TR-31 Equivalence of Key Blocks in Cryptosec Dekaton

Version 1.0 (20201109)

Ernst-G. Giessmann
Humboldt-Universität zu Berlin[1]

Contents

[1] mailto:giessmann@informatik.hu-berlin.de

History

Version	Date	Remark
1.0	2020-11-09	Final Version

1 Introduction

1 For the management of encrypted symmetric keys in structures, the ANSI X9 Standard TR 31-2018 [TR-31] specifies a key usage binding to the key value in the key block. Whereas there is substantially only one method specified, the other is for legacy TDEA only, it is allowed to use equivalent methods.

2 The present report analyzes the Dekaton Container Format V1 and shows that it uses such an equivalent method and bases on generally accepted algorithms.

2 Key Blocks according to TR-31

2.1 Key Block Elements

3 The key block consists of three parts:

- The Key Block Header (KBH) which contains attribute information about the key and the key block.
- The confidential data that is being exchanged/stored.
- The MAC that binds the Key Block Header and Encrypted Key Block.

4 The Key Block Header (KBH), which contains attribute information about the key and the key block, is not encrypted. It contains a first section of 16 bytes with a fixed format defined.

5 The confidential data to be encrypted contains two bytes indicating the encrypted key length, the key/sensitive data that is being exchanged and/or stored and an optional random padding.

6 The MAC, which is 64 bits for the key derivation method, is calculated over the header and the binary key data, using CMAC with the Key Block Authentication key.

2.2 Key Block Binding Method Using Key Derivation

7 The encryption and authentication keys are derived from the Key Block Protection Key using CMAC [SP800-38B] as a pseudorandom function to produce 16-byte MAC values as detailed in the remainder of this section. The length of the derived binding keys is equal to the length of the key used to derive them.

8 The input data to the CMAC function consists of a counter, a key usage indicator, an algorithm indicator and a length of the keying material being generated.

2.3 Defined values for Key Block Headers

9 The ANSI standard contains currently [TR-31, A.5.1] a list of 37 key usage types associated to the key to be protected (2 bytes).

10 The ANSI standard contains also [TR-31, A.5.2] a list of 7 different algorithms (1 byte), e.g., AES and ECC, and a list of 8 different modes [TR-31, A.5.3], e.g., encryption or signature.

11 Other values are the Key Block version number, the Key Block length, the key itself version number, exportability, and the optional fields value.

3 Key Containers used in Cryptosec Dekaton

12 The structure corresponding to the Key Blocks is called *Container* in Cryptosec Dekaton (cf. [DCF]). For this document only version 1 of this format is relevant.

3.1 Container V1 Elements

13 The Container consists of four parts:

- The fixed length header block (24 bytes) which contains container information, key type, IV value, and the lengths of the plaintext and the ciphertext fields.
- Plaintext data, which is authenticated but not encrypted,
- the confidential data that is being exchanged/stored, and
- the MAC that binds the plaintext and key data.

14 The Container header contains the cryptographic relevant Key Type (2 bytes) and 16 bytes nonce used for the key encryption as Initialization Value.

3.2 Key Type

15 The Key Type field in the header is not sufficient to encode alone all the different key types of the TR-31 Specification, e.g., it is set to 0x0000 for any Banking AES key. Nevertheless, the key usage information according to TR-31 is stored for Banking Keys by the LMK type [DBLHC], e.g. LMK type 7 is assigned to Data Encryption, whereas type 8 is for Authentication Keys. Note that TR-31 does not require to implement all the key usage types, but to avoid usage for different purposes. This requirement is fulfilled by the different LMK selections.

16 For PKCS#11 keys a flag byte in the plaintext field of the Container can be used for differentiation. Note that the different key usages are associated with the bits of the flag, and therefore an “all-purpose” key could be defined. But when operated in PCI PTS HSM v3.0 mode it is guaranteed by the implementation, that the key flag bits correspond bijective to LMK indices. This implies that the key type together with the key flags byte are equivalent to the key type as required by TR-31.

17 The Key Usage types are stored in the Container header and the plaintext field. Both are protected by the MAC and are cryptographically bound together.

3.3 The Nonce

18 The nonce in the header field is used as Initialization Value (IV) for the encryption of the key data. This is different to the TR-31 specification of using the MAC value as IV value in the CBC mode encryption [TR-31, 5.3.2]. The advantage of the TR-31 approach is space saving. The MAC of TR-31 is a pseudorandom value.

19 The selection of a random value generated by the HSM during container setup is neither better nor worse than the TR-31 approach from a cryptographic point of view. Both are unpredictable[2] by an adversary and the birthday paradox is not applicable due to the IV length of 128 bit.

3.4 Plaintext Field

20 The Key Block Header of TR-31 is of fixed length, whereas the plaintext field of the Container considered as an additional header is of variable length. But TR-31 (see Annex A.1 [TR-31, p. 15) allows for optional headers too. Therefore, the plaintext field of the Container V1 is even covered by the ANSI specification.

21 The plaintext field length is specified in the Container header. The plaintext itself depends on the key type. Both length and content of the plaintext field are protected by the MAC.

3.5 Key Derivation

22 The encryption and the authentication key in TR-31 and for the Container V1 are derived by a CMAC application. As input serves in both cases ([TR-31, 5.3.2.3] and [DKT, 5.3] the concatenation of a counter (1 byte), a key usage field (label, 2 bytes), one zero byte as separator, an algorithm identifier (indicator, 2 bytes), and a 2 byte integer representing the key length in bits.

23 The encryption key in both cases is applied as an AES-CBC key. They differ in the usage of the message authentication key, which is a CMAC key in TR-31 and an HMAC key for Container V1 (see section 3.8 in this document). As they provide the same level of randomness, they can be considered as equivalent.

3.6 Encrypted Data

24 The encrypted data field, i.e. the encrypted key, is built by AES-CBC encryption [DCF, 3.8], which is also the preferred algorithm in TR-31. The input data for the encryption is padded at least to the encryption block size. But in fact the padding is extended to cover the maximal key length for the selected key type in the encrypted data. For example, an encrypted AES-128 key cannot be distinguished from an encrypted AES-256 key because of the padding, and the key length cannot be inferred from the length of the encrypted data.

25 In TR-31 section 5.3.2.1 the key length is part of the encrypted data. This can be used “to hide the true length of short keys” [TR-31, Annex A.1 p. 15]. The approach undertaken in [DCF, 3.8] which uses not the key length but the length of the padding, is fully equivalent to TR-31 and hides the key length as well.

26 The key length in TR-31 is measured in bits with two bytes reserved for the length. This allows for a key length of less than 2^{16} bits or less than 256 bytes. The padding length in

[2] The DRNG of the HSM is certified.

Container V1 is given in bytes by the padding bytes itself and allows for a padding of up to 256 bytes, i.e. the maximal key length of TR-31.

27 The CBC encryption relies on the randomness of the Initialization Value. This is guaranteed by the random selection of the nonce in the header block (see 3.3).

3.7 Header Block Padding

28 A header block padding to a multiple of 8 bytes as described in [TR-31, A.2 and Fig. 8] is not necessary for the Container V1 implementation, as the integrity protection of the header is not based on a block encryption algorithm. The header block has a fixed length of 24 bytes, which is a multiple of the word size of SHA-256, the base for the HMAC algorithm [DCF, 3.9].

3.8 Message Authentication Code

29 The Message Authentication Code is computed over all Container fields except the MAC itself, whereas in the ANSI Specification [TR-31] the MAC is computed over the header and the key plaintext data. These different approaches are known as "*encrypt-then-authenticate*" and "*encrypt-and-authenticate*". Both are used in standard applications, e.g., the first in the IPSec protocol and the latter in the SSH protocol. There are several papers in the scientific community on the differences (cf. [BelNam] or [Kraw]). Summarizing their results it can be concluded that the TR-31 application "*encrypt-and-authenticate*" using a CBC encryption is secure, whereas the Container V1 approach "*encrypt-then-authenticate*" has even security advantages. Any changes to the ciphertext can be filtered out before decryption applies. Certainly, the used encryption and authentication methods must be approved and secure, which is here the case. Note as well, that the Initialization Value (IV) and the used encryption method must be protected by the "*encrypt-then-authenticate*" method, which is also fulfilled by the Container V1 specification.

30 The Message Authentication Code in the ANSI Specification is a CMAC, whereas Container V1 uses the HMAC function [DCF] with the hash function SHA-256. The HMAC function is standardized in RFC2104 and it is secure as long as the hash function used is approved, which applies here. The output of SHA-256 is 32 bytes long and therefore even longer then the CMAC used in TR-31, which consists of 16 bytes.

4 Requirements of PCI PTS HSM Evaluation FAQs

31 Recently a new version of the FAQ was published. In the following the required equivalence properties of Q18 ([FAQ, HSM Requirement B11]), which partly exceed the requirements of TR-31, are analyzed.

32 *It must prevent the loading of PIN, MAC, and/or Data keys - or any keys used to manage these within the key hierarchy - from being used for another purpose. IPEK, KEKs, and derivation keys must be uniquely identified where supported* (Q18a).

The key usage is unambiguously identified by the key (object) type in the Container header and additional information in the flags of the plaintext field. Both are protected by the MAC.

33 *It must prevent the determination of key length for variable length keys.* (Q18b).

In Container Format V1 a padding is always applied up to the maximal key length of the algorithm type, i.e. to 256 bit for AES and 192 bit for DEA. Therefore, the real key length

cannot be determined. The padding with a zero byte before the CMS padding according to RFC5652 (sec. 6.3] is appropriate for hiding the key length for variable length keys.

34 *It must ensure that the key can only be used for a specific algorithm (such as TDES or AES, but not both)* (Q18c).

This is ensured because there is a bijective correspondence between key types and algorithms. TDES and AES have different key types and can only be used according their key type..

35 *It must ensure a modified key or key block can be rejected prior to use, regardless of the utility of the key after modification. Modification includes changing any bits of the key, as well as the reordering or manipulation of individual single DES keys within a TDES key block* (Q18d).

The specification of the Container V1 implementation requires that the HMAC of a key block is checked before encryption takes place. If the MAC verification fails, the key will be rejected. The probability of a MAC collision is negligible due to the avalanche property of the SHA-256 hash function.

36 *Where different key block formats are supported, with some providing the above protections and some not, it must be humanly readable from the key block prior to loading/use which format is implemented. E.g., by looking at the commands sent to the device* (Q18e).

The Container V1 specification does not support multiple key block formats. The command dump clearly identifies the key block format used.

37 *It must support all symmetric algorithms implemented by the device(s) that are to use the key blocks* (Q18f).

This is fulfilled, all symmetric algorithms implemented by the device can be protected by the Container V1 blocks.

38 *Where asymmetric algorithms are supported, the algorithm type, padding and signature formats must be identified in the key block* (Q18g).

Asymmetric keys, considered as protected by the key block, are supported by the Container V1 format. Their identification as an algorithm of a specific type, as an encryption/decryption or signature/verification key is given by the key type field, e.g., an ECDSA (standardized in [FIPS186]) private key is a signature key, whereas an RSA (standardized in [RFC8017]) public key with the encryption flag is an encryption key [DCF, chap. 4].

39 *It must use NIST approved modes of operation, with separate keys used for confidentiality and authenticity. Any keys used must not be related in a reversible way* (Q18h).

The key derivation is implemented as required by TR-31, so this requirement is fulfilled automatically [DKD, sec. 3.2].

5 Conclusions and Recommendations

40 The review shows that the encrypted key and its attributes in the Key Block have integrity protection (see section 3.8), and it is computationally infeasible for the key to be used if the key or its attributes have been modified. Any change of the attributes, even of any bit, in the Container header leads to a different HMAC value and will be detected before the key data is recovered.

41 The algorithms used in Container V1 for encryption (see section 3.6) and their modes of operation are also listed in TR 31, so that the equivalence is given for confidentiality too.

42 The key derivation [DKD] is implemented in the TR-31 scheme, and so the equivalence is fulfilled a-priori.

43 The key usage attributes required by TR-31 section 5.3.2, some additional attributes are also listed in Annex A.5.1 [TR-31, pp. 20-21] are implemented as Container V1 attributes in form of the type attribute in the Container header or by the flags in the plaintext field. They are sufficiently granular and clearly distinguish between different usage types. Their protection by the HMAC guarantees the binding of the key and its usage.

Appendix Glossary and Acronyms

Acronyms

Acronym	Term
ANSI	American National Standards Institute
CMAC	Cipher-based Message Authentication Code
DRNG	Deterministic Random Number Generator
ECDSA	Elliptic Curve Digital Signature Algorithm
HMAC	Hash-based Message Authentication Code
HSM	Hardware Security Module
IPSec	Internet Protocol Security
IV	Initialization Value
LMK	Local Master Key
MAC	Message Authentication Code
RNG	Random Number Generator
RSA	Asymmetric Signature Algorithm named after L. Adleman, R. Rivest and A. Shamir
SSH	Secure Shell

References

[BelNam]

M. Bellare, C. Namprempre: Authenticated Encryption, Relations among notions and analysis of the generic composition paradigm3, in Advances in Cryptology (ASIACRYPT 2000), Lecture Notes in Computer Science vol 1976, p. 531–545, Springer-Verlag Berlin 2000-10

[DBLHC]

Banking and LAN Host Commands, Cryptosec Dekaton, Revision: 12.9.4790, Realia Technologies, S.L., 2020-09

3 http://cseweb.ucsd.edu/~mihir/papers/oem.pdf

[DCF]

Container Format, Cryptosec Dekaton, Revision: 12.9.4812, Realia Technologies, S.L., 2020-10

[DKD]

Key Derivation, Cryptosec Dekaton, Revision: 12.9.4788, Realia Technologies, S.L., 2020-09

[DKT]

Key Transport (Internal), Cryptosec Dekaton, Revision: 12.9.4788, Realia Technologies, S.L., 2020-09

[FAQ]

Payment Card Industry (PCI) PTS HSM Security Requirements, Technical FAQs for use with Version 3.0[4], September 2020

[FIPS186]

Digital Signature Standard (DSS), Federal Information Processing Standards Publication, FIPS 186-4, NIST, 2013-07

[Kraw]

H. Krawczyk: The order of encryption and authentication for protecting communications (Or: how secure is SSL?), Cryptology ePrint Archive[5], Report 2001/045

[RFC2104]

H. Krawczyk, M. Bellare, R. Canetti: HMAC: Keyed-Hashing for Message Authentication, RFC 2104, IETF, 1997-02

[RFC5652]

R. Housley: Cryptographic Message Syntax (CMS), RFC 5652, IETF, 2009-09

[RFC8017]

K. Moriarty, (Ed.), B. Kaliski, J. Jonsson, A. Rusch: PKCS #1: RSA Cryptography Specifications Version 2.2, RFC 8017, IETF, 2016-11

[SP800-38B]

Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, NIST Special Publication 800-38B, National Institute of Standards and Technology, May 2005

[TR-31]

Interoperable Secure Key Exchange Key Block Specification, Accredited Standards Committee X9, American National Standards Institute, 2018-04

[4] http://www.pcisecuritystandards.org/documents/PTS_HSM_Technical_FAQs_v3_September_2020.pdf
[5] http://eprint.iacr.org/2001/045