ETSI TS 104 053-2 V1.1.1 (2024-07)

TETRA Air Interface Security, Algorithms Specification;
Part 2: TETRA Encryption Algorithms TEA Set B

Reference
DTS/TCCE-06212

Keywords
air interface, DMO, security, TETRA, V+D

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee TETRA and Critical Communications Evolution (TCCE).

The present document is part 2 of a multi-part deliverable covering the specifications of the TETRA standard encryption, authentication and key management algorithms, as identified below:

Part 1: "TETRA Encryption Algorithms Set A";

Part 2: "TETRA Encryption Algorithms TEA Set B";

Part 3: "TETRA and Authentication and Key Management Algorithms TAA1";

Part 4: "TETRA Authentication and Key Management Algorithms TAA2".

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies the Terrestrial Trunked Radio system (TETRA) set B encryption algorithms TEA 5, 6 and 7. These algorithms are designed to meet the requirements set out in the requirements specification for the Additional TETRA Encryption Algorithm Suite [i.2].

The TETRA Air interface security function provides mechanisms for confidentiality of control signalling and user speech and data at the air interface, authentication and key management mechanisms for the air interface and for the Inter-System Interface (ISI). TETRA Air Interface security mechanisms are described in the TETRA V+D security specification [1] and the TETRA Direct Mode security specification [2].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document:

[1] ETSI TS 100 392-7: "Terrestrial Trunked Radio (TETRA); Voice plus Data (V+D); Part 7: Security".

[2] ETSI TS 100 396-6: "Terrestrial Trunked Radio (TETRA); Direct Mode Operation (DMO); Part 6: Security".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Daemen, J. and Rijmen, V. (1999): "AES proposal: Rijndael', document version 2". Submission to NIST AES competition (1999).

[i.2] ETSI TCCE(21)000002r2: "Requirements Specification for the Additional TETRA Encryption Algorithm Suite".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

Cipher Key (CK): value that is used to determine the transformation of plain text to cipher text in a cryptographic algorithm

Initialization Vector (IV): sequence of symbols that randomize the KSG inside the encryption unit

key stream: pseudo random stream of symbols that is generated by a KSG for encipherment and decipherment

Key Stream Generator (KSG): cryptographic algorithm which produces a stream of binary digits, which can be used for encipherment and decipherment

NOTE: The initial state of the KSG is determined by the IV value.

Key Stream Segment (KSS): key stream of arbitrary length

LENGTH: required length of the key stream in bits

TEA set A: set of air interface encryption algorithms comprising TEA1, TEA2, TEA3 and TEA4

TEA set B: set of air interface encryption algorithms comprising TEA5, TEA6 and TEA7

TETRA algorithm: mathematical description of a cryptographic process used for either of the security processes authentication or encryption

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CK	Cipher Key
CKM	Mode Key
GF	Galois Field
IV	Initialization Vector
IVM	Mode IV
IVX	eXpanded IV
KSS	Key Stream Segment
ISI	Inter System Interface
KSG	Key Stream Generator

4 TEA encryption Set B Algorithm specifications

4.1 Input and Output Parameters

As specified in clause 8.3 of [i.2], the input parameters to the algorithm are:

- an initialization vector IV consisting of 80 bits IV[0], …, IV[79];
- a cipher key CK consisting of 192 bits CK[0], …, CK[191];

- the required length LENGTH of the key stream in bits. This can, according to [i.2], take any value from 1 up to 8 288. However, the design of the algorithm allows it to deliver, securely, a length of keystream up to 2^{40} bits, making it potentially suitable for future applications where an increased length of KSS output is required.

The corresponding output from the algorithm is then a key stream segment KSS consisting of LENGTH bits KSS[0], …, KSS[LENGTH-1].

5 TEA5 - Specification of the Algorithm

5.1 Introduction

In outline, the algorithm operates as follows:

- the 80 bits of initialization vector IV, considered as 10 elements of the Galois field GF (2^8), are mixed using a 10-stage linear recursion over GF(2^8) to give 24 bytes which form a 192-bit mixed initialization vector IVX;
- the cipher key CK and mixed initialization vector IVX are combined to produce a 192-bit Mode Key, CKM, and a 192-bit Mode IV, IVM;
- successive 256-bit blocks are formed as a concatenation Mode IV || 'T', 'E', 'A', '5' || counter, where the byte values 'T', 'E', 'A', '5' code the name of the algorithm in ASCII, and the 32-bit counter takes successive values 0, 1, …;
- these successive 256-blocks are encrypted using the variant of Rijndael [i.1] with parameters giving a block length of 256 bits and key length 192 bits. The Mode Key is used as the Rijndael key. The 256-bit blocks obtained as a result of these Rijndael encryptions are concatenated to form KSS; some bits will be discarded from the final ciphertext block if LENGTH is not exactly divisible by 256.

The algorithm is specified precisely in clauses 5.2 to 5.6.

5.2 TEA5 IV Expansion

The 80-bit IV is expanded to a 192-bit mixed IV, IVX, as follows:

- from the initialization vector bits IV[0], …, IV[79], form 10 bytes b[0], …, b[9], where $b[i] = 2^7\, IV[8\times i] + 2^6\, IV[8\times i+1] + \ldots + IV[8\times i+7]$, for i = 0, …, 9;
- for any bits B[0], …, B[7], the byte is identified as $2^7 B[0] + 2^6\, B[1] + \ldots + B[7]$ with the element $z^7 B[0] + z^6\, B[1] + \ldots + B[7]$ of GF(2^8), where z is a generator of GF(2^8) satisfying the Rijndael polynomial $x^8 + x^4 + x^3 + x + 1$ in GF(2)[x];
- for i = 10, …, 43, a byte b[i] is obtained from bytes b[i-1], …, b[i-10] according to the rule $b[i] = b[i-10] \oplus b[i-9] \oplus (z^7 + z^6 + z^4 + z^2 + z + 1)\, b[i-1]$. The byte $(z^7 + z^6 + z^4 + z^2 + z + 1)\, b[i-1]$ can be obtained from b[i-1] using the lookup table defined in clause 5.5;
- the 24 bytes b[20], b[21], …, b[43] contain the bits IVX[0], …, IVX[191], where $b[20+i] = 2^7\, IVX[8i] + 2^6\, IVX[8i + 1] + \ldots + IVX[8i + 7]$ for i = 0, …, 23.

This process is illustrated in figures 1 and 2 below.

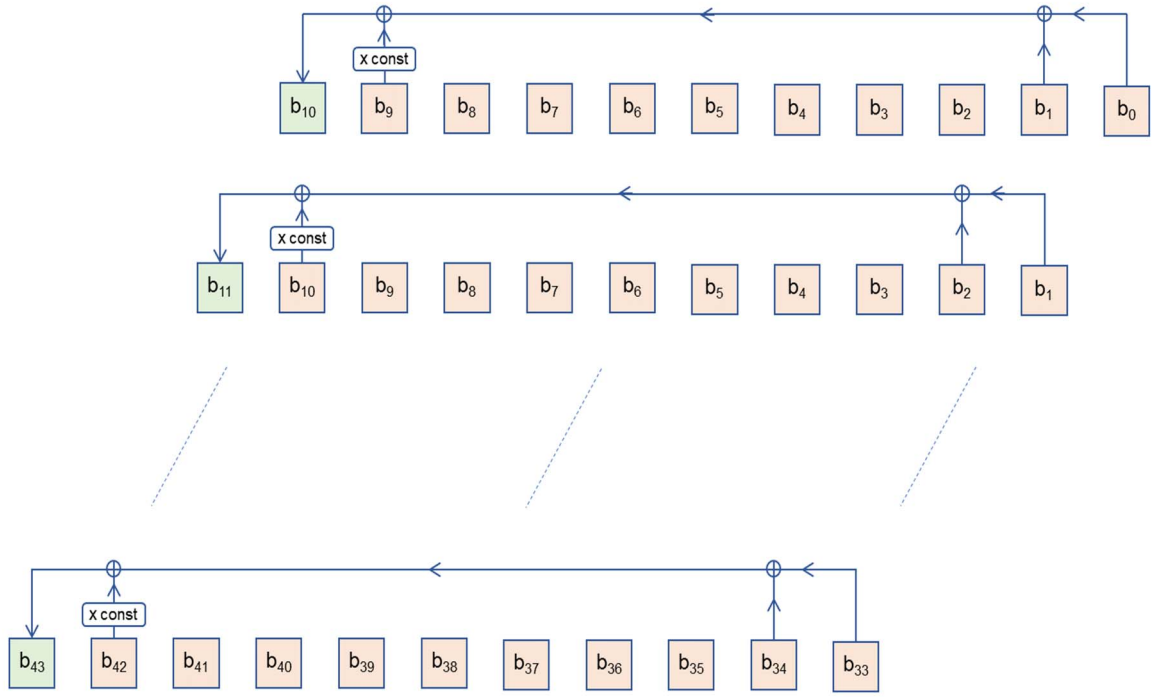

Figure 1: IV expansion

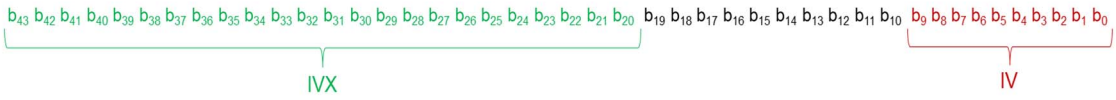

Figure 2: IVX extraction

5.3 TEA5 Derivation of the Mode Key and Mode IV

The Mode Key consists of 192 bits CKM[0], …, CKM[191] and the Mode IV consists of 192 bits IVM[0], …, IVM[191]. An 8-bit to 8-bit combining function f is applied to successive 8-bit inputs formed from 4 bits of the cipher key CK and 4 bits from the mixed initialization vector IVX, and the result is taken to be a further 4 bits of Mode Key and 4 bits of Mode IV. More precisely, for each i in the range 0, …, 47,

2^7CKM[4i] + 2^6CKM[4i+1] + … + 2^4CKM[4i+3] + 2^3IVM[4i] + 2^2IVM[4i+1] + … + IVM[4i+3] =

f(2^7CK[4i] + 2^6CK[4i+1] + … + 2^4CK[4i+3] + 2^3IVX[4i] + 2^2IVX[4i+1] + … + IVX[4i+3])

The combining function f is defined in clause 5.6.

The process for deriving the Mode Key and Mode IV from the cipher key CK and mixed initialization vector IVX is illustrated in figure 3 below.

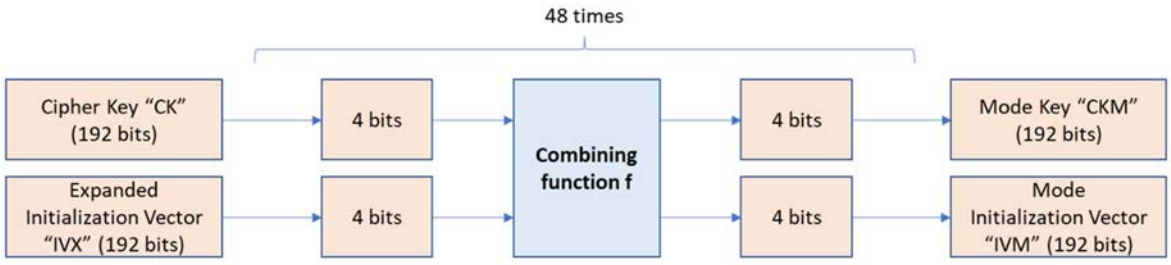

Figure 3: Mode Key and Mode IV derivation

5.4 TEA5 Derivation of the Keystream Bits

The keystream bits KSS are obtained using Rijndael [i.1] with key length 192 bits and block size 256 bits, used in a counter mode. The key used is the Mode Key CKM, arranged into bytes 2^7CKM[8i] + 2^6CKM[8i+1] + … + CKM[8i+7] for i = 0, …, 23. The Rijndael algorithm is run in encryption mode to encrypt $\lceil$ LENGTH/256 $\rceil$ successive plaintext blocks, where the notation $\lceil$ LENGTH/256 $\rceil$ denotes the least integer ≥ the floating-point quotient LENGTH/256. The plaintext for encryption j, for j = 0, …, $\lceil$ LENGTH/256 $\rceil$ -1, is, informally, IVM || 'T', 'E', 'A', '5' || j, where the byte values 'T', 'E', 'A', '5' code the name of the algorithm in ASCII, and j is coded as 4 bytes; more precisely, it is the 32-byte sequence p_0, …, p_{31}, where:

- $p_i = 2^7$IVM[8i] + 2^6IVM[8i+1] + … + IVM[8i+7] for i = 0, …, 23;
- $p_{24} = 84$, $p_{25} = 69$, $p_{26} = 65$, $p_{27} = 53$ (those four values being in decimal);
- $2^{24}p_{28} + 2^{16}p_{29} + 2^8p_{30} + p_{31} = j$.

The keystream bit KSS[i] is the bit $C_s[t]$, which is written as:

- i = 256r + 8s + t, for $0 \le s \le 31$ and $0 \le t \le 7$;
- c_0, …, c_{31} are the ciphertext bytes obtained from the encryption where j = r;
- $c_s = 2^7C_s[0] + 2^6C_s[1] + … + C_s[7]$, for bits $C_s[0]$, …, $C_s[7]$.

Note that if (LENGTH mod 256) ≤ 248 then one or more higher numbered ciphertext bytes from the last block will be discarded. If (LENGTH mod 8) > 0 then one or more less significant bits from the last used ciphertext byte will be discarded.

Note that the maximum value of LENGTH, the number of bits of required keystream, is 8 288, according to the specification [i.2]. Since this maximum number of required keystream bits ≤ 2^{16}, the 32-bit counter j can be implemented as an 8-bit counter with the other three bytes fixed to zero. If, in a future application, the maximum number of bits of required keystream is no more than 2^{24} bits, then the 32-bit counter can be implemented as a 16-bit counter with the other two bytes fixed to zero. If a full range of values for the 32-bit counter is implemented, keystream sequences of length up to 2^{40} can be generated.

The use of Rijndael in counter mode to produce keystream bits is shown in figure 4 below.

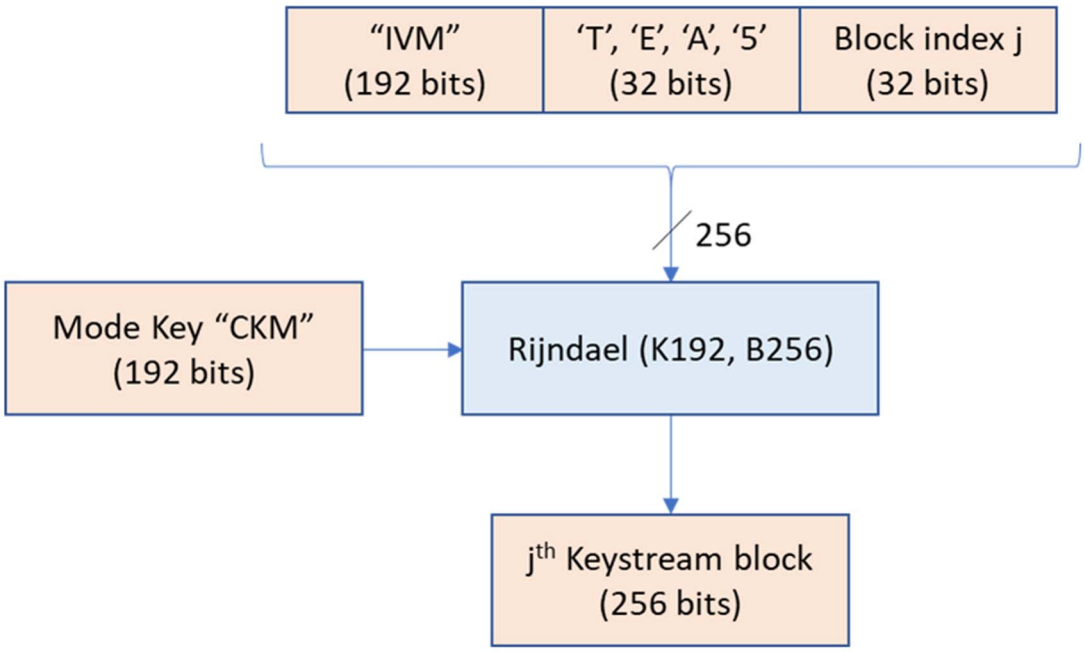

Figure 4: Keystream generation

5.5 TEA5 - Lookup Table for IV Mixing

Table 1 implements Galois Field multiplication by $z^7 + z^6 + z^4 + z^2 + z + 1$, as discussed in clause 5.2. Different rows correspond to different values of the most significant 4 bits of the input, and columns to the least significant 4 bits. For example, the value corresponding to 0x12 is found in the row labelled 0x1? and column labelled 0x?2, and is the byte value 0x6a.

Table 1

	0x?0	**0x?1**	**0x?2**	**0x?3**	**0x?4**	**0x?5**	**0x?6**	**0x?7**	**0x?8**	**0x?9**	**0x?a**	**0x?b**	**0x?c**	**0x?d**	**0x?e**	**0x?f**
0x0?	0x00	0xd7	0xb5	0x62	0x71	0xa6	0xc4	0x13	0xe2	0x35	0x57	0x80	0x93	0x44	0x26	0xf1
0x1?	0xdf	0x08	0x6a	0xbd	0xae	0x79	0x1b	0xcc	0x3d	0xea	0x88	0x5f	0x4c	0x9b	0xf9	0x2e
0x2?	0xa5	0x72	0x10	0xc7	0xd4	0x03	0x61	0xb6	0x47	0x90	0xf2	0x25	0x36	0xe1	0x83	0x54
0x3?	0x7a	0xad	0xcf	0x18	0x0b	0xdc	0xbe	0x69	0x98	0x4f	0x2d	0xfa	0xe9	0x3e	0x5c	0x8b
0x4?	0x51	0x86	0xe4	0x33	0x20	0xf7	0x95	0x42	0xb3	0x64	0x06	0xd1	0xc2	0x15	0x77	0xa0
0x5?	0x8e	0x59	0x3b	0xec	0xff	0x28	0x4a	0x9d	0x6c	0xbb	0xd9	0x0e	0x1d	0xca	0xa8	0x7f
0x6?	0xf4	0x23	0x41	0x96	0x85	0x52	0x30	0xe7	0x16	0xc1	0xa3	0x74	0x67	0xb0	0xd2	0x05
0x7?	0x2b	0xfc	0x9e	0x49	0x5a	0x8d	0xef	0x38	0xc9	0x1e	0x7c	0xab	0xb8	0x6f	0x0d	0xda
0x8?	0xa2	0x75	0x17	0xc0	0xd3	0x04	0x66	0xb1	0x40	0x97	0xf5	0x22	0x31	0xe6	0x84	0x53
0x9?	0x7d	0xaa	0xc8	0x1f	0x0c	0xdb	0xb9	0x6e	0x9f	0x48	0x2a	0xfd	0xee	0x39	0x5b	0x8c
0xa?	0x07	0xd0	0xb2	0x65	0x76	0xa1	0xc3	0x14	0xe5	0x32	0x50	0x87	0x94	0x43	0x21	0xf6
0xb?	0xd8	0x0f	0x6d	0xba	0xa9	0x7e	0x1c	0xcb	0x3a	0xed	0x8f	0x58	0x4b	0x9c	0xfe	0x29
0xc?	0xf3	0x24	0x46	0x91	0x82	0x55	0x37	0xe0	0x11	0xc6	0xa4	0x73	0x60	0xb7	0xd5	0x02
0xd?	0x2c	0xfb	0x99	0x4e	0x5d	0x8a	0xe8	0x3f	0xce	0x19	0x7b	0xac	0xbf	0x68	0x0a	0xdd
0xe?	0x56	0x81	0xe3	0x34	0x27	0xf0	0x92	0x45	0xb4	0x63	0x01	0xd6	0xc5	0x12	0x70	0xa7
0xf?	0x89	0x5e	0x3c	0xeb	0xf8	0x2f	0x4d	0x9a	0x6b	0xbc	0xde	0x09	0x1a	0xcd	0xaf	0x78

5.6 TEA5 - Definition of the Combining Function f

Table 2 defines the combining function f, which is used as defined in clause 5.3. Different rows correspond to different values of the most significant 4 bits of the input, and columns to the least significant 4 bits. For example, the value corresponding to 0x12 is found in the row labelled 0x1? and column labelled 0x?2, and is the byte value 0xcc.

Table 2

	0x?0	**0x?1**	**0x?2**	**0x?3**	**0x?4**	**0x?5**	**0x?6**	**0x?7**	**0x?8**	**0x?9**	**0x?a**	**0x?b**	**0x?c**	**0x?d**	**0x?e**	**0x?f**
0x0?	0x91	0x64	0x2c	0xc3	0x80	0xd8	0x32	0x5e	0x16	0xe7	0x09	0xbd	0x4f	0xa5	0xfa	0x7b
0x1?	0xbb	0x41	0xcc	0x67	0x36	0xe3	0x7d	0xa9	0x8e	0x52	0xf0	0xd4	0x28	0x1f	0x9a	0x05
0x2?	0xaf	0x92	0x78	0x33	0x4e	0xb6	0x8d	0xc7	0xd5	0xf9	0x11	0x60	0xec	0x04	0x5a	0x2b
0x3?	0x7c	0xd1	0x6f	0x57	0xa6	0x10	0xb9	0x25	0x43	0x0d	0x3b	0x9e	0xf8	0xe4	0x82	0xca
0x4?	0x5c	0x8a	0xe9	0x0e	0xb8	0xa2	0x66	0xf3	0x34	0x15	0x70	0x47	0x9f	0xcd	0x21	0xdb
0x5?	0x4c	0xb5	0xf1	0xe2	0x7f	0xce	0x90	0x1a	0x63	0x88	0xd6	0x2d	0x07	0x39	0xab	0x54
0x6?	0x2f	0x1d	0x89	0xf6	0xe1	0x0c	0xae	0xb3	0x97	0x45	0xc8	0x3a	0x74	0x50	0xd2	0x6b
0x7?	0x3e	0x01	0xdd	0x20	0xcf	0x62	0x1c	0xe8	0xba	0x76	0x55	0xa3	0x87	0x99	0x44	0xfb
0x8?	0x8f	0xee	0x13	0x7a	0xf5	0x49	0xc0	0xd7	0x08	0x3d	0xa4	0x5b	0x61	0x26	0xb2	0x9c
0x9?	0xdf	0x3c	0xa8	0x94	0x27	0x73	0x0a	0x8b	0x51	0xc9	0x65	0xe6	0xb0	0xfd	0x1e	0x42
0xa?	0x1b	0xfe	0x37	0xa1	0xd0	0x23	0xea	0x9d	0x72	0x6c	0xbf	0xc4	0x59	0x48	0x06	0x85
0xb?	0xe5	0x24	0x98	0xd3	0x5d	0x81	0xfc	0x69	0xc6	0xa0	0x4a	0x0b	0x12	0xb7	0x7e	0x3f
0xc?	0xff	0xc2	0x00	0x84	0x93	0x58	0x46	0x75	0xa7	0x2e	0xeb	0x19	0xda	0x6d	0x31	0xbc
0xd?	0x0f	0x79	0x56	0x40	0x14	0xf7	0x22	0x35	0xed	0xbe	0x9b	0x83	0xc1	0xdc	0x68	0xaa
0xe?	0xc5	0xad	0x4b	0xb1	0x6e	0x96	0x53	0x02	0x2a	0xd9	0x8c	0xf4	0x30	0x77	0xef	0x18
0xf?	0x6a	0x5f	0xb4	0x17	0x03	0x38	0xde	0x4d	0xf2	0x95	0x29	0x71	0xac	0x86	0xcb	0xe0

6 TEA6 - Specification of the Algorithm

6.1 Introduction

In outline, the algorithm operates as follows:

- the 80 bits of initialization vector IV, considered as 10 elements of the Galois field GF(2^8), are mixed using a 10-stage linear recursion over GF(2^8) to give 24 bytes which form a 192-bit mixed initialization vector IVX;
- the cipher key CK and mixed initialization vector IVX are combined to produce a 192-bit Mode Key, CKM, and a 192-bit Mode IV, IVM;
- successive 256-bit blocks are formed as a concatenation Mode IV || 'T', 'E', 'A', '6' || counter, where the byte values 'T', 'E', 'A', '6' code the name of the algorithm in ASCII, and the 32-bit counter takes successive values 0, 1, …;
- these successive 256-blocks are encrypted using the variant of Rijndael [i.1] with parameters giving a block length of 256 bits and key length 192 bits. The Mode Key is used as the Rijndael key. The 256-bit blocks obtained as a result of these Rijndael encryptions are concatenated to form KSS; some bits will be discarded from the final ciphertext block if LENGTH is not exactly divisible by 256.

The algorithm is specified precisely in clauses 6.2 to 6.6.

6.2 TEA6 - IV Expansion

The 80-bit IV is expanded to a 192-bit mixed IV, IVX, as follows:

- from the initialization vector bits IV[0], …, IV[79], form 10 bytes b[0], …, b[9], where b[i] = 2^7 IV[8×i] + 2^6 IV[8×i+1] + … + IV[8×i+7], for i = 0, …, 9;
- for any bits B[0], …, B[7], the byte is identified as 2^7B[0] + 2^6 B[1] + … + B[7] with the element z^7B[0] + z^6 B[1] + … + B[7] of GF(2^8), where z is a generator of GF(2^8) satisfying the Rijndael polynomial $x^8 + x^4 + x^3 + x + 1$ in GF(2)[x];
- for i = 10, …, 43, a byte b[i] is obtained from bytes b[i-1], …, b[i-10] according to the rule b[i] = b[i-10] ⊕ b[i-9] ⊕ ($z^7 + z^6 + z^4 + z^2 + z + 1$) b[i-1]. The byte ($z^7 + z^6 + z^4 + z^2 + z + 1$) b[i-1] can be obtained from b[i-1] using the lookup table defined in clause 6.5;
- the 24 bytes b[20], b[21], …, b[43] contain the bits IVX[0], …, IVX[191], where b[20+i] = 2^7 IVX[8i] + 2^6 IVX[8i + 1] + … + IVX[8i + 7] for i = 0, …, 23.

This process is illustrated in figures 5 and 6 below.

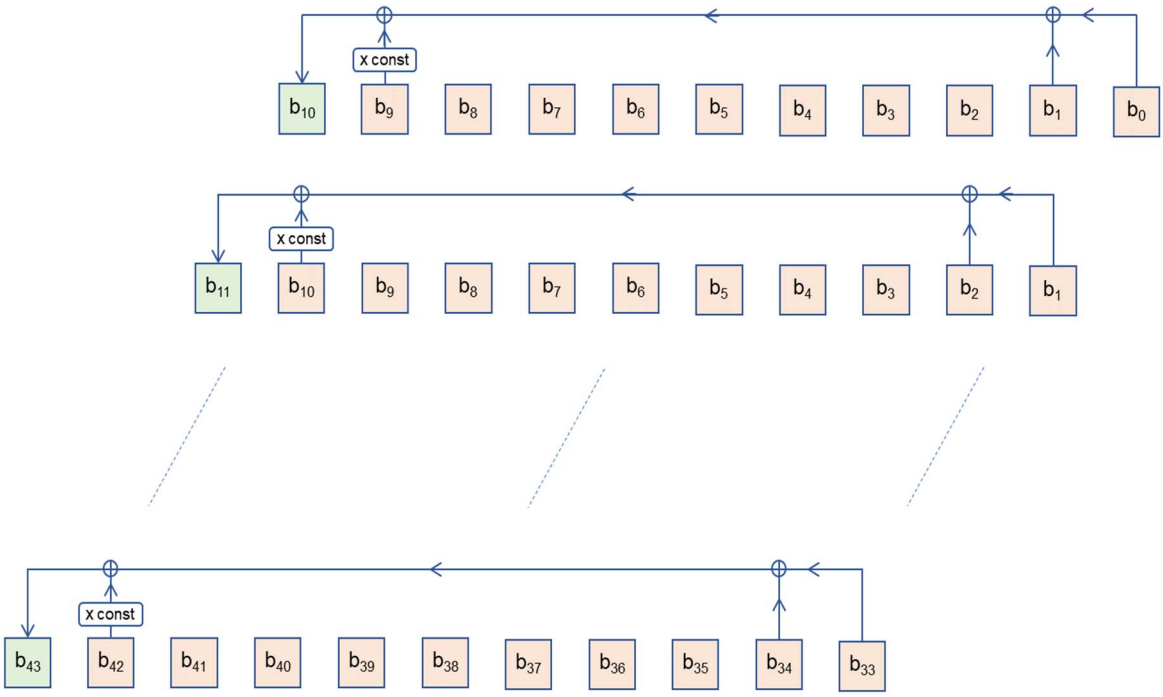

Figure 5: TEA6 - IV expansion

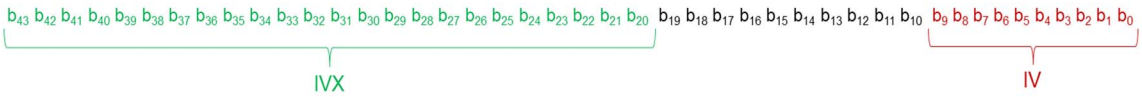

Figure 6: TEA6 - IVX extraction

6.3 TEA6 - Derivation of the Mode Key and Mode IV

The Mode Key consists of 192 bits CKM[0], …, CKM[191] and the Mode IV consists of 192 bits IVM[0], …, IVM[191]. An 8-bit to 8-bit combining function f is applied to successive 8-bit inputs formed from 4 bits of the cipher key CK and 4 bits from the mixed initialization vector IVX, and the result is taken to be a further 4 bits of Mode Key and 4 bits of Mode IV. More precisely, for each i in the range 0, …, 47,

$2^7CKM[4i] + 2^6CKM[4i+1] + \ldots + 2^4CKM[4i+3] + 2^3IVM[4i] + 2^2IVM[4i+1] + \ldots + IVM[4i+3] =$

$f(2^7CK[4i] + 2^6CK[4i+1] + \ldots + 2^4CK[4i+3] + 2^3IVX[4i] + 2^2IVX[4i+1] + \ldots + IVX[4i+3])$

The combining function f is defined in clause 6.6.

The process for deriving the Mode Key and Mode IV from the cipher key CK and mixed initialization vector IVX is illustrated in figure 7 below.

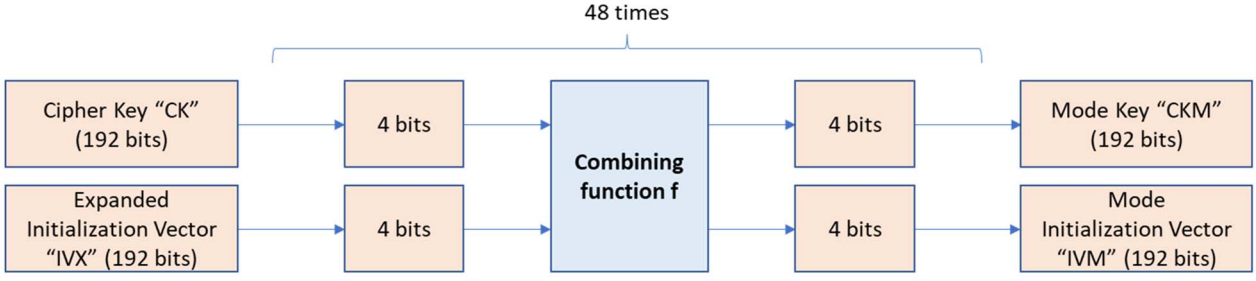

Figure 7: TEA6 Mode Key and Mode IV derivation

6.4 TEA6 - Derivation of the Keystream Bits

The keystream bits KSS are obtained using Rijndael [i.1] with key length 192 bits and block size 256 bits, used in a counter mode. The key used is the Mode Key CKM, arranged into bytes 2^7CKM[8i] + 2^6CKM[8i+1] + … + CKM[8i+7] for i = 0, …, 23. The Rijndael algorithm is run in encryption mode to encrypt $\lceil$ LENGTH/256 $\rceil$ successive plaintext blocks, where the notation $\lceil$ LENGTH/256 $\rceil$ denotes the least integer ≥ the floating-point quotient LENGTH/256. The plaintext for encryption j, for j = 0, …, $\lceil$ LENGTH/256 $\rceil$ -1, is, informally, IVM || 'T', 'E', 'A', '6' || j, where the byte values 'T', 'E', 'A', '6' code the name of the algorithm in ASCII, and j is coded as 4 bytes; more precisely, it is the 32-byte sequence p_0, …, p_{31}, where:

- $p_i = 2^7$IVM[8i] + 2^6IVM[8i+1] + … + IVM[8i+7] for i = 0, …, 23;
- p_{24} = 84, p_{25} = 69, p_{26} = 65, p_{27} = 53 (those four values being in decimal);
- $2^{24}p_{28} + 2^{16}p_{29} + 2^8p_{30} + p_{31}$ = j.

The keystream bit KSS[i] is the bit C_s[t], where:

- it is written as i = 256r + 8s + t, for 0 ≤ s ≤ 31 and 0 ≤ t ≤ 7;
- c_0, …, c_{31} are the ciphertext bytes obtained from the encryption where j = r;
- $c_s = 2^7C_s[0] + 2^6C_s[1] + … + C_s[7]$, for bits $C_s[0]$, …, $C_s[7]$.

Note that if (LENGTH mod 256) ≤ 248 then one or more higher numbered ciphertext bytes from the last block will be discarded. If (LENGTH mod 8) > 0 then one or more less significant bits from the last used ciphertext byte will be discarded.

Note that the maximum value of LENGTH, the number of bits of required keystream, is 8 288, according to the specification [i.2]. Since this maximum number of required keystream bits ≤ 2^{16}, the 32-bit counter j can be implemented as an 8-bit counter with the other three bytes fixed to zero. If, in a future application, the maximum number of bits of required keystream is no more than 2^{24} bits, then the 32-bit counter can be implemented as a 16-bit counter with the other two bytes fixed to zero. If a full range of values for the 32-bit counter is implemented, keystream sequences of length up to 2^{40} can be generated.

The use of Rijndael in counter mode to produce keystream bits is shown in figure 8 below.

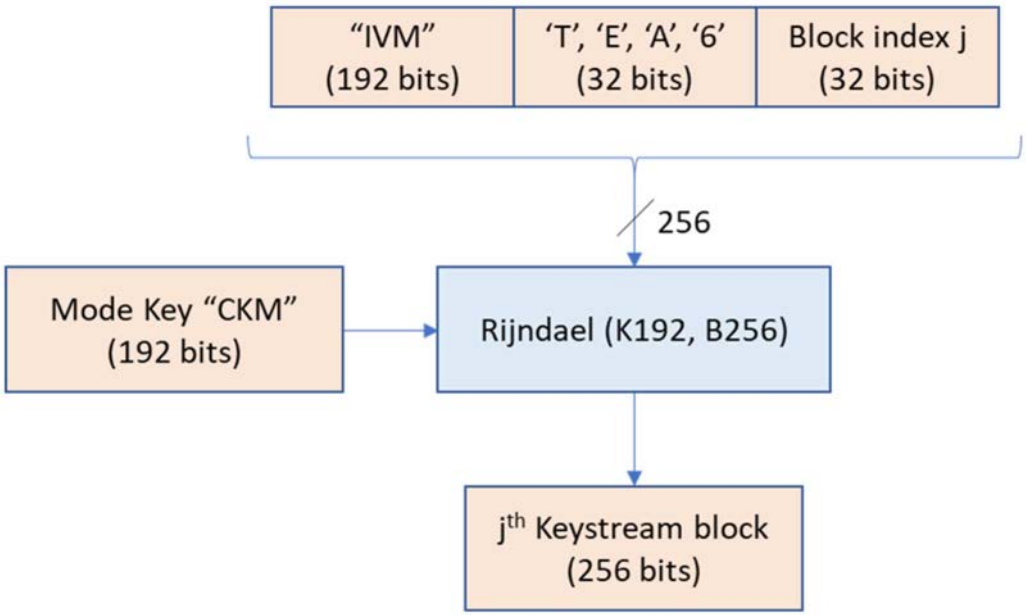

Figure 8: TEA6 Keystream generation

6.5 TEA6 - Lookup Table for IV Mixing

Table 3 implements Galois Field multiplication by $z^7 + z^6 + z^4 + z^2 + z + 1$, as discussed in clause 6.2. Different rows correspond to different values of the most significant 4 bits of the input, and columns to the least significant 4 bits. For example, the value corresponding to 0x12 is found in the row labelled 0x1? and column labelled 0x?2, and is the byte value 0x6a.

Table 3

	0x?0	**0x?1**	**0x?2**	**0x?3**	**0x?4**	**0x?5**	**0x?6**	**0x?7**	**0x?8**	**0x?9**	**0x?a**	**0x?b**	**0x?c**	**0x?d**	**0x?e**	**0x?f**
0x0?	0x00	0xd7	0xb5	0x62	0x71	0xa6	0xc4	0x13	0xe2	0x35	0x57	0x80	0x93	0x44	0x26	0xf1
0x1?	0xdf	0x08	0x6a	0xbd	0xae	0x79	0x1b	0xcc	0x3d	0xea	0x88	0x5f	0x4c	0x9b	0xf9	0x2e
0x2?	0xa5	0x72	0x10	0xc7	0xd4	0x03	0x61	0xb6	0x47	0x90	0xf2	0x25	0x36	0xe1	0x83	0x54
0x3?	0x7a	0xad	0xcf	0x18	0x0b	0xdc	0xbe	0x69	0x98	0x4f	0x2d	0xfa	0xe9	0x3e	0x5c	0x8b
0x4?	0x51	0x86	0xe4	0x33	0x20	0xf7	0x95	0x42	0xb3	0x64	0x06	0xd1	0xc2	0x15	0x77	0xa0
0x5?	0x8e	0x59	0x3b	0xec	0xff	0x28	0x4a	0x9d	0x6c	0xbb	0xd9	0x0e	0x1d	0xca	0xa8	0x7f
0x6?	0xf4	0x23	0x41	0x96	0x85	0x52	0x30	0xe7	0x16	0xc1	0xa3	0x74	0x67	0xb0	0xd2	0x05
0x7?	0x2b	0xfc	0x9e	0x49	0x5a	0x8d	0xef	0x38	0xc9	0x1e	0x7c	0xab	0xb8	0x6f	0x0d	0xda
0x8?	0xa2	0x75	0x17	0xc0	0xd3	0x04	0x66	0xb1	0x40	0x97	0xf5	0x22	0x31	0xe6	0x84	0x53
0x9?	0x7d	0xaa	0xc8	0x1f	0x0c	0xdb	0xb9	0x6e	0x9f	0x48	0x2a	0xfd	0xee	0x39	0x5b	0x8c
0xa?	0x07	0xd0	0xb2	0x65	0x76	0xa1	0xc3	0x14	0xe5	0x32	0x50	0x87	0x94	0x43	0x21	0xf6
0xb?	0xd8	0x0f	0x6d	0xba	0xa9	0x7e	0x1c	0xcb	0x3a	0xed	0x8f	0x58	0x4b	0x9c	0xfe	0x29
0xc?	0xf3	0x24	0x46	0x91	0x82	0x55	0x37	0xe0	0x11	0xc6	0xa4	0x73	0x60	0xb7	0xd5	0x02
0xd?	0x2c	0xfb	0x99	0x4e	0x5d	0x8a	0xe8	0x3f	0xce	0x19	0x7b	0xac	0xbf	0x68	0x0a	0xdd
0xe?	0x56	0x81	0xe3	0x34	0x27	0xf0	0x92	0x45	0xb4	0x63	0x01	0xd6	0xc5	0x12	0x70	0xa7
0xf?	0x89	0x5e	0x3c	0xeb	0xf8	0x2f	0x4d	0x9a	0x6b	0xbc	0xde	0x09	0x1a	0xcd	0xaf	0x78

6.6 TEA6 - Definition of the Combining Function f

Table 4 defines the combining function f, which is used as defined in clause 6.3. Different rows correspond to different values of the most significant 4 bits of the input, and columns to the least significant 4 bits. For example, the value corresponding to 0x12 is found in the row labelled 0x1? and column labelled 0x?2, and is the byte value 0x20.

Table 4

	0x?0	**0x?1**	**0x?2**	**0x?3**	**0x?4**	**0x?5**	**0x?6**	**0x?7**	**0x?8**	**0x?9**	**0x?a**	**0x?b**	**0x?c**	**0x?d**	**0x?e**	**0x?f**
0x0?	0xc3	0x56	0x32	0x09	0x70	0xd8	0xfc	0xed	0x2a	0xb7	0x1b	0x61	0xaf	0x84	0x9e	0x45
0x1?	0x91	0x3f	0x20	0xc6	0x65	0x73	0x49	0xae	0x0c	0x5b	0x88	0xba	0x12	0xc7	0xfd	0xd4
0x2?	0xef	0xc0	0x55	0x3e	0x42	0xb4	0x68	0x9c	0xa9	0x07	0xd3	0x2b	0xfa	0x1d	0x76	0x81
0x3?	0xb1	0x24	0xe9	0xdb	0xa6	0x80	0x95	0x5c	0x6d	0xf7	0x3a	0xc8	0x72	0x4f	0x13	0x0e
0x4?	0x14	0xb6	0x4b	0x90	0xdc	0xc2	0x78	0x2f	0xe3	0x8e	0x0a	0xa1	0x37	0xf5	0x5d	0x69
0x5?	0x62	0x9b	0xad	0xc4	0x5e	0x15	0xd7	0x00	0x8a	0x36	0xf9	0x71	0xbf	0x23	0x4c	0xe8
0x6?	0xab	0x7e	0x67	0x1a	0x85	0x21	0x38	0xbd	0xf0	0x44	0x5f	0x03	0x92	0xd9	0xec	0xc6
0x7?	0x43	0x0d	0xc5	0x89	0xff	0x60	0x52	0x11	0x34	0xe7	0xbe	0x9a	0xd6	0x7b	0x2c	0xa8
0x8?	0x01	0x64	0x9f	0x4a	0x30	0xfe	0x26	0x8b	0xce	0x18	0x77	0xdd	0xe2	0xb5	0xa3	0x59
0x9?	0x79	0xd2	0x8f	0xf6	0xee	0x93	0xb8	0x3c	0x41	0xcd	0xa0	0x17	0x6a	0x54	0x0b	0x25
0xa?	0x87	0xe1	0xd5	0x6e	0x22	0xa4	0x08	0xcb	0x1c	0x9d	0x46	0xf3	0x5a	0x39	0xb0	0x7f
0xb?	0xdf	0x82	0x7d	0x51	0xce	0x47	0x16	0xfb	0x99	0xaa	0x28	0xe0	0x04	0x6c	0x35	0xb3
0xc?	0x3d	0x10	0x02	0x7c	0x97	0x53	0xa5	0x48	0xb9	0x2e	0x6b	0x86	0xcf	0xe4	0xda	0xf1
0xd?	0xf4	0xac	0x1f	0x27	0xb2	0xeb	0xc9	0x7a	0x58	0xde	0x96	0x40	0x8d	0x05	0x63	0x31
0xe?	0x29	0xf2	0xbc	0xa7	0x06	0x3b	0x83	0x6f	0xd1	0x74	0xe5	0x50	0x4d	0x98	0xca	0x1e
0xf?	0x57	0x4e	0xf8	0xbb	0x19	0x0f	0xea	0xd0	0x75	0x66	0xc1	0x33	0x2d	0xa2	0x8c	0x94

7 TEA7-Specification of the Algorithm

7.1 Introduction

In outline, the algorithm operates as follows:

- the 80 bits of initialization vector IV, considered as 10 elements of the Galois field GF(2^8), are mixed using a 10-stage linear recursion over GF(2^8) to give 24 bytes which form a 192-bit mixed initialization vector IVX;
- the cipher key CK and mixed initialization vector IVX are combined to produce a 192-bit Mode Key, CKM, and a 192-bit Mode IV, IVM;
- successive 256-bit blocks are formed as a concatenation Mode IV || 'T', 'E', 'A', '7' || counter, where the byte values 'T', 'E', 'A', '7' code the name of the algorithm in ASCII, and the 32-bit counter takes successive values 0, 1, …;
- these successive 256-blocks are encrypted using the variant of Rijndael [i.1] with parameters giving a block length of 256 bits and key length 192 bits. The Mode Key is used as the Rijndael key. The 256-bit blocks obtained as a result of these Rijndael encryptions are concatenated to form KSS; some bits will be discarded from the final ciphertext block if LENGTH is not exactly divisible by 256.

The algorithm is specified precisely in clauses 7.2 to 7.6.

7.2 TEA7- IV Expansion

The 80-bit IV is expanded to a 192-bit mixed IV, IVX, as follows:

- from the initialization vector bits IV[0], …, IV[79], form 10 bytes b[0], …, b[9], where $b[i] = 2^7\ IV[8\times i] + 2^6\ IV[8\times i+1] + \ldots + IV[8\times i+7]$, for i = 0, …, 9;
- for any bits B[0], …, B[7], the byte is identified as $2^7B[0] + 2^6\ B[1] + \ldots + B[7]$ with the element $z^7B[0] + z^6\ B[1] + \ldots + B[7]$ of GF(2^8), where z is a generator of GF(2^8) satisfying the Rijndael polynomial $x^8 + x^4 + x^3 + x + 1$ in GF(2)[x];
- for i = 10, …, 43, a byte b[i] is obtained from bytes b[i-1], …, b[i-10] according to the rule $b[i] = b[i-10] \oplus b[i-9] \oplus (z^7 + z^6 + z^4 + z^2 + z + 1)\ b[i-1]$. The byte $(z^7 + z^6 + z^4 + z^2 + z + 1)\ b[i-1]$ can be obtained from b[i-1] using the lookup table defined in clause 7.5;
- the 24 bytes b[20], b[21], …, b[43] contain the bits IVX[0], …, IVX[191], where $b[20+i] = 2^7\ IVX[8i] + 2^6\ IVX[8i + 1] + \ldots + IVX[8i + 7]$ for i = 0, …, 23.

This process is illustrated in figures 9 and 10 below.

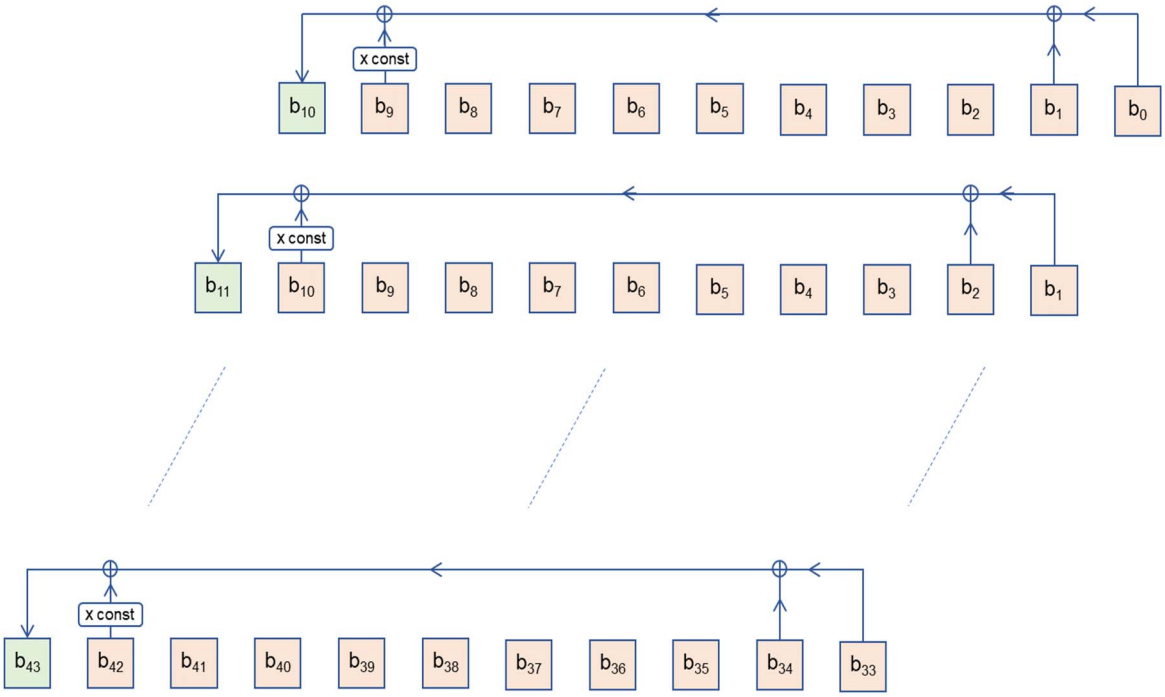

Figure 9: TEA7- IV expansion

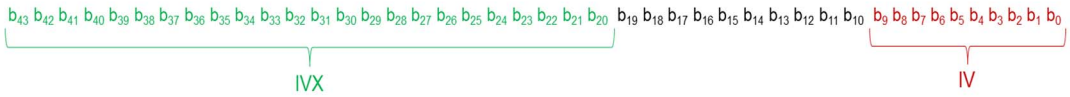

Figure 10: TEA7- IVX extraction

7.3 TEA7- Derivation of the Mode Key and Mode IV

The Mode Key consists of 192 bits CKM[0], …, CKM[191] and the Mode IV consists of 192 bits IVM[0], …, IVM[191]. An 8-bit to 8-bit combining function f is applied to successive 8-bit inputs formed from 4 bits of the cipher key CK and 4 bits from the mixed initialization vector IVX, and the result is taken to be a further 4 bits of Mode Key and 4 bits of Mode IV. More precisely, for each i in the range 0, …, 47,

2^7CKM[4i] + 2^6CKM[4i+1] + … + 2^4CKM[4i+3] + 2^3IVM[4i] + 2^2IVM[4i+1] + … + IVM[4i+3] =

f(2^7CK[4i] + 2^6CK[4i+1] + … + 2^4CK[4i+3] + 2^3IVX[4i] + 2^2IVX[4i+1] + … + IVX[4i+3])

The combining function f is defined in clause 7.5.

The process for deriving the Mode Key and Mode IV from the cipher key CK and mixed initialization vector IVX is illustrated in figure 11 below.

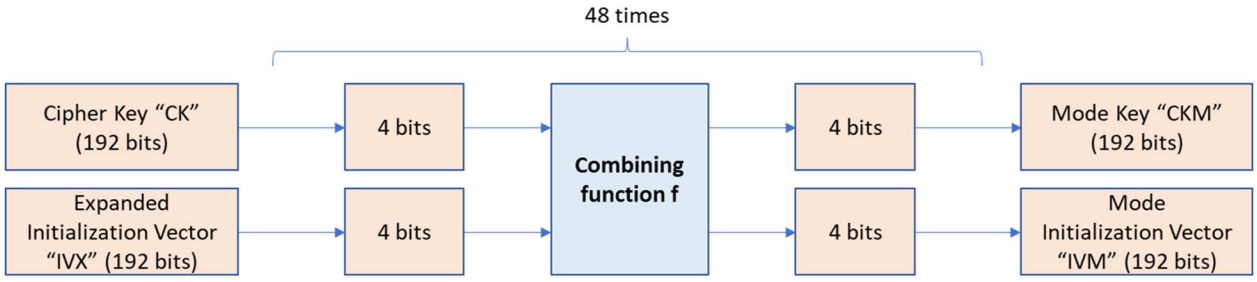

Figure 11: TEA7- Mode Key and Mode IV derivation

7.4 TEA7- Derivation of the Keystream Bits

The keystream bits KSS are obtained using Rijndael [i.1] with key length 192 bits and block size 256 bits, used in a counter mode. The key used is the Mode Key CKM, arranged into bytes $2^7CKM[8i] + 2^6CKM[8i+1] + \ldots + CKM[8i+7]$ for i = 0, …, 23. The Rijndael algorithm is run in encryption mode to encrypt $\lceil LENGTH/256 \rceil$ successive plaintext blocks, where the notation $\lceil LENGTH/256 \rceil$ denotes the least integer ≥ the floating-point quotient LENGTH/256. The plaintext for encryption j, for j = 0, …, $\lceil LENGTH/256 \rceil$ -1, is, informally, IVM || 'T', 'E', 'A', '7' || j, where the byte values 'T', 'E', 'A', '7' code the name of the algorithm in ASCII, and j is coded as 4 bytes; more precisely, it is the 32-byte sequence p0, …, p31, where:

- $p_i = 2^7IVM[8i] + 2^6IVM[8i+1] + \ldots + IVM[8i+7]$ for i = 0, …, 23;
- $p_{24} = 84$, $p_{25} = 69$, $p_{26} = 65$, $p_{27} = 55$ (those four values being in decimal);
- $2^{24}p_{28} + 2^{16}p_{29} + 2^8p_{30} + p_{31} = j$.

The keystream bit KSS[i] is the bit $C_5[t]$, where:

- it is written as $i = 256r + 8s + t$, for $0 \leq s \leq 31$ and $0 \leq t \leq 7$;
- $c_0, \ldots, c_{31}$ are the ciphertext bytes obtained from the encryption where j = r;
- $c_s = 2^7C_s[0] + 2^6C_s[1] + \ldots + C_s[7]$, for bits $C_s[0], \ldots, C_s[7]$.

Note that if (LENGTH mod 256) ≤ 248 then one or more higher numbered ciphertext bytes from the last block will be discarded. If (LENGTH mod 8) > 0 then one or more less significant bits from the last used ciphertext byte will be discarded.

Note that the maximum value of LENGTH, the number of bits of required keystream, is 8 288, according to the specification [i.2]. Since this maximum number of required keystream bits ≤ 2^{16}, the 32-bit counter j can be implemented as an 8-bit counter with the other three bytes fixed to zero. If, in a future application, the maximum number of bits of required keystream is no more than 2^{24} bits, then the 32-bit counter can be implemented as a 16-bit counter with the other two bytes fixed to zero. If a full range of values for the 32-bit counter is implemented, keystream sequences of length up to 2^{40} can be generated.

The use of Rijndael in counter mode to produce keystream bits is shown in figure 12 below.

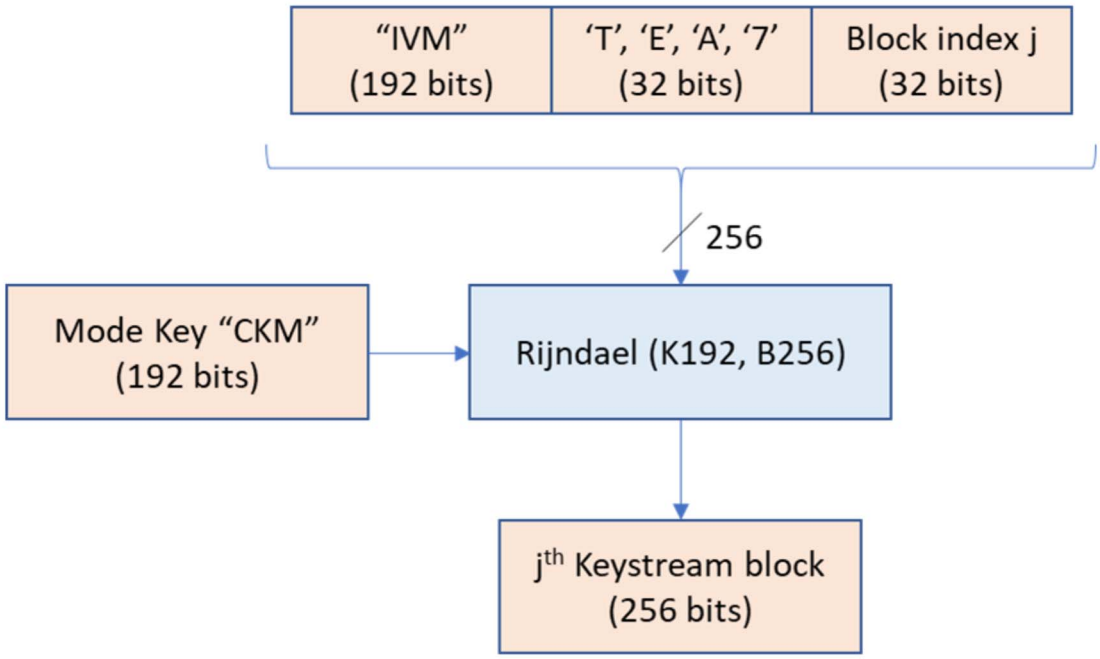

Figure 12: Keystream generation

7.5 TEA7 - Lookup Table for IV Mixing

Table 5 implements Galois Field multiplication by $z^7 + z^6 + z^4 + z^2 + z + 1$, as discussed in clause 7.2. Different rows correspond to different values of the most significant 4 bits of the input, and columns to the least significant 4 bits. For example, the value corresponding to 0x12 is found in the row labelled 0x1? and column labelled 0x?2, and is the byte value 0x6a.

Table 5

	0x?0	**0x?1**	**0x?2**	**0x?3**	**0x?4**	**0x?5**	**0x?6**	**0x?7**	**0x?8**	**0x?9**	**0x?a**	**0x?b**	**0x?c**	**0x?d**	**0x?e**	**0x?f**
0x0?	0x00	0xd7	0xb5	0x62	0x71	0xa6	0xc4	0x13	0xe2	0x35	0x57	0x80	0x93	0x44	0x26	0xf1
0x1?	0xdf	0x08	0x6a	0xbd	0xae	0x79	0x1b	0xcc	0x3d	0xea	0x88	0x5f	0x4c	0x9b	0xf9	0x2e
0x2?	0xa5	0x72	0x10	0xc7	0xd4	0x03	0x61	0xb6	0x47	0x90	0xf2	0x25	0x36	0xe1	0x83	0x54
0x3?	0x7a	0xad	0xcf	0x18	0x0b	0xdc	0xbe	0x69	0x98	0x4f	0x2d	0xfa	0xe9	0x3e	0x5c	0x8b
0x4?	0x51	0x86	0xe4	0x33	0x20	0xf7	0x95	0x42	0xb3	0x64	0x06	0xd1	0xc2	0x15	0x77	0xa0
0x5?	0x8e	0x59	0x3b	0xec	0xff	0x28	0x4a	0x9d	0x6c	0xbb	0xd9	0x0e	0x1d	0xca	0xa8	0x7f
0x6?	0xf4	0x23	0x41	0x96	0x85	0x52	0x30	0xe7	0x16	0xc1	0xa3	0x74	0x67	0xb0	0xd2	0x05
0x7?	0x2b	0xfc	0x9e	0x49	0x5a	0x8d	0xef	0x38	0xc9	0x1e	0x7c	0xab	0xb8	0x6f	0x0d	0xda
0x8?	0xa2	0x75	0x17	0xc0	0xd3	0x04	0x66	0xb1	0x40	0x97	0xf5	0x22	0x31	0xe6	0x84	0x53
0x9?	0x7d	0xaa	0xc8	0x1f	0x0c	0xdb	0xb9	0x6e	0x9f	0x48	0x2a	0xfd	0xee	0x39	0x5b	0x8c
0xa?	0x07	0xd0	0xb2	0x65	0x76	0xa1	0xc3	0x14	0xe5	0x32	0x50	0x87	0x94	0x43	0x21	0xf6
0xb?	0xd8	0x0f	0x6d	0xba	0xa9	0x7e	0x1c	0xcb	0x3a	0xed	0x8f	0x58	0x4b	0x9c	0xfe	0x29
0xc?	0xf3	0x24	0x46	0x91	0x82	0x55	0x37	0xe0	0x11	0xc6	0xa4	0x73	0x60	0xb7	0xd5	0x02
0xd?	0x2c	0xfb	0x99	0x4e	0x5d	0x8a	0xe8	0x3f	0xce	0x19	0x7b	0xac	0xbf	0x68	0x0a	0xdd
0xe?	0x56	0x81	0xe3	0x34	0x27	0xf0	0x92	0x45	0xb4	0x63	0x01	0xd6	0xc5	0x12	0x70	0xa7
0xf?	0x89	0x5e	0x3c	0xeb	0xf8	0x2f	0x4d	0x9a	0x6b	0xbc	0xde	0x09	0x1a	0xcd	0xaf	0x78

7.6 TEA7 - Definition of the Combining Function f

Table 6 defines the combining function f, which is used as defined in clause 7.3. Different rows correspond to different values of the most significant 4 bits of the input, and columns to the least significant 4 bits. For example, the value corresponding to 0x12 is found in the row labelled 0x1? and column labelled 0x?2, and is the byte value 0xcf.

Table 6

	0x?0	**0x?1**	**0x?2**	**0x?3**	**0x?4**	**0x?5**	**0x?6**	**0x?7**	**0x?8**	**0x?9**	**0x?a**	**0x?b**	**0x?c**	**0x?d**	**0x?e**	**0x?f**
0x0?	0xe8	0xf8	0xc0	0x7a	0x44	0x07	0x2b	0x9e	0x54	0x7e	0x64	0x67	0xd5	0x8a	0x38	0x04
0x1?	0x10	0xde	0xcf	0x71	0x42	0x2e	0xfd	0x95	0xe0	0x8b	0xa1	0xaf	0xb8	0x80	0x5f	0x19
0x2?	0x1e	0xf7	0x15	0x03	0xad	0x0c	0x29	0xc7	0x5b	0x8f	0x46	0xa7	0xb9	0x82	0x3d	0x1d
0x3?	0xef	0xd0	0x93	0x0e	0xa0	0x79	0xf9	0xca	0x51	0x7b	0x6b	0x6f	0xd1	0x3f	0x58	0xb2
0x4?	0x18	0xd3	0x9c	0x0d	0x9f	0x05	0xfc	0xc1	0xe4	0x8d	0x4a	0x60	0xb5	0x81	0x5e	0x1f
0x5?	0xe5	0xd9	0xc3	0x70	0x4e	0x22	0x24	0x9a	0xed	0x7d	0x68	0xf2	0xba	0x3a	0x5a	0x1c
0x6?	0xeb	0xf1	0x97	0x76	0x48	0x27	0xf4	0xcd	0x6c	0x7c	0x4b	0x65	0xdc	0x8c	0x53	0x17
0x7?	0xe3	0xfb	0x99	0x00	0x41	0x2c	0xf3	0x3b	0xe9	0x74	0x40	0xa5	0xd4	0x86	0x36	0xbb
0x8?	0x1b	0xea	0x91	0x0b	0xa3	0x26	0xff	0xc8	0xe6	0x89	0x62	0x63	0xbf	0x8e	0x32	0xb0
0x9?	0xee	0xd8	0x98	0x75	0xa8	0x28	0xfa	0xc4	0x55	0x72	0x4f	0xae	0x57	0x39	0x50	0xbc
0xa?	0xe1	0xf5	0xcb	0x7f	0x45	0x0a	0xf6	0x96	0x52	0x87	0x49	0x6a	0xdb	0x3e	0x20	0x14
0xb?	0x11	0xd2	0xc2	0x01	0xa4	0x0f	0x25	0x9b	0x5c	0x43	0x66	0x6e	0xb4	0x30	0x56	0xbe
0xc?	0xce	0xfe	0xcc	0x09	0xa6	0x2f	0x23	0x94	0x5d	0x84	0x47	0xac	0xda	0x3c	0x31	0xb7
0xd?	0x16	0xd7	0xc5	0x73	0xaa	0x02	0xb6	0x90	0xec	0x77	0x69	0x61	0xb1	0x33	0x34	0xb3
0xe?	0x12	0xd6	0xc9	0x08	0xab	0x06	0x2d	0x92	0xe2	0x83	0x6d	0xa2	0xbd	0xdd	0x37	0x1a
0xf?	0xe7	0xf0	0x9d	0x88	0x4d	0x21	0x2a	0xc6	0x59	0x78	0x4c	0xa9	0xdf	0x85	0x35	0x13

Annex A (informative): Bibliography

Barreto, P. and Rijmen, V. (2002): "Rijndael reference code in ANSI C", v2.2.

ETSI TS 101 053-5: "Rules for the management of the TETRA standard encryption algorithms; Part 5: TEA5".

ETSI TS 101 053-6: "Rules for the management of the TETRA standard encryption algorithms; Part 6: TEA6".

ETSI TS 101 053-7: "Rules for the management of the TETRA standard encryption algorithms; Part 7: TEA7".

History

Document history		
V1.1.1	July 2024	Publication