Version 1.0, 2016-06-10

AxCrypt Version 2 Algorithms and File Format

Introduction

This document provides information on the file format used by the AxCrypt family of software, which provides confidentiality and integrity of stored data using encryption.

General Terms

- AxCrypt - The name of the original software version 1.x as well as a general term for the whole family of software which is compatible with AxCrypt-encrypted files.
- AxCrypt.NET or AxCrypt 2.x - The version 2.x level software, which is a complete rewrite of the original software, replacing both AxCrypt 1.x and AxCrypt2Go 1.x.
- AxDecrypt 1.x - A stand-alone utility which can only decrypt, and also implements self-decryption capabilities. This is obsoleted by version 2.x, as self-decrypting files are too restricted in use due to restrictions in transmitting executable files and version 2.x offers install-free full-functionality.
- AxCrypt2Go 1.x - A stand-alone utility which can decrypt and encrypt, but does not integrate with application launching.

Technical Terms

- Password - A string of characters entered by a user, which indirectly provides access to encrypted data. The password is never directly used to encrypt data. A fixed-length binary Key Encrypting Key is derived from the password using a key derivation algorithm.
- Key Encrypting Key - A key used to encrypt a master key, abbreviated KEK. The KEK is typically derived from a variable length password.
- Master Encryption Key - A randomly generated key, which is used to encrypt headers, calculate an HMAC and to actually encrypt data. Abbreviated MEK. The MEK is protected by a Key Wrap, using the KEK as the key.
- HMAC - Message authentication code as defined by RFC 2104.
- AxCrypt Software Version - the major version of the software suite. 1.x has been actively developed approximately 2000 - 2013, it is obsoleted by 2.x, released in April 2016.
- AxCrypt File Format Version - identifies the format of the encrypted data. AxCrypt Software Version 1.x supports File Format Version <= 3, while Software Version 2.x supports at least File Format Version <= 4.
- Key Stream Index - Encryption is performed by XOR:ing the plain text with a key stream. The key stream is generated either directly from a stream cipher, or by operating a block cipher in CTR mode. A Key Stream Index identifies a point in this stream.
- Number radix in this document is 10, unless specifically noted otherwise.

General functions

The AxCrypt file format provides support for storing data and meta data in a way to ensure both confidentiality and integrity. It also provides support for compression according to RFC 1950 ZLIB and RFC 1951 DEFLATE .

AxCrypt 2.x

The AxCrypt Software Version 2.x (File Format Version >= 4) file format is designed to be interpretable by AxCrypt 1.x software in the sense that it will detect in a controlled manner that the file is produced by a newer version of the software. This leads to the need for a few dummy blocks.

The overall-format of an AxCrypt 2.x file is:

GUID | Headers | Encrypted Data blocks | Headers | Trailers | End Of Stream

The GUID is c0b9072e4f93f146a015792ca1d9e821 .

The structure of Headers and Trailers are identical, the difference is that the Trailers contain information that is not known until all data has been encrypted, which includes data lengths and HMAC. The format is intended to be robust, and streamable. It is possible to encrypt a forward-only readable data stream to a forward-only writable data stream without whole-stream buffering, or backtracking in the written data stream. This is not possible with AxCrypt 1.x.

The headers, which include the Master Encryption Key data as well as salts and IV's are repeated after the Encrypted Data blocks for robustness.

Algorithms and modes of operation

Symmetric Bulk Encryption Algorithms

The file format is designed to accommodate block ciphers with block sizes <= 256 bits, and both stream and block ciphers with key sizes <= 512 bits. The file format does not contain any indication of what cipher is used, so certain fields are fixed at lengths to handle these maximum parameters.

Asymmetric Key Encryption

The Master Encryption Key is optionally also encrypted with RSA 4096-bit keys and OEAP padding. The purpose of this is to facilitate effective password changing for a single user, and to enable information sharing without sharing passwords with other users.
The private key is protected by serializing to PEM format, and encrypting the result as an AxCrypt-file.

Bit reduction

In many situations more bits than needed are made available, for example by the requirement to support up to 256-bit blocks and up to 512-bit keys in an indistinguishable manner. This reduction is made by initializing 'short' to all zero and then setting short[i modulo short-size] = short[i modulo short-size] XOR long[i] for i = 0..long-size.

Truncation

In situations where bit reduction is inappropriate, truncation is used and is always done by taking the leftmost, first or lowest-addressed bytes in the array or stream containing more data than needed.

Endianess

Integers and other data where endianness is relevant, is stored in little endian order unless otherwise specifically noted as this is assumed to be the native byte order in most cases.

Password derivation algorithm

The password is UTF-8 encoded without BOM. It is then passed to a PBKDF2-HMAC-SHA512 key derivation function. 64 bytes of output are produced and then reduced to the target key size. The salt is 32 bytes, and the iterations are set to a fixed value of 1000.
The resulting data of appropriate key-size is used as the key for the key-wrapping algorithm either to protect the master encryption key or to produce a symmetrical key thumbprint, and this is the only use of the derived key. Since it is effectively equivalent to the actual password, and is only used as input to a heavily iterated key-wrap, the brute-force protection implied by iteration is thus only present in the key-wrap. For purposes of cryptanalysis the derivation algorithm should not be considered to be part of the security, except to ensure that it is not likely to produce collisions.

Master Key Wrap algorithm

The master encryption key is wrapped using a generalized version of the NIST AES Key Wrap Algorithm.
There are three generalizations applied.

1. Use of a different block cipher algorithm than AES. This is done by a simple substitution of algorithm in the key wrapping specification.
2. Using a different block size. This is done by redefining the key-wrap block size in the specification from 64 bits to ½ the algorithm block size thus allowing for other block sizes than 128.
3. Use of stream cipher. This is done by using the stream cipher as a block cipher, with the block size equal to the key length. The wrap salt is used as IV if required for the stream cipher.

Confidentiality Mode of Operation

All encryption and decryption of data or headers using block ciphers is made using Counter Mode of Operation, CTR as described in "NIST Special Publication 800-38A". The counter is an unsigned 64-bit value initialized to zero. The counter is then combined with an appropriately

wide IV with XOR. The counter is in this case represented in big-endian order. If the IV is wider than 64 bits, the counter is XOR:ed with the 8 highest addressed bytes in the IV, i.e. the counter is aligned to the right with respect to the IV.
Incrementing and adding to the counter is done modulo 2^64, i.e. with wrap-around if required. Key stream indexes as used in the rest of the document are used to index the key stream. The actual counter is calculated by taking the byte index modulo the block size, and then indexing that key-stream block by the remainder and continuing from there.

Integrity Mode of Operation

The integrity of the data is provided by way of an HMAC-SHA512, applied after encryption. The HMAC is the last trailer block emitted, and thus will validate all bytes up to the first byte of the last trailer block.

Key material generation

When calculating the HMAC-SHA512 used for integrity validation, a key is required. To generate this, the cipher stream is indexed by a constant value of zero, and a sufficient amount of key material is generated. The various encrypted headers are also encrypted by providing them with constant byte indexes in the key stream to use as a starting point.

Rationales

Key size <= 512

There are no current or known future candidates of symmetric algorithms requiring more than 512 bits to represent a key. The highest current value is 448 which is used by Blowfish. Provided the algorithm is strong, there is currently no perceived need for an algorithm to require more than 256 bits, so 512 should provide a sufficient safety margin.

Block size <= 256

The most common block size for block ciphers is 128, there are a few using either 64 or 256 today. There seems to be no trend towards more than 256 bits block size. A stream cipher such as Salsa20 does provide a key stream in blocks of 512 bits, but this is trivially reduced to 256 bits by truncation or partitioning of the output when shorter blocks than 512 bits needs to be handled.

SHA512

A keyed hash function is needed for both PBKDF2 and for integrity validation, and it is suitable to use the same function in both places. Since the effective key space of PBKDF2 is limited by the underlying hash function, and keys up to 512 bits are supported the hash function also needs to be at least this wide. Therefore SHA512.

CTR Mode of Operation

CTR mode has many advantages, but the primary advantage here is that makes a block cipher behave very much as a stream cipher. By supporting this as the primary mode of operation for block ciphers, support of a pure stream cipher becomes much easier to fit in the same model.

A stream cipher has other properties that are desirable, among the more important there is no need for padding.
The use of CTR also makes random access to the stream possible (with the caveat that compressed data may not be easily randomly accessible, and a future stream cipher implementation may not support it either).

50ms per Key Wrap

When using iteration to increase the work factor for an attacker, a fixed count that is determined to suffice at the time of writing can be used. The problem with this is that according to Moore's law this number should probably increase by a factor of two every 12-18 months. A better approach is thus to dynamically determine the count by using the current hardware and fixing the amount of real time that the iteration is allowed to consume.
A reasonable compromise has been arrived at by targeting 50ms per key-wrap. This should let users access data without perceptible delay in most situations, while providing an effective guard against brute-force attacks. Since the number of iterations used is always recorded along with the resulting wrap, this parameter can be adjusted without compromising compatibility at any time, should for example mobile platforms show to be too slow to work comfortably at this level.

Header and Header Blocks

A header block follows the following general format:

```
  0   1   2   3   4   5 ......... Length-1
+---+---+---+---+---+=====================+
|LSB  Length MSB|HBT| Length-5 bytes data |
+---+---+---+---+---+=====================+
```

The Length is the total length of the data block, including the length itself. The length is stored as a 32-bit unsigned integer in little endian order, i.e. with the least significant byte (LSB) first and the most significant byte (MSB) last.

The Header Block Type (HBT) is a one-byte integer specifying what kind of data is stored in the block.

The following header block types are recognized by AxCrypt 2.x:

2 - Header. Preamble, Must be first.
3 - Header/Trailer. File and Program Version, Version information etc.
4 - Reserved - not used.
5 - Reserved - not used.
6 - Reserved - not used.

11 - Trailer. HMAC-SHA512/End of File. Key Stream Index 0 is used for the key.
13 - Header/Trailer. Symmetric Key Wrap - NIST Key Wrap Algorithm wrap of the Master Data Encrypting Key. This header/trailer may occur many times.
14 - Header/Trailer. RSA Key Encryption - The Master key encrypted using a RSA public key. This header/trailer may occur many times.
20 - Encrypted Data - one of potentially many data blocks. Starting Key Stream Index 1048576.

63 - Header. Start of Data/End of Headers marker. Identical to a AxCrypt 1.x data header with zero length.

64 - Encrypted,Start of headers containing encrypted header data. There is no actual header with this value.

65 - Reserved - not used.
66 - Reserved - not used.
67 - Reserved - not used.
68 - Header/Trailer. FileInfo, Time stamps of the original file. Optional. Key Stream Index 256.
69 - Header/Trailer. Compression, Indicates if the data is compressed. Key Stream Index 512.
70 - Header/Trailer. UTF-8 Encoded File Name. Optional. Key Stream Index 768.

101 - Trailer. Plain text lengths. Key Stream Index 2048.
102 - Header/Trailer. UTF-8 Encoded list of recipient e-mails, serialized as a JSON array. Key Stream Index 3072.
103 - Header/Trailer. Algorithm Verifier. Key Stream Index 4096.

All block types except those listed are undefined. Blocks with these types should be skipped and silently ignored, since they may have been defined by future versions of the software.

Header Block 2 - Preamble

The first header block must always be of type 2, Preamble.

The format is:

```
  0   1   2   3   4   5 .......... 20
+---+---+---+---+---+===============+
| 21  0   0   0 | 2 | 16 bytes zero |
+---+---+---+---+---+===============+
```

This block is for compatibility with AxCrypt 1.x.

Header Block 3 - File and Program Version

The version header block records information about the File Format version as well as the software generator version.

The format is:

```
  0   1   2   3   4
+---+---+---+---+---+---+---+---+---+---+
| 10  0   0   0 | 3 | a   b   x   y   z |
+---+---+---+---+---+---+---+---+---+---+
```

Where:

- a => File Major Version
- b => File Minor Version
- x => Program Major Version
- y => Program Minor Version
- z => Program Minor Version

The Program version is mostly for informative purposes. The File Minor Version can be used to control behavior for newer versions when reading older data.

The most important value is the File Major Version. A reading program that detects a higher File Major Version than it knows it can interpret, should stop reading and present a message to the effect that while the file is an AxCrypt file, the current program must be upgraded.

The most recent File Major.Minor Version at the time of writing is 4.0 and corresponds to AxCrypt 2.0, i.e. this specification.

Header Block 11 - HMAC-SHA512 End of File

This is the last block in the data stream, there are no more bytes following. Any kind of data, or no data may follow, including a new encrypted file stream.

The format is:

```
  0   1   2   3   4   5          68
+---+---+---+---+---+============+
| 69  0   0   0 | 11| HMAC-SHA512 |
+---+---+---+---+---+============+
```

The HMAC-SHA512 consists of the 64 bytes of checksum, validating all bytes from the starting GUID up to the last byte before the start of this header.
The key for the HMAC are the first 64 bytes of the key stream from the cipher, i.e. 64 bytes from Key Stream Index 0.

13 - Symmetrical Key Wrap - NIST Key Wrap Algorithm wrap of the Master Data Encrypting Key

This is the master encryption key, wrapped using the NIST AES Key Wrap Algorithm, or the generalized version described above if the algorithm used is not AES. The key is derived from a password using PBKDF2-SHA512.

The format is:

```
  0   1   2   3   4   5 ..147 149..... 212 213 ......... 216
+---+---+---+---+---+========+===========+=================+
|252  0   0   0 | 13|Wrap(144)|WrapSalt(64)|WrapIterations(4)|
+---+---+---+---+---+========+===========+=================+

 217 ..... 248 249 .......... 252
+=============+==================+
|DerivSalt(32)|DerivIterations(4)|
+=============+==================+
```

The Key Wrap consists of 128 + 16 bytes, which is the result of the generalized NIST Key Wrap when wrapping a 128 byte quantity with a 256-bit block cipher. The Key Wrap uses the suggested IV 0xA6 repeated to ½ the block-size to ensure the integrity of the wrapped key, and also to test for correctness. The 128 bytes are random key material, used as Master Key, IV etc as appropriate for each algorithm implementation. For a n-bit key block cipher, with m-bits data blocks, the first n-bits are used as the Master Key, and the subsequent m-bits as the IV for CTR mode of operation. Any extra bits are discarded. For stream ciphers, the allocation of bits as key material depends on the algorithm chosen.

A best effort brute force attack is likely to focus on the Key Wrap, not the data which is why the iterations are dynamically determined and calculated to take approximately 50ms on the encrypting system (it is recognized that this may cause an attacker to make a better guess as to the algorithm used. If there are excessive differences in speed between algorithms, countermeasures can be taken when calculating the dynamic iteration count).

The iteration counter is stored as a 32-bit little endian unsigned integer.

The key wrap and salt fields are fixed in size to accommodate key-sizes up to 512 bits and block sizes up to 256 bits. The fields are always filled with key material quality random data, and used as appropriate for the algorithm. When more data is available than needed it is truncated.

This specification removes the need to specify what algorithm was used when encrypting the file, as long as the parameters of the cipher do not exceed 512 bits key, and 256 bits block.

The master encryption key is used directly for counter mode operation.

The key input to the Key Wrap is derived from the user password using PBKDF2-HMAC-SHA512, with a dynamically determined iteration count taking approximately 50ms on the encrypting system. The salt is fixed at 256 bits. The iteration count for the derivation function is stored as a 32-bit little endian unsigned integer.

Header Block 63 - End of Headers

This header block signals the start of the actual data which is encrypted, and optionally compressed as specified in preceding headers.

This header is always the last header block before the data.

The format is:

```
  0   1   2   3   4   5   6   7   8   9   10  11  12
+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 13  0   0   0 | 63| 0   0   0   0   0   0   0   0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

This is the last header block, and is designed to look like an AxCrypt 1.x Start of Data with no Data.

14 - RSA Key Encryption - The Master key encrypted an RSA public key.

This header block contains the master data encrypting key, encrypted with RSA. The private key is typically stored as an AxCrypt-encrypted PEM-representation in the local file system. The RSA key is currently fixed to 4096 bits.

The format is:

```
  0   1   2   3   4
+---+---+---+---+---+===========================+
|    n+5        | 14| n-bytes encrypted key blob |
+---+---+---+---+---+===========================+
```

The encrypted key blob is made by encrypting the master data encrypting key and IV padded to 128 bytes with random data plus 16 bytes 0xA6 bytes, i.e. the same 144 bytes that form the unwrapped key wrap for the symmetric key wrap algorithm. There is no salt or iteration since the underlying private key is assumed to be a full strength-key and not password derived. (It is recognized that use of RSA Key Encryption ultimately depends on the level of protection of the private key, which is why it is protected with AxCrypt-encryption thus simplifying security analysis.)

Header Block 20 - Data

This header block represents one block of data in the encrypted data stream.

The format is:

```
  0   1   2   3   4   5                          n-1
+---+---+---+---+---+==============================+
|       n       |20 | n-5 bytes of Encrypted Data |
+---+---+---+---+---+==============================+
```

n-5 is determined by the Data Block Size header, except for the last block which may be shorter.

Header Block 68 - File Information

This header block is encrypted.

Original timestamps are stored in this header. The decrypted format is:

```
  0   1   2   3   4   5   12  13  20  21  28
+---+---+---+---+---+=======+======+=======+
| 29  0   0   0 | 68| CRT(8)| LAT(8)|LWT(8) |
+---+---+---+---+---+=======+======+=======+
```

The time stamps are as follows:

- CRT - Creation Time
- LAT - Last Access Time
- LWT - Last Write Time

The time is encoded as a 64-bit field, representing the number of ticks since January 1, 1601, UTC. One single tick is one hundred nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond.

Header Block 69 - Compression flag

This header block is encrypted.

This header is a simple flag, indicating if the data is compressed or not. If it is compressed, the format is according to the zlib format with the deflate algorithm.

The decrypted format is:

```
  0    1    2    3    4    5                6
+---+---+---+---+---+================+
| 9   0    0    0  | 69| Is Compressed(1)|
+---+---+---+---+---+================+
```

The is compressed flag is a byte, where a value different than zero is a ‘true’ value, i.e. indicates that the data is in fact compressed.

Header Block 70 - UTF-8 Encoded File Name

This header block is encrypted.

The decrypted format is:

```
  0    1    2    3    4    5            8   9                    8+m       n-1
+---+---+---+---+---+---+---+---+---+====================+=======+
|        n        | 70|        m        | Encrypted File Name | extra |
+---+---+---+---+---+---+---+---+---+====================+=======+
```

The value n is the total length of the header block = 5 + 4 + m + k bytes where m+k >= 256. ‘m’ is the actual length of the UTF-8 encoded file name . ‘k’ is a number such that m+k >= 256, in order to hide the actual length of the file name which would otherwise be obvious.

101 - Plain text lengths

This trailer block is encrypted with key stream index 2048.

The decrypted format is:

```
  0    1    2    3    4    5                  12  13                   20
+---+---+---+---+---+===================+=====================+
| 21   0    0    0  |101| Original Length(8) | Compressed Length(8) |
+---+---+---+---+---+===================+=====================+
```

The length of the original, uncompressed data and the length of the compressed plain-text stream as a 64-bit unsigned integer in little endian order. If the data is not compressed, these fields contain the same value. The purpose of this data is to verify that the uncompressing function restores the same amount of data as was compressed.

Header/Trailer Block 102 - UTF-8 Encoded list of recipient e-mails

This header block is encrypted.

The list of email-addresses is serialized as a JSON array. Key Stream Index 3072.

The decrypted format is:

```
  0    1    2    3    4    5           n-1
+---+---+---+---+----+--------+---+---+
|        n        | 102| JSON []| 0 | 0 |
+---+---+---+---+----+--------+---+---+
```

The value n is the total length of the header block = 5 + 2 + the length of the JSON-encoded array of recipient e-mails.

103 - Algorithm Verifier

This header block is encrypted with key stream index 4096.

The decrypted format is:

```
  0    1    2    3    4    5         20 21                 36
+---+---+---+---+---+============+=====================+
| 21   0    0    0 |101| Random(16) | Random XOR 0xFF(16) |
+---+---+---+---+---+============+=====================+
```

A random sequence of 16 bytes are followed by it's one complement. The purpose of this data is to verify that the correct master key and algorithm are known, this is needed for asymmetric encryption since the encrypted master key is independent of the actual algorithm used.