filevault::algorithms::pqc::Dilithium, 201
sign_ms
filevault::cli::SignatureBenchmarkResult, 340
signature
filevault::algorithms::asymmetric::ECDSASignResult, 218
signature_path_
filevault::cli::commands::VerifyCommand, 385
signature_size
filevault::algorithms::asymmetric::ECDSA, 217
filevault::algorithms::pqc::Dilithium, 202
SignCommand
filevault::cli::commands::SignCommand, 341
size
filevault::core::FileHeader, 234
SM4_GCM
filevault::algorithms::symmetric::SM4_GCM, 345
filevault::core, 56, 57
src Directory Reference, 41
src/algorithms Directory Reference, 29
src/algorithms/asymmetric Directory Reference, 31
src/algorithms/asymmetric/ecc.cpp, 506
src/algorithms/asymmetric/rsa.cpp, 507
src/algorithms/classical Directory Reference, 32
src/algorithms/classical/caesar.cpp, 508
src/algorithms/classical/classical_ciphers.cpp, 509
src/algorithms/classical/hill.cpp, 509
src/algorithms/classical/playfair.cpp, 510
src/algorithms/classical/substitution.cpp, 511
src/algorithms/classical/vigenere.cpp, 512
src/algorithms/pqc Directory Reference, 40
src/algorithms/pqc/post_quantum.cpp, 513
src/algorithms/symmetric Directory Reference, 43
src/algorithms/symmetric/aes_cbc.cpp, 514
src/algorithms/symmetric/aes_cfb.cpp, 515
src/algorithms/symmetric/aes_ctr.cpp, 515
src/algorithms/symmetric/aes_ecb.cpp, 516
src/algorithms/symmetric/aes_gcm.cpp, 517
src/algorithms/symmetric/aes_ofb.cpp, 517
src/algorithms/symmetric/aes_xts.cpp, 518
src/algorithms/symmetric/aria_gcm.cpp, 519
src/algorithms/symmetric/camellia_gcm.cpp, 520
src/algorithms/symmetric/chacha20_poly1305.cpp, 521
src/algorithms/symmetric/serpent_gcm.cpp, 522
src/algorithms/symmetric/sm4_gcm.cpp, 522
src/algorithms/symmetric/triple_des.cpp, 523
src/algorithms/symmetric/twofish_gcm.cpp, 524
src/archive Directory Reference, 30
src/archive/archive_format.cpp, 524
src/cli Directory Reference, 34
src/cli/app.cpp, 525
src/cli/commands Directory Reference, 35
src/cli/commands/archive_cmd.cpp, 526
src/cli/commands/benchmark_cmd.cpp, 526
src/cli/commands/compress_cmd.cpp, 527
src/cli/commands/config_cmd.cpp, 527
src/cli/commands/decompress_cmd.cpp, 528
src/cli/commands/decrypt_cmd.cpp, 528
src/cli/commands/dump_cmd.cpp, 529
src/cli/commands/encrypt_cmd.cpp, 529
src/cli/commands/hash_cmd.cpp, 530
src/cli/commands/info_cmd.cpp, 531
src/cli/commands/keygen_cmd.cpp, 531
src/cli/commands/keyinfo_cmd.cpp, 532
src/cli/commands/list_cmd.cpp, 532
src/cli/commands/sign_cmd.cpp, 533
src/cli/commands/stego_cmd.cpp, 533
src/cli/commands/verify_cmd.cpp, 534
src/compression Directory Reference, 36
src/compression/compressor.cpp, 535
src/core Directory Reference, 37
src/core/crypto_engine.cpp, 535

src/core/modes.cpp, 536
src/core/streaming.cpp, 537
src/core/types.cpp, 538
src/format Directory Reference, 39
src/format/file_format.cpp, 538
src/format/file_header.cpp, 539
src/main.cpp, 539
src/steganography Directory Reference, 42
src/steganography/lsb.cpp, 540
src/utils Directory Reference, 45
src/utils/config.cpp, 541
src/utils/console.cpp, 542
src/utils/crypto_utils.cpp, 542
src/utils/file_io.cpp, 543
src/utils/password.cpp, 544
src/utils/progress.cpp, 544
src/utils/table_formatter.cpp, 545
STANDARD
    filevault::core::presets, 62
STB_IMAGE_IMPLEMENTATION
    lsb.cpp, 541
STB_IMAGE_WRITE_IMPLEMENTATION
    lsb.cpp, 541
STBI_FAILURE_USERMSG
    lsb.cpp, 541
STREAM_MAGIC
    filevault::core, 61
STREAM_VERSION
    filevault::core, 61
StreamProgressCallback
    filevault::core, 55
strength
    filevault::core::PasswordAnalysis, 312
STRONG
    filevault::core, 60
STUDENT
    filevault::core, 60
subcommand_
    filevault::cli::CompressCommand, 162
    filevault::cli::ConfigCommand, 175
    filevault::cli::DecompressCommand, 194
SUBSTITUTION
    filevault::core, 56, 58
SubstitutionMap
    filevault::algorithms::classical::SubstitutionCipher, 362
success
    filevault::algorithms::asymmetric::ECDHResult, 214
    filevault::algorithms::asymmetric::ECDSASignResult, 218
    filevault::cli::AsymmetricBenchmarkResult, 123
    filevault::cli::BenchmarkResult, 135
    filevault::cli::PQCBenchmarkResult, 321
    filevault::cli::SignatureBenchmarkResult, 340
    filevault::compression::CompressionResult, 163
    filevault::core::CryptoResult, 188
    filevault::core::Result< T >, 325
    filevault::core::Result< void >, 326
    filevault::core::StreamingResult, 360
    filevault::utils::Console, 180
suggestions
    filevault::core::PasswordAnalysis, 312
symmetric_only_
    filevault::cli::BenchmarkCommand, 135

table_
    filevault::utils::TableFormatter, 370
TableFormatter
    filevault::utils::TableFormatter, 368
tag
    filevault::core::CryptoResult, 189
    filevault::core::EncryptionConfig, 226
    filevault::format::FileHeader, 243
tag_
    filevault::format::FileHeader, 245
tag_size
    filevault::algorithms::symmetric::AES_GCM, 91
    filevault::algorithms::symmetric::ARIA_GCM, 122
    filevault::algorithms::symmetric::Camellia_GCM, 149
    filevault::algorithms::symmetric::ChaCha20Poly1305, 154
    filevault::algorithms::symmetric::SM4_GCM, 348
    filevault::algorithms::symmetric::Twofish_GCM, 381
    filevault::cli::InfoCommand::FileInfo, 247
throughput_mbps
    filevault::core::StreamingResult, 360
tick
    filevault::utils::ProgressBar, 323
timestamp
    filevault::format::FileHeader, 243
timestamp_
    filevault::format::FileHeader, 245
to_algorithm_id
    filevault::core::FileFormatHandler, 231
to_compression_id
    filevault::core::FileFormatHandler, 231
to_json
    filevault::utils::Config, 169
to_kdf_id
    filevault::core::FileFormatHandler, 231
to_string
    filevault::utils::TableFormatter, 370
total_bytes
    filevault::core::ChunkInfo, 155
total_chunks
    filevault::core::ChunkInfo, 155
total_size
    filevault::format::FileHeader, 243
TRIPLE_DES_CBC
    filevault::core, 56, 58
TripleDES
    filevault::algorithms::symmetric::TripleDES, 372
tweak_size
    filevault::algorithms::symmetric::AES_XTS, 102

TWOFISH_128_GCM
filevault::core, 56, 57
TWOFISH_192_GCM
filevault::core, 56, 57
TWOFISH_256_GCM
filevault::core, 56, 57
Twofish_GCM
filevault::algorithms::symmetric::Twofish_GCM, 377
type
filevault::algorithms::asymmetric::ECCHybrid, 210
filevault::algorithms::asymmetric::RSA, 331
filevault::algorithms::classical::Caesar, 144
filevault::algorithms::classical::HillCipher, 263
filevault::algorithms::classical::Playfair, 320
filevault::algorithms::classical::SubstitutionCipher, 365
filevault::algorithms::classical::Vigenere, 390, 391
filevault::algorithms::pqc::Kyber, 289
filevault::algorithms::pqc::KyberHybrid, 293
filevault::algorithms::symmetric::AES_CBC, 70
filevault::algorithms::symmetric::AES_CFB, 75
filevault::algorithms::symmetric::AES_CTR, 80
filevault::algorithms::symmetric::AES_ECB, 86
filevault::algorithms::symmetric::AES_GCM, 91
filevault::algorithms::symmetric::AES_OFB, 96
filevault::algorithms::symmetric::AES_XTS, 102
filevault::algorithms::symmetric::ARIA_GCM, 122
filevault::algorithms::symmetric::Camellia_GCM, 149
filevault::algorithms::symmetric::ChaCha20Poly1305, 155
filevault::algorithms::symmetric::Serpent_GCM, 339
filevault::algorithms::symmetric::SM4_GCM, 348
filevault::algorithms::symmetric::TripleDES, 375
filevault::algorithms::symmetric::Twofish_GCM, 381
filevault::core::ICryptoAlgorithm, 272
type_
filevault::algorithms::asymmetric::ECCHybrid, 211
filevault::algorithms::asymmetric::RSA, 332
filevault::algorithms::pqc::Kyber, 289
filevault::algorithms::symmetric::AES_CBC, 70
filevault::algorithms::symmetric::AES_CFB, 75
filevault::algorithms::symmetric::AES_CTR, 80
filevault::algorithms::symmetric::AES_ECB, 86
filevault::algorithms::symmetric::AES_GCM, 91
filevault::algorithms::symmetric::AES_OFB, 96
filevault::algorithms::symmetric::AES_XTS, 103
filevault::algorithms::symmetric::ARIA_GCM, 123
filevault::algorithms::symmetric::Camellia_GCM, 150
filevault::algorithms::symmetric::Twofish_GCM, 382

UNKNOWN
filevault::core, 56
unwrap
filevault::core::Result< T >, 325
filevault::core::Result< void >, 326
uppercase
filevault::core::HashConfig, 256
uppercase_
filevault::cli::HashCommand, 256
UserMode
filevault::core, 60

validate
filevault::format::FileHeader, 243
value
filevault::core::Result< T >, 325
filevault::core::Result< void >, 326
value_
filevault::cli::ConfigCommand, 175
Variant
filevault::algorithms::pqc::Dilithium, 200
filevault::algorithms::pqc::Kyber, 286
variant_
filevault::algorithms::pqc::Dilithium, 203
filevault::algorithms::pqc::Kyber, 289
verbose
filevault::core::EncryptionConfig, 226
verbose_
filevault::cli::Application, 105
filevault::cli::commands::ArchiveCommand, 112
filevault::cli::commands::StegoCommand, 354
filevault::cli::CompressCommand, 162
filevault::cli::DecompressCommand, 194
filevault::cli::DecryptCommand, 199
filevault::cli::EncryptCommand, 223
filevault::cli::HashCommand, 256
filevault::cli::InfoCommand, 276
filevault::cli::KeygenCommand, 280
filevault::utils::Config, 170
verify
filevault::algorithms::asymmetric::ECDSA, 217
filevault::algorithms::asymmetric::RSA, 331
filevault::algorithms::pqc::Dilithium, 202
verify_hash_
filevault::cli::HashCommand, 256
verify_mode
filevault::cli::HashCommand, 254
filevault::core::HashConfig, 257
verify_ms
filevault::cli::SignatureBenchmarkResult, 340
VerifyCommand
filevault::cli::commands::VerifyCommand, 383
VERSION
filevault::archive::ArchiveFormat, 116
version
filevault::cli::InfoCommand::FileInfo, 247
VERSION_MAJOR
filevault::format::FileHeader, 245
version_major
filevault::core::FileHeader, 235
VERSION_MINOR
filevault::format::FileHeader, 246

version_minor
filevault::core::FileHeader, 235
VERY_STRONG
filevault::core, 60
VERY_WEAK
filevault::core, 60
VIGENERE
filevault::core, 56, 58
Vigenere
filevault::algorithms::classical::Vigenere, 387, 388

warning
filevault::utils::Console, 181
warnings
filevault::core::PasswordAnalysis, 312
WEAK
filevault::core, 60
write_file
filevault::core::FileFormatHandler, 232
filevault::utils::FileIO, 249
write_stream_header
filevault::core::StreamingCrypto, 359
write_uint16
filevault::format::FileHeader, 243
write_uint32
filevault::archive::ArchiveFormat, 115
filevault::format::FileHeader, 244
write_uint64
filevault::archive::ArchiveFormat, 115
filevault::format::FileHeader, 244

X25519
filevault::algorithms::asymmetric, 48

ZLIB
filevault::core, 58, 59