AES & FLAM®

(Frankenstein-Limes-Access-Method)

The Advanced Encryption Standard (AES) for encryption
the Advanced Data Compression (ADC) as of FLAM® V4.0

Table of contents

List of figures

1 AES Cryptography as of FLAM® V4.0

1.1 Introduction

The Advanced Encryption Standard (AES) replaces the outdated Data Encryption Standard (DES). This modern symmetric block cipher forms the basis for the cryptographic protection of a FLAMFILE® as of FLAM® 4.0. Compared to DES, the cipher is considerably more secure and requires only a tenth of the processing time. In combination with the ADC compression, this makes it possible to use strong cryptography on large data sets.

Remark: For security reasons, DES has been applied thrice for years. *Triple DES* tripled the already high consumption of CPU time.

In FLAM®, AES is used with a block and key length of 128 bits (16 bytes). The following figure shows the two basic operations (AES schedule and AES encryption) which are used in combination.

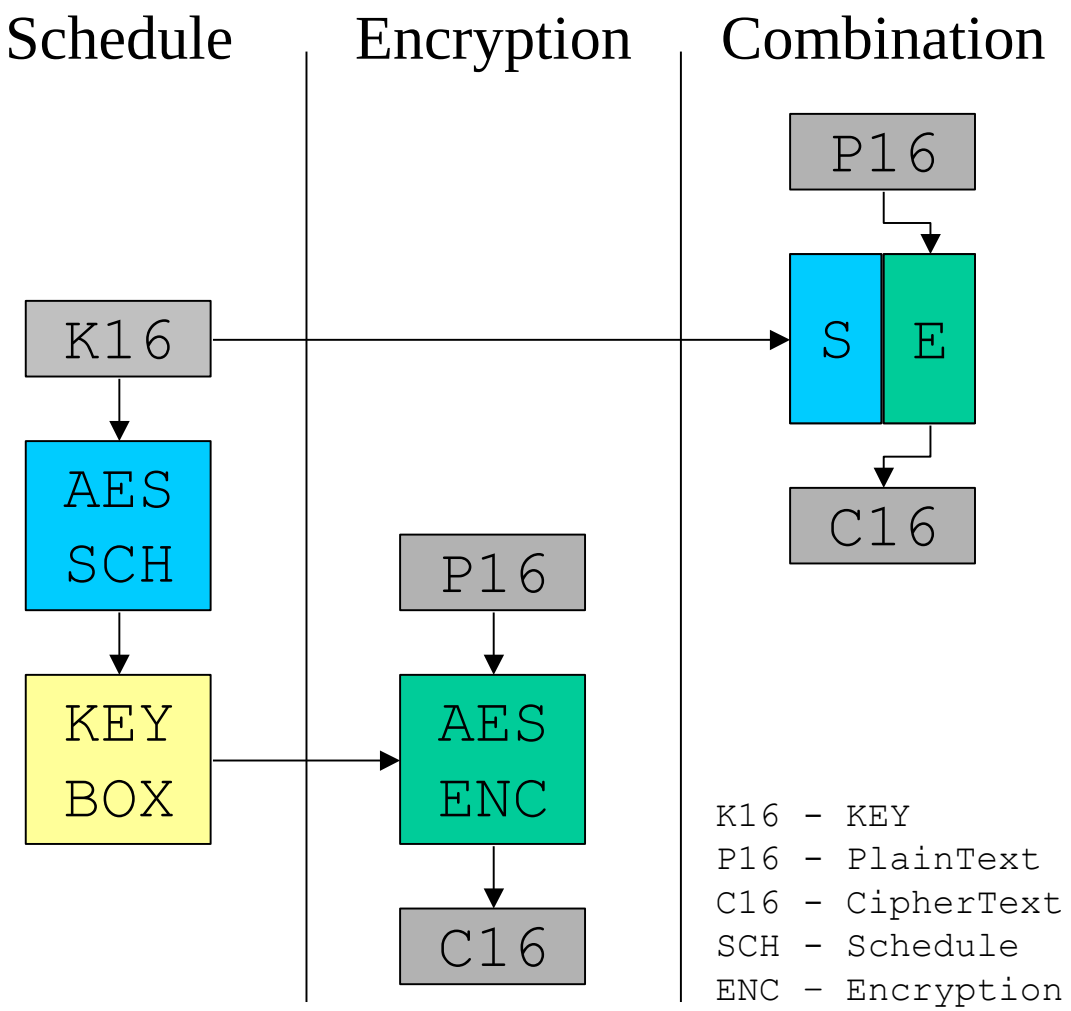

Figure 1 The base algorithm

The decryption operation (Decryption (DEC)) is not required by FLAM® because the data is encrypted and decrypted by a stream cipher in CFB mode (Cipher Feedback) (see Bruce Schneier, Applied Cryptography).

AES encryption with FLAM® is possible in MODE=ADC® (Advanced Data Compression) or MODE=NDC (No Data Compression) which is a sub-function of the ADC algorithm. When using NDC, the input data is compressed without prior modifications. Thus, any FLAMFILE® can be encrypted on retrospect without performance loss (two-step method). This way, even an "empty" file can be encrypted so that it is no longer recognizable as "empty".

How the confidentiality and the integrity of data is protected in a FLAMFILE® is described in the following sections.

This data protection uses pure software cryptography, which means that the unencrypted keys used exist in the system's memory for a brief period of time when a FLAMFILE® is created. However, since the original unencrypted data is also present on this very system, an attacker with access to the system can read the plaintext data anyway. The cryptographic key is only useful for the attacker if it is used another time and the attacker no longer has access to the system.

The maximum security FLAM® can provide you with AES is dependent on the security of the computer on which the FLAMFILE® is written or read. FLAM® ensures through AES that no one on the transmission path can eavesdrop or manipulate the data without knowledge of the key. This security can be improved even further by transmitting the encrypted FLAMFILE® between servers on which neither FLAM® nor the original data are available. This is a simple organizational measure which substantially increases security. This organizational solution is also much safer than a combination of file transfer and the use of cryptography in direct communication between transmitting and receiving system (transport security).

Cryptography alone is not a guarantee for safety without a corresponding organizational environment.

An interesting organizational solution offered by FLAM® V4.0 in conjunction with cryptography is Parallel Splitting. Through splitting of an encrypted FLAMFILE® into units of 4 bytes and uniform distribution of these units to multiple files, decryption is only possible when the correct key and all file parts are passed to FLAM®. This may solve the problem of key synchronization (e.g. distribution to different locations for long-term archiving).

There is a FLAM® V4.0 feature that allows you to check the technical integrity of a FLAMFILE® (whether encrypted or not) through checksums on the basis of CRC routines. Such techniques are international standard, e.g., in conjunction with file transfer. *They do not protect against manipulation.*

The integrity of a FLAMFILE® encrypted with FLAM® V4.0 and AES can be validated cryptographically *without decompression.* This, however, requires using the key with which the FLAMFILE® has been generated.

1.2 Overview

FLAM® accepts a passphrase of up to 64 bytes as key information for encryption. It can be passed in different ways when calling FLAM® (see manual for details). It is the basis for encrypting the FLAMFILE®. This corresponds to the input interface of FLAM® V3.0 which also accepts keys with a length of up to 64 bytes (=512 Bits) for the encryption of a FLAMFILE®.

The old high-performance encryption was not based on well-established cryptographical procedures, but on an extension to the compression algorithm ADC (Advanced Data Compression) and was a development of limes datentechnik® gmbh. In many areas of information security, established cryptographic methods are required to substantiate security. Therefore, FLAM® 4.0 also offers encryption based on the Advanced Encryption Standard (AES). The principle of how the protection of FLAMFILE® using AES works is described below.

FLAM® is a compression utility that can process multiple files. Files in FLAM® are called members. These members are compressed into autonomous segments of up to 64 kilobytes. All compressed members are combined into one FLAMFILE®. This results in a tree with 3 levels (file, members, segments). This hierarchy must reflect the cryptographic system. The following figure provides an overview of the data format in a FLAMFILE®.

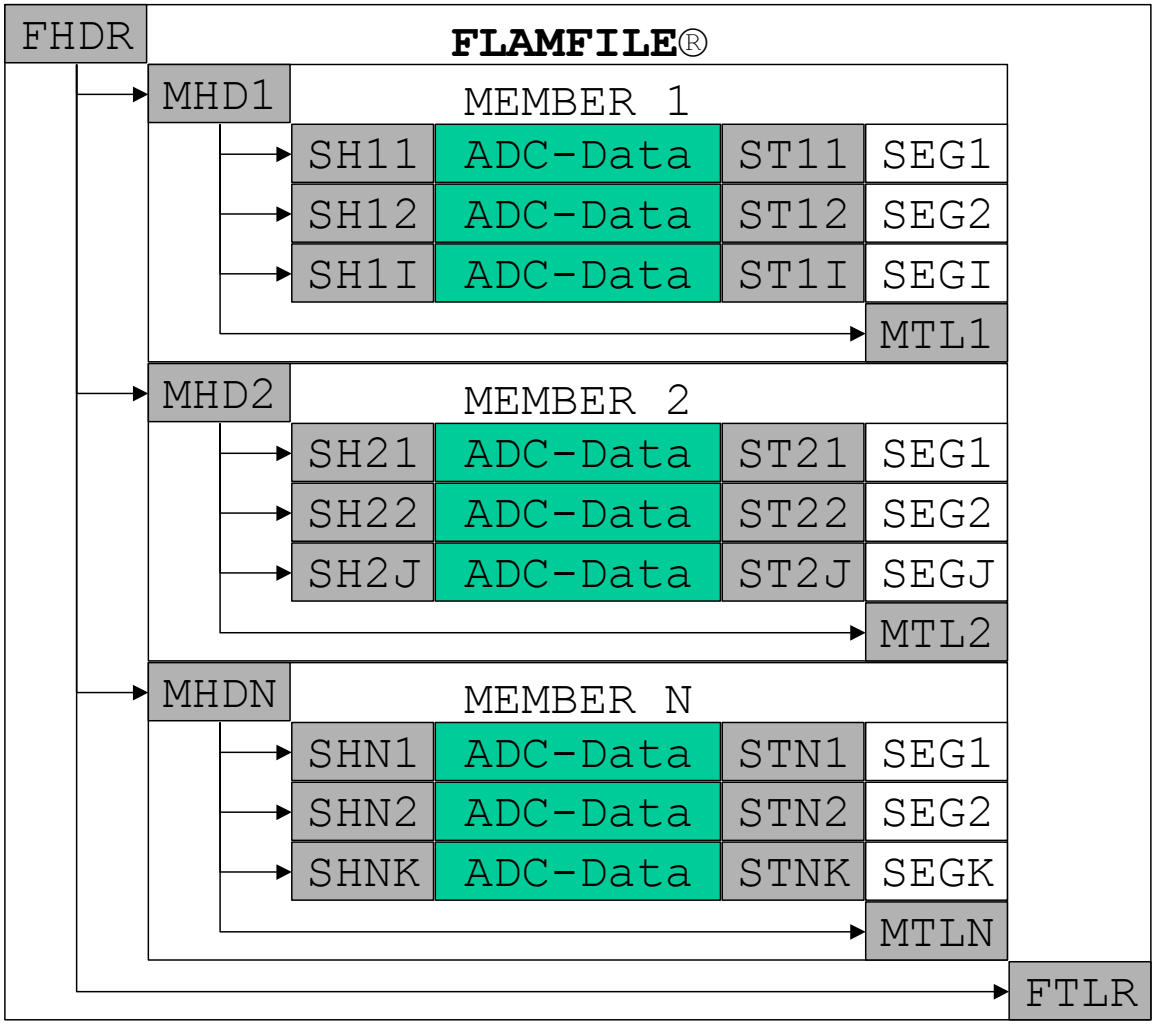

Figure 2 The data format of a FLAMFILE®

The peculiarity of FLAM® is the direct access to individual segments of logical units (records of variable length). This makes it possible that only those parts of the FLAMFILE® are decompressed that are needed. For this purpose, FLAM® is also available as a subsystem for file access. In case of errors or sabotage remaining intact segments can still be rescued. The capability to directly access specific segments requires autarkic cryptographical treatment of each segment. This includes the protection of the confidentiality and integrity of the compressed segments. The composition of a compressed segment in FLAM® is illustrated in the following figure.

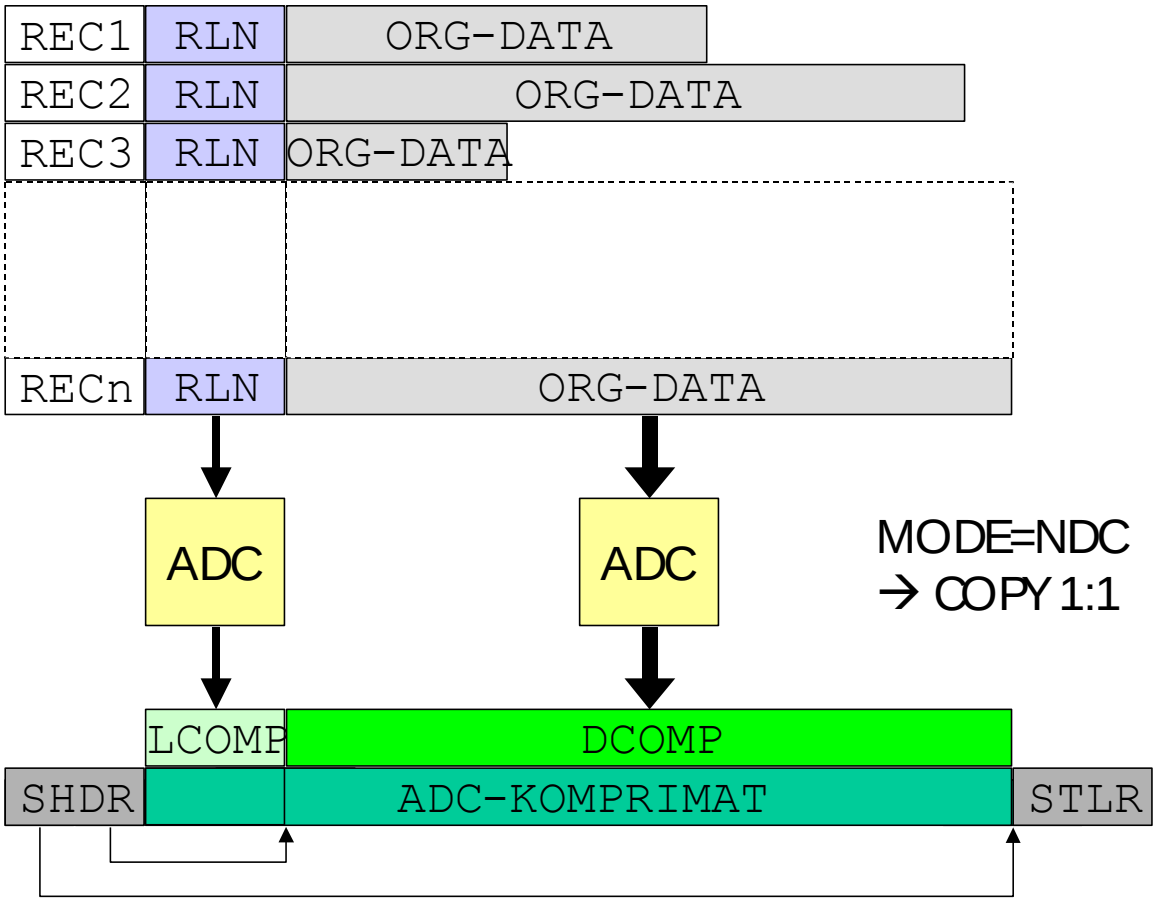

Figure 3 The composition of a compressed FLAM® segment

In FLAM®, only the (compressed) data depicted in green (dark) is encrypted (compressed dataset attributes + compressed data (ADC) or uncompressed data (NDC)). All the rest, that is, meta information in the headers and trailers of a FLAMFILE®, is protected against manipulation by means of cryptographic checksums, so-called MACs (Message Authentication Codes).

The figure below illustrates that all components of a FLAMFILE® are protected cryptographically.

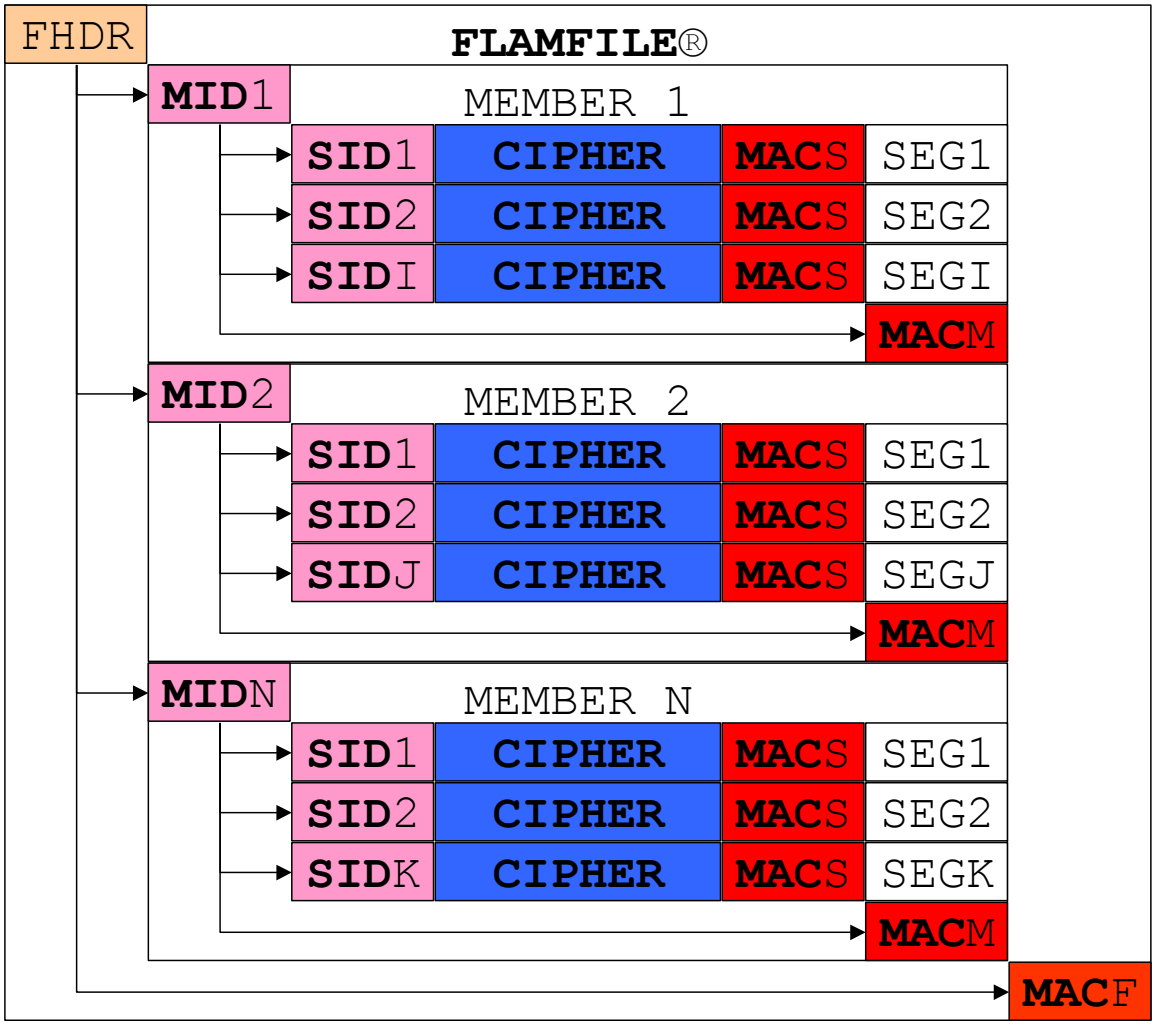

Figure 4 The cryptography in a FLAMFILE®

Due to the self-sufficiency of members / segments, it would be possible to add, remove or exchange whole valid members / segments from a FLAMFILE®, which would correspond to a manipulation. Therefore, the order and completeness of the members / segments must be checked. In summary, we define the following cryptographic protection measures for a segment:

- Encryption of the compressed segment
- MAC calculation for encrypted and compressed segments (MAC1)
- MAC calculation for the cryptographic chaining of the segments (MAC2SS)
- MAC calculation for the segment header and trailer, including MAC1 & MAC2SS (MAC3)

In result, the segment and the member identification (MID, SID) are cryptographically secured data (by means of AES).

This procedure is determined by the proven internal program structure of FLAM®, which is retained in the version 4.0.

The following figure shows the integrity protection for a segment.

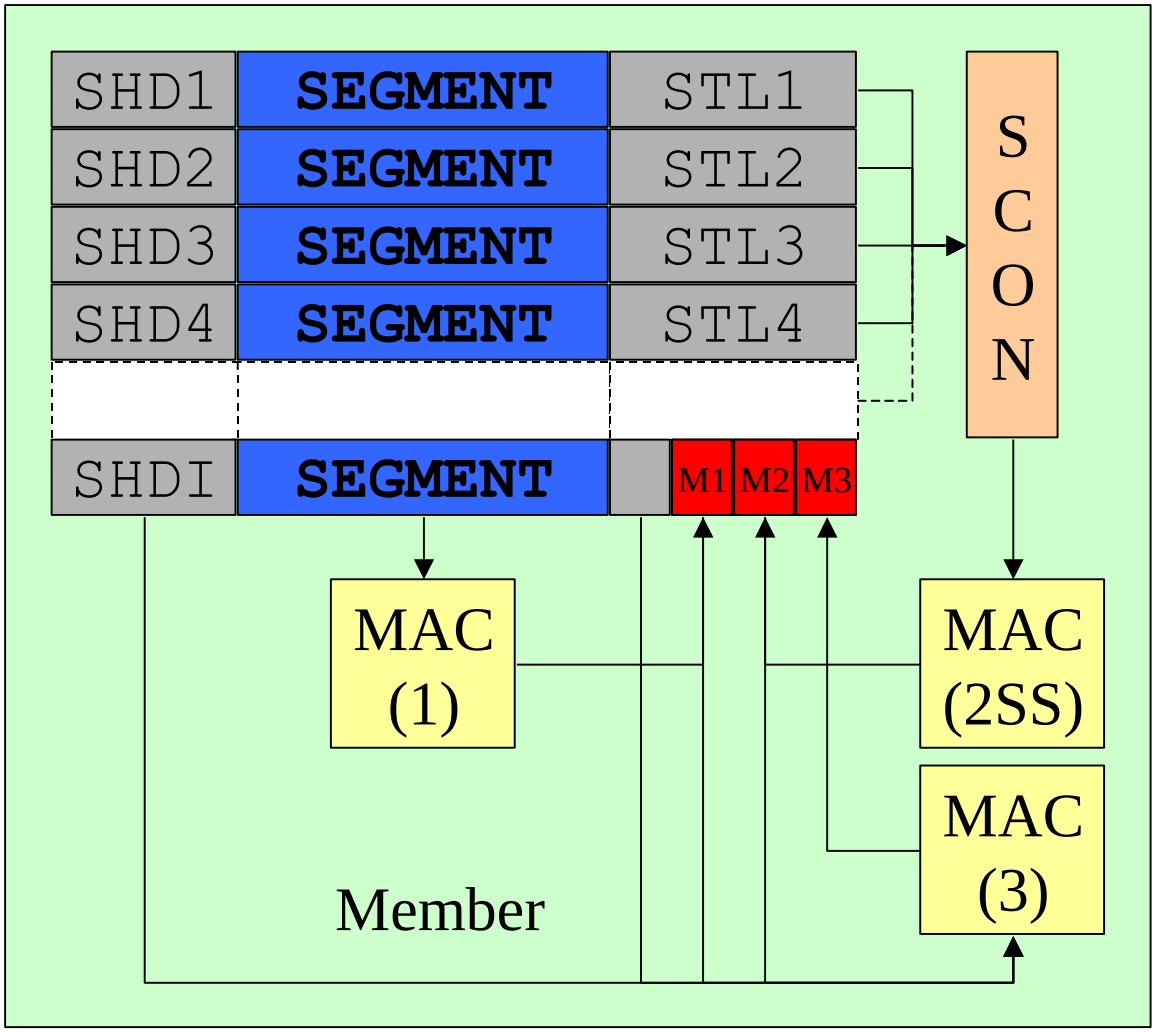

Figure 5 Integrity protection on segment level

On member level, the order and completeness of entire members must be ensured. The segment connectors (SCON) with 128 bits ensure an AES-based chaining of significant information between segment trailers in a member.

This includes the transition from the last segment to the member trailer (MAC2SM) and the transition from member to member (MAC2MM). These two MACs, along with the member header and additional meta information in the member' s trailer, are protected by MAC3.

The following figure shows the integrity protection for a member.

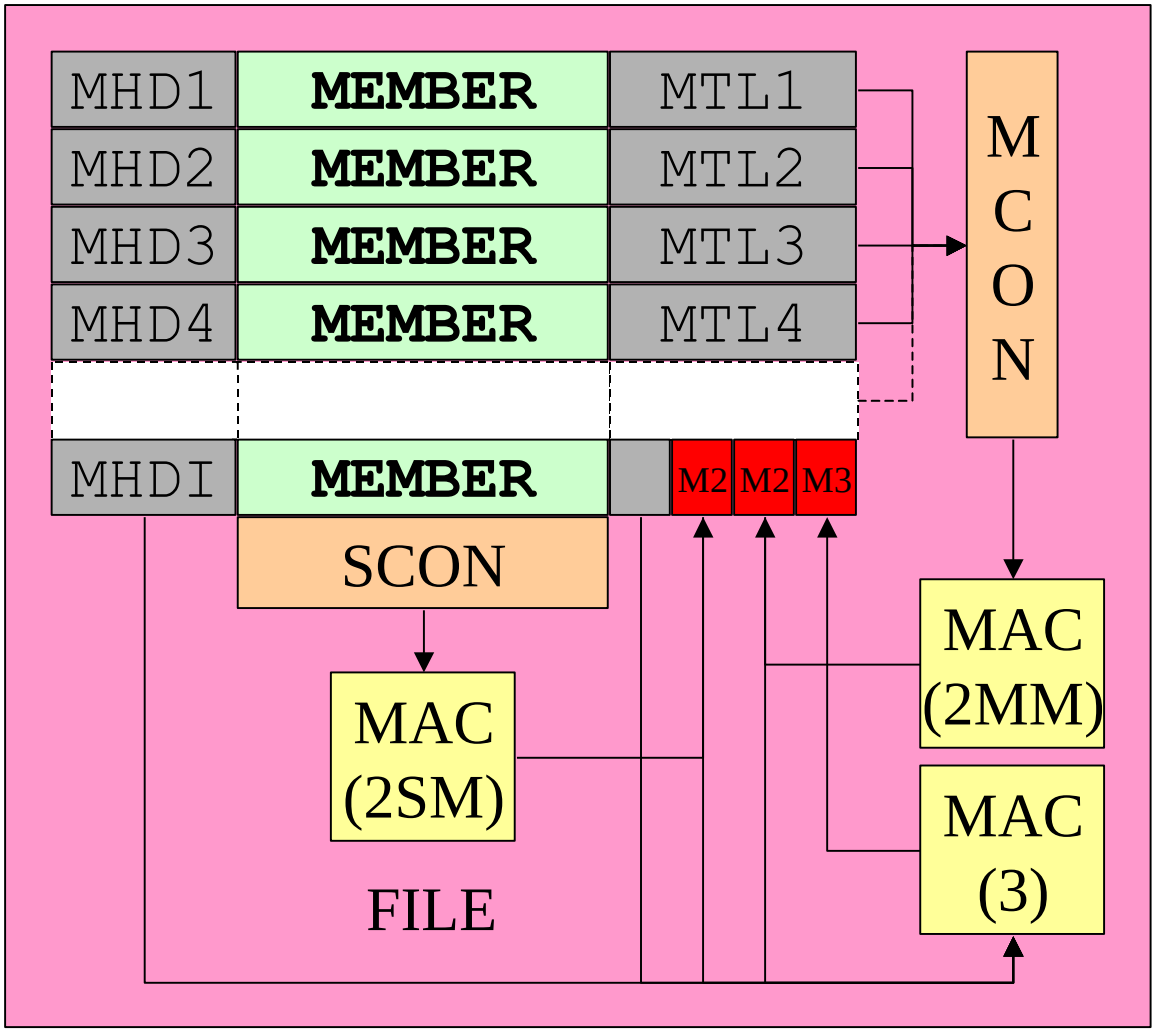

Figure 6 Integrity protection on member level

The member connectors (MCON) with 128 bits ensure an AES-based chaining of significant information between member trailers.

On file level, the chaining of the last member and the FLAMFILE® trailer (MAC2MF) need to be ensured before MAC3 can be calculated for the file header and file trailer (incl. MAC2MF).

The following figure illustrates this process.

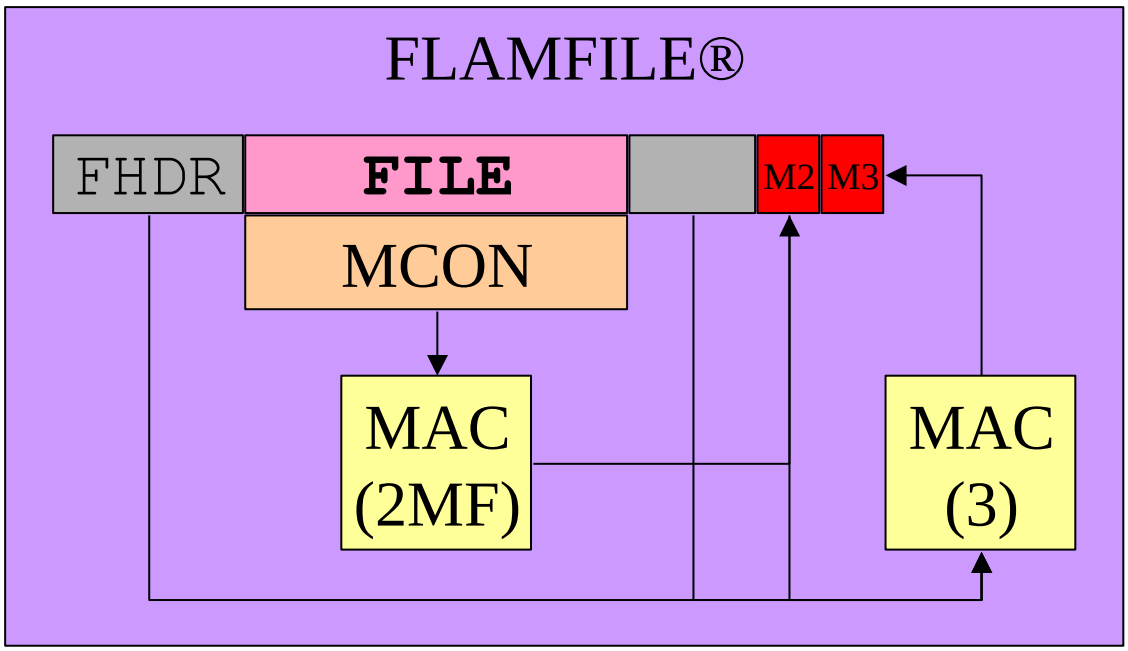

Figure 7 Integrity protection on file level

Due to the dedicated integrity protection on all levels, members / segments can optionally be accessed directly or the file can be processed as a whole, while ensuring that no one has manipulated the accessed data.

Due to the fact that whole compressed segments are encrypted, a potential attacker does not have a single byte of plaintext available for an attack. Data compressed with ADC ideally consists of redundancy-free "white noise". This further aggravates a potential attack. Even when MODE=NDC (No Data Compression) is used, the data is permuted (scrambling) before encryption so that it cannot be distinguished from compressed data.

Tiniest bit errors in the decrypted data render the reconstruction of the original data by un-scrambling / decompression impossible. Errors in the MACs indicate that further processing of the decrypted data would not make sense. Possible reasons may be errors in the data or the use of an incorrect key. Data errors due to technical problems are extremely rare nowadays.

To check whether an input key is correct or not, FLAM® has to decrypt the whole first block. The MAC checks at the segment level eventually provide information as to whether the key was correct. Due to this relatively expensive operation, FLAM® itself is not an effective attacking tool for brute force attacks. This does not pose any disadvantage to the authorized user in regard to performance. Due to the complexity of the layered model of FLAM® (see above), it is not possible to use a custom solution for significantly faster cracking attempts.

1.3 The Procedures

All cryptographically relevant processes of FLAM® are solely based on the AES algorithm, which - in the worst case - can be easily replaced by another algorithm (transparent replacement). Thus, the FLAM® concept is invariant and future-proof in regard to such changes.

1.3.1 Key derivation

At the segment level, four cryptographic operations are performed. A basic rule of cryptography is:

- Use a separate key for each operation.

This means that four keys are needed at segment level. Another basic rule of cryptography can be formulated as follows:

- For each element, a unique key must be used.

It follows that it is necessary to generate four unique keys for each segment, three for each member and two for every FLAMFILE®.

FLAM® can directly access specific segments through the help of unique identifying data in the segment headers (unique addressing scheme). This information is used as derivation data for the segment-specific key, which leads to unique key values. The same applies on member level, where 3 different keys are needed. From the up to 64 bytes long passphrase, 16 bytes (128 bits) long keys are derived on file level. They are used to calculate the MACs on file level and to derive four member-level keys. These, in turn, are used for MAC calculation and derivation of the four segment-specific keys.

To calculate MACs and encrypt the data, an associated initial value is required. Taken together, this forms four pillars which provide four specific unique keys and four specific unique initial values on each level.

The following figure illustrates this.

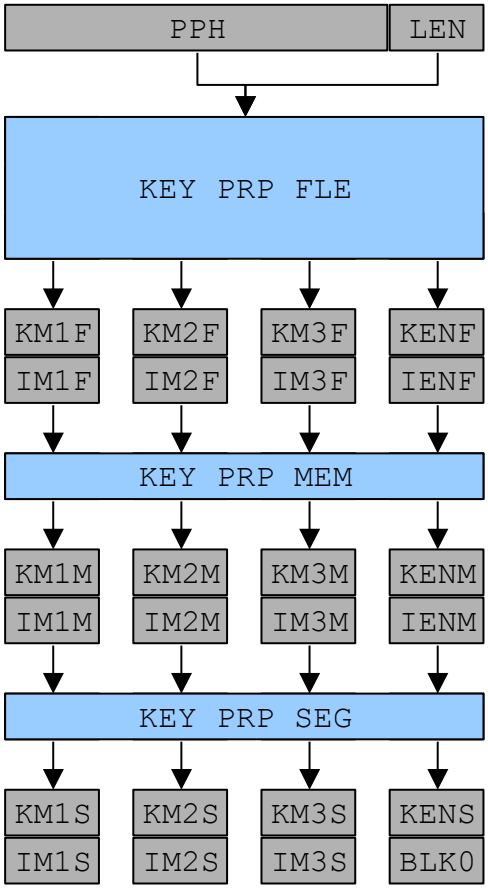

Figure 8 The four pillars of key derivation

The methods used for key derivation are cryptographic one-way functions based on AES which makes it impossible to recover the original key from the derived keys and represents a pseudo-random process.. Further information can be found in the detailed specifications.

Additionally, the current time, the machine ID and other context-specific information is included in the key derivation process, resulting in a fundamentally different ciphertext even if the same input key and input data is used on consecutive runs. This characteristic increases security.

The data in the headers that are responsible for this uniqueness property are protected cryptographically by MACs using AES.

1.3.2 Encryption of a compressed segment

As already mentioned, encryption of the compressed segments is done in Cipher Feedback Mode (CFB), avoiding the need to pad the end of segments to full 16 bytes blocks. Known plaintext attacks are effectively prevented by a pre and post XOR operation with additional pre-computed segment-specific pseudo-random sequences (S256/Sk, MS256/Mk). For 16 plaintext blocks, it looks like this:

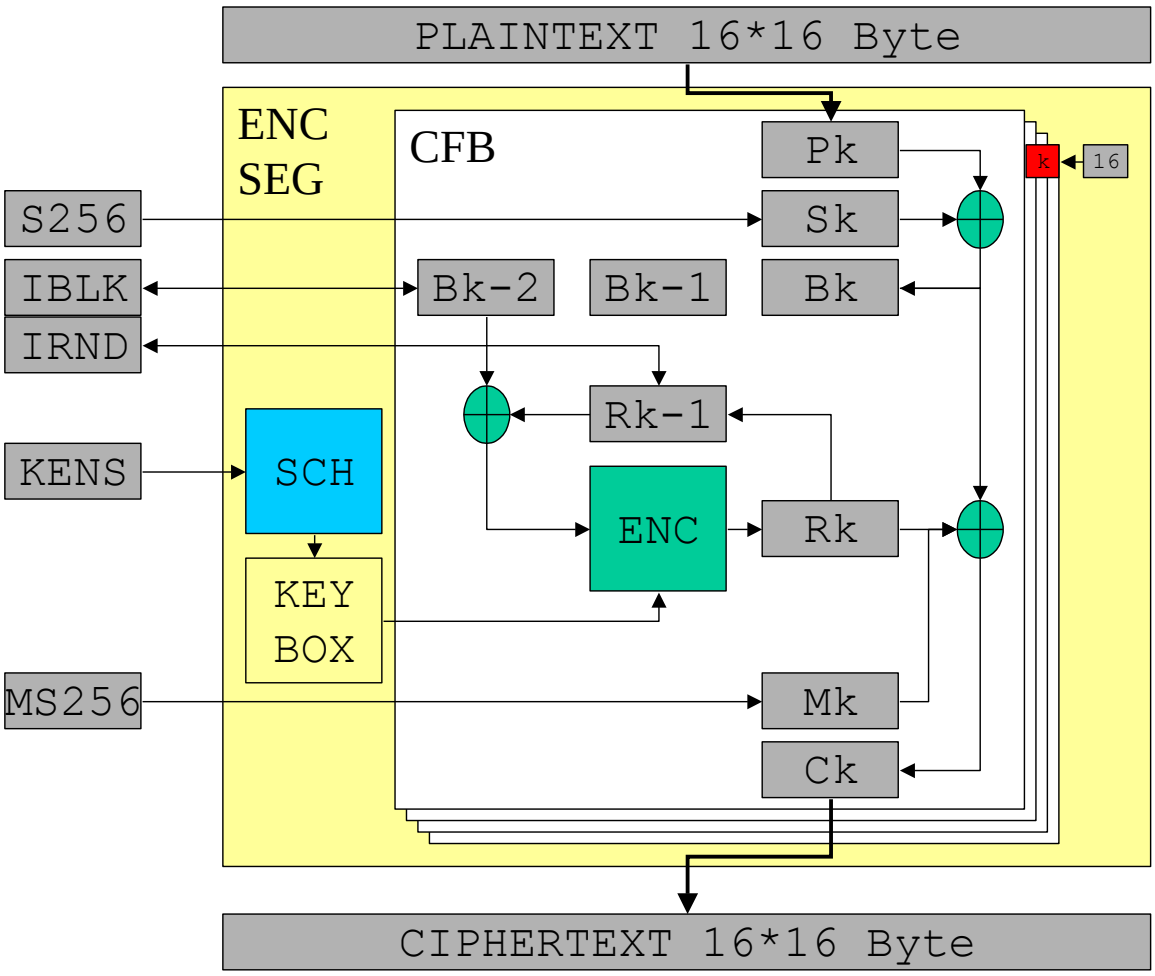

Figure 9 Encryption of a compressed segment

The random value (Rk) used for the encryption of the block (Bk) is built from the random value (Rk-1) and the block (Bk-2). The XOR product of (Rk-1) and (Bk-2) is encrypted with the key box (KENS). This results in the random value (Rk). IBLK and IRND are initial values.

Further information can be found in the detailed specifications.

1.3.3 MAC calculation over the encrypted and compressed segment

MAC computation over the previously encrypted compressed segment takes place in the cipher block chaining mode (CBC). This MAC conforms to the ANSI 9.9 MAC process in which DES was replaced by AES. The following figure illustrates this.

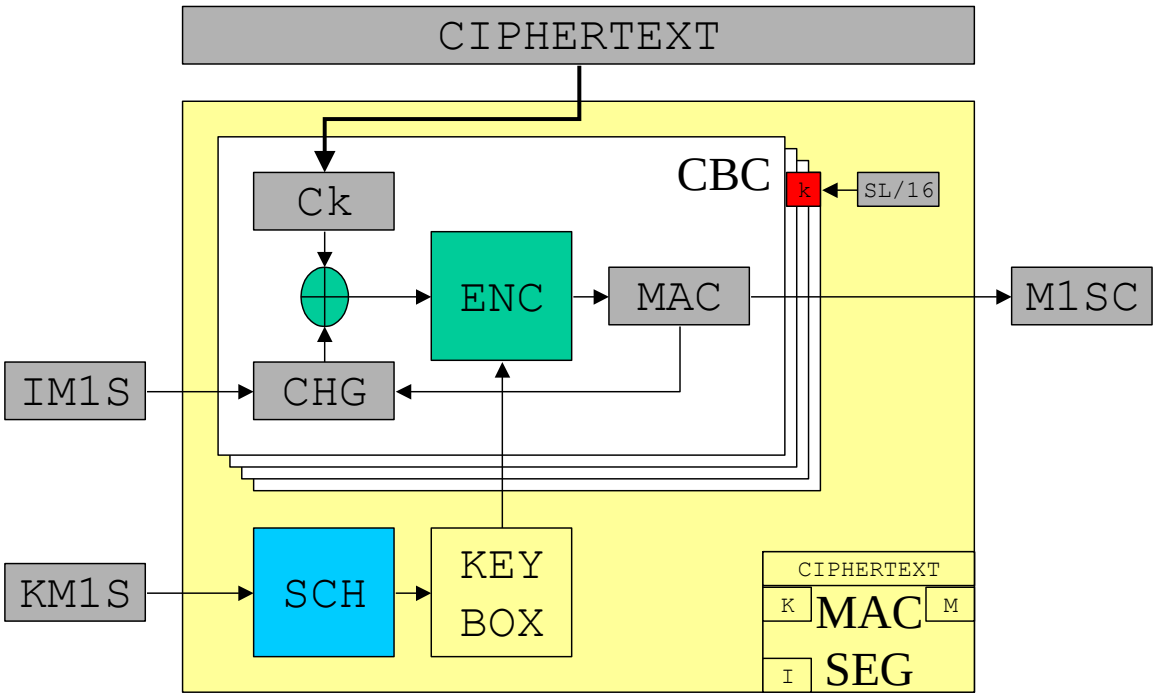

Figure 10 MAC calculation over the encrypted and compressed segment

The intermediate result (CHG) is XORed with the cipher block (Ck). The XOR product is encrypted with the key box (KM1S). This results in the new intermediate result (CHG). IM1S is the initial value of this sequence (CHG). The final result is the MAC (M1SC).

Further information can be found in the detailed specifications.

1.3.4 Remaining MAC calculations

For the remaining MAC calculations a hashed MAC method is used. AES is used to calculate a cryptographic 128 bit hash value. The method used is the simplest of those considered safe for computing a cryptographic hash value based on a symmetric block algorithm when the block length and key length are equal to the hash length.

- $HSHi=ENC_{HSHi-1}(BLKi)$ XOR BLKi (see Bruce Schneier, Applied Cryptography)

This hash value is then encrypted with AES using the respective key, resulting in the MAC.

Further information can be found in the detailed specifications.

1.4 Summary

The structure of a FLAMFILE®, the characteristics of the product and the internal program structure constitute the cryptographic system that is inevitably characterized by a certain complexity. Furthermore, it was the goal of the limes datentechnik® gmbh to eliminate any attack vector against keys on any level, which has led to extensive additional protective measures. All these measures are described in the detailed specification, as they are beyond the scope of this documentation. The limes datentechnik® gmbh feels obliged to publish all security measures.

Remark: AES with 128 bit key length is US federal government standard since May 26, 2002.

A comprehensive online documentation for AES can be found at the following address:

http://csrc.nist.gov/archive/aes/

NIST is the US agency that declared AES as US federal standard.