ETSI TS 104 053-1 V1.1.1 (2024-07)

TETRA Air Interface Security, Algorithms Specifications;
Part 1: TETRA Encryption Algorithms Set A

Reference
DTS/TCCE-06211

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 1 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 and 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 A encryption algorithms TEA1, TEA2, TEA3 and TEA4.

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.

Not applicable.

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

cipher text: data produced through the use of encipherment

decipherment: reversal of a corresponding reversible encipherment

encipherment: cryptographic transformation of data to produce cipher text

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

LENGTH: required length of the key stream in bits

plain text: un-encrypted source data

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
ISI	Inter Systems Interface
IV	Initialization Vector
KSG	Key Stream Generator
TC	Technical Committee
TCCE	TETRA and Critical Communications Evolution
TEA1	TETRA Encryption Algorithm No. 1
TEA2	TETRA Encryption Algorithm No. 2
TEA3	TETRA Encryption Algorithm No. 3
TEA4	TETRA Encryption Algorithm No. 4
TETRA	TErrestrial Trunked Radio

4 TEA SET A Specifications

The algorithms TEA1, 2, 3 and 4 specified in the present document generate a sequence of key bytes from a Cipher Key CK and an Initialization Vector IV.

The key bytes are used to encrypt or to decrypt information transmitted via the TETRA system.

The Cipher Key has a length of 80 bits; the Initialization Vector is 29 bits.

The present document is organized as follows: clause 5 describes TEA1, clause 6 TEA2, clause 7 TEA3 and clause 8 TEA4.

Clauses 5.1, 6.1, 7.1 and 8.1 provide a functional specification of the functional components of the algorithms, clauses 5.2, 6.2, 7.2 and 8.2 specify how each of these components are used to generate the key bytes and clauses 5.3, 6.3, 7.3 and 8.3 contain diagrams and tables for the respective algorithms.

5 TEA1 ALGORITHM DESCRIPTION

5.1 TEA1 Functional Components

5.1.1 Summary of Components

The cryptographic algorithm TEA1 consists of the following functional components as depicted in Figure 1:

- A set of eight 8-bit wide shift registers, called Output Register (clause 5.1.3).
- A set of four 8-bit wide shift registers, called Key Register (clause 5.1.4).
- A byte permutation function *P* (byte substitution) (clause 5.1.5).
- Two functions *E* to expand the 16-bit output of two 8-bit registers to 32 bits word (clause 5.1.6).
- Two nonlinear functions f_1 and f_2 to compute a byte from the expanded 32-bit word (clause 5.1.7).
- An 8-bit permutation function *BP* (wire crossing) (clause 5.1.8).
- Five 8-bit wide XOR functions to combine Section outputs (bytes). And
- Functions in the feedback of the key register and the output register.

5.1.2 Notation

The symbol $\oplus$ denotes the bytewise addition modulo two (exclusive or); i.e. msb of byte x is added to msb of byte y , ..., lsb of byte x to lsb of byte y.

x (i)	*y(i)*	*x(i) ⊕ y(i)*
0	0	0
0	1	1
1	0	1
1	1	0
x(i) is bit *i* of byte *x* *y(i)* is bit *i* of byte *y* i = *0,...,7*		

5.1.3 Output Register

The output register is denoted by R_0, R_1..., R_6, and R_7, and functions as an 8-bit wide shift register. The output bytes of sections R_1, R_2 and R_4 up to R_6 are also used as input for the other functional components of the TEA1. The output of R_7 is added to the outputs of the other functional components and returned to the first section R_0 as depicted in Figure 1.

Each 19 steps the byte output of R_7 is used as key byte for encryption or decryption.

Before the process as described above can take place the output register is loaded with an Initialization Vector (see clause 5.2.3) and initialized (see clause 5.2.4).

5.1.4 Key Register

The key register, denoted by K_0, K_1, K_2 and K_3, is a four section 8-bit wide shift register. The bytes of a Cipher Key (CK) are loaded into the key register as shown in Figure 2, and before the Initialization Vector is loaded into the Output Register. During loading the CK bytes are added modulo two to the result of the modulo two addition of the output of sections K_0 and K_3. This result is substituted by the permutation function P and passed to section K_0. The loading process starts from the reset state 0000 of the K_i register.

During initialization and the generation of key bytes the same feedback process, however without the CK_i input (= 0), is continued as follows:

with K'_i denoting the next byte value (i.e. after one step) of the register section K'_i.

$K'_0 = P\ (K_3 \oplus K_0)$

$K'_1 = K_0$

$K'_2 = K_1$

$K'_3 = K_2$

The series of mixed and permutated bytes is also offered to the feedback circuit of the output register.

5.1.5 Byte Permutation Function

The byte permutation function P is a randomly chosen permutation in the set of all 256 possible bytes. The look up table for this permutation is given in Figure 3. The higher nibble of the output is the left one of the output value, e.g. P (27) = 6A.

5.1.6 Expander E

The expander function converts a two-byte input to a 32-bit word, i.e. eight 4-bit nibbles for the nonlinear functions f_1 and f_2. The structure of expander E and functions f_1 and f_2 is shown in Figure 4.

In this figure, bits 1-8 are the bits of R_2, respectively R_6.

Bits 9-16 are the bits of R_1, respectively R_5. Bits 1 and 9 are the most significant bits. The 4-bit nibble inputs to the S_i boxes are the result of reading R_2 R_1 (for the input of f_1), respectively R_6 R_5 (for the input of f_2). The bit left is the msb of each 4-bit nibble input.

5.1.7 Nonlinear Function f_1

The nonlinear functions f_1 and f_2 have the structure shown in Figure 4. Each of the boxes S_1 to S_8 receives the nibble input concerned from the expander E and computes a bit for the output byte. S_1 is the most significant bit in the output byte, and S_8 is the least significant one.

The functions f_1 and f_2 are given in the truth tables, Figure 5 and Figure 6, respectively. The hexadecimal value of the input nibble for each S_i is the one given in the top row of these figures. The computed output bit for the corresponding S_i; can be read in the column below the input nibble row.

5.1.8 8-bit Permutation BP

The permutation BP is a so-called wire-crossing with a fixed pattern. If the eight bits of R_4 are numbered 12345678 the order of the bits after BP becomes 58417326. The left bit in both bytes is the most significant; the right bit is the least significant.

5.1.9 Feedback of output register

The results of the functional components BP, f_1, f_2 and CK byte permutation P are added in the feedback path of the output register and affect. the operation (i.e. operation after loading the CK and Initialization vector) as follows:

with R'_i denoting the next byte value (i.e. after one step) of the output register Section R_i:

$R'_o = R_7 \oplus f_2\ (R_6, R_5) \oplus BP\ (R_4) \oplus P_{out}$

$R'_1 = R_o$

$R'_2 = R_1$

$R'_3 = R_2$

$R'_4 = R_3 \oplus f_1 (R_2, R_1)$

$R'_5 = R_4$

$R'_6 = R_5$

$R'_7 = R_6$

5.2 Key Stream Generation

5.2.1 Summary

The algorithm consists of four main phases: the CK loading, the IV loading, the run-up and the key byte generation proper.

During the CK loading the initial state of the key register is determined. Next, the initial state of the output register is determined by loading of the Initialization Vector as described in clause 5.2.3. Thereafter, a run-up is carried out during which the initial contents of the Output Register (converted and adapted IV) and the K_i Register are changed by the effect of the *BP*, *E*, f_1, f_2 and *P* functions (see clause 5.2.4).

After the run-up is completed, each output cycle produces a key byte from the key byte stream as specified in clause 5.2.5. At any point during the run-up and the generation of the key byte stream, the state of the algorithm is determined by the states of the output and the K_i registers.

5.2.2 CK loading

The CK loading depends on the 80-bit Cipher Key, the feedback method and the permutation function *P*. The 10 CK bytes are loaded as depicted in clause 5.3, Figure 2. The reduced CK, i.e. the 32 bits available in the K_i register after the loading process, will affect the further initialization of the algorithm and the generation of key bytes.

5.2.3 IV loading

The output register is first loaded with a.29-bit Initialization Vector. This IV is converted to a 32-bit word by taking the three most significant bits as zeroes, and the remaining 29 bit as the given IV. For instance, if the running counter is the binary value:

11010 00011010 11100010 00000110

the 32-bit word becomes:

00011010 00011010 11100010 00000110.

The least significant byte of that 32-bit word is loaded into R_3, the next byte into R_4, the next one into R_5, and, finally, the most significant (the padded one) becomes the R_6 initial value. The cells R_0, R_1, R_2 and R_7 are loaded in the same order with the 32-bit word XORed with the constant mask 96724FA1. Then, we have the following initialization assignments, with the running counter as a four-byte number $F_1F_2F_3F_4$:

$R_7 = F_1 \oplus 96$

$R_6 = F_1$

$R_5 = F_2$

$R_4 = F_3$

$R_3 = F_4$

$R_2 = F_2 \oplus 72$

$R_1 = F_3 \oplus 4F$

$R_0 = F_4 \oplus Al$

Using the IV, mentioned in the example above, the result is:

1 8

10001 100 00011010 00011010

00000110 01101000 10100111

for the R_7, ..., R_4 and R_3, ..., R_0 bytes.

The eight bits of each R; numbered from 1 to 8 are loaded into the register locations:

R_i (7), R_i (6), ..., R_i (0) respectively.

5.2.4 Run-up

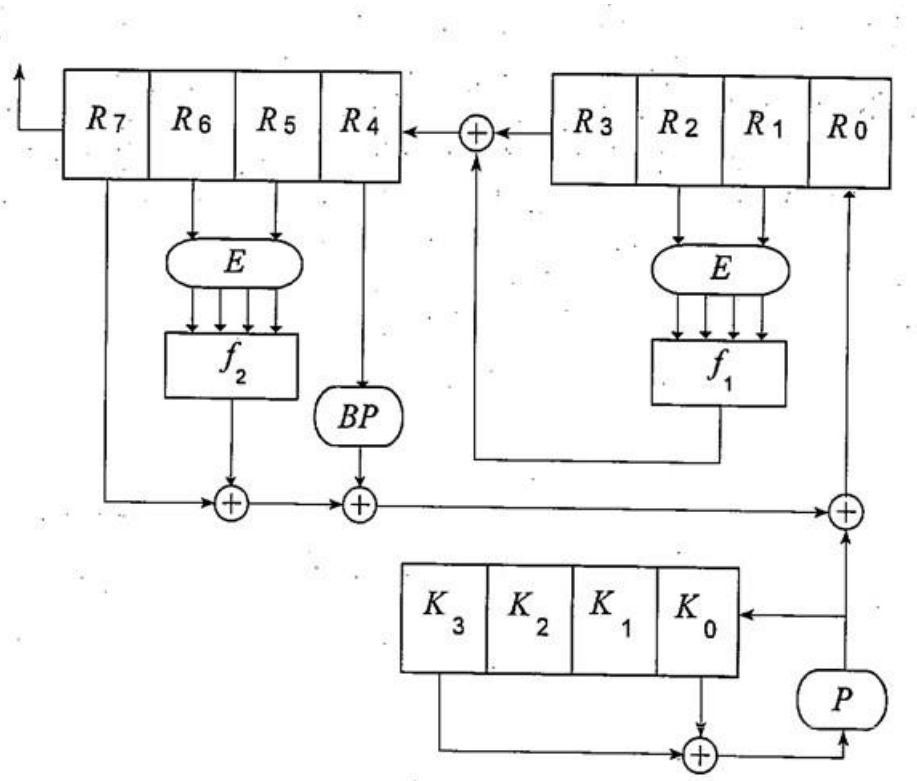
After the IV load into the output register all functional components of the TEA1 become operational and 53 initializing steps are performed to bring the algorithm in the CK and IV dependent starting point from which the generation of key bytes can start.

For run-up and key byte generation a step is defined as applying the formulae in clauses 5.1.4 and 5.1.9 once.

5.2.5 Key byte generation

After the run-up cycle one step is made to produce the first key byte. Successive key bytes are generated each 19 steps.

5.3 Figures of TEA1 Algorithm



$R_0 \ldots R_7$	Output Register (eight 8-bit wide registers)
$K_0 \ldots K_3$	Crypto Key load register (four 8-bit wide registers)
E	16 to 32 bits expansion
f_1, f_2	Nonlinear function: each eight functions of 4 bits
BP	Permutation of 8 bits (wire crossing)
P	Permutation on 2^8 elements (byte substitution)

Figure 1: Functional components of TEA1

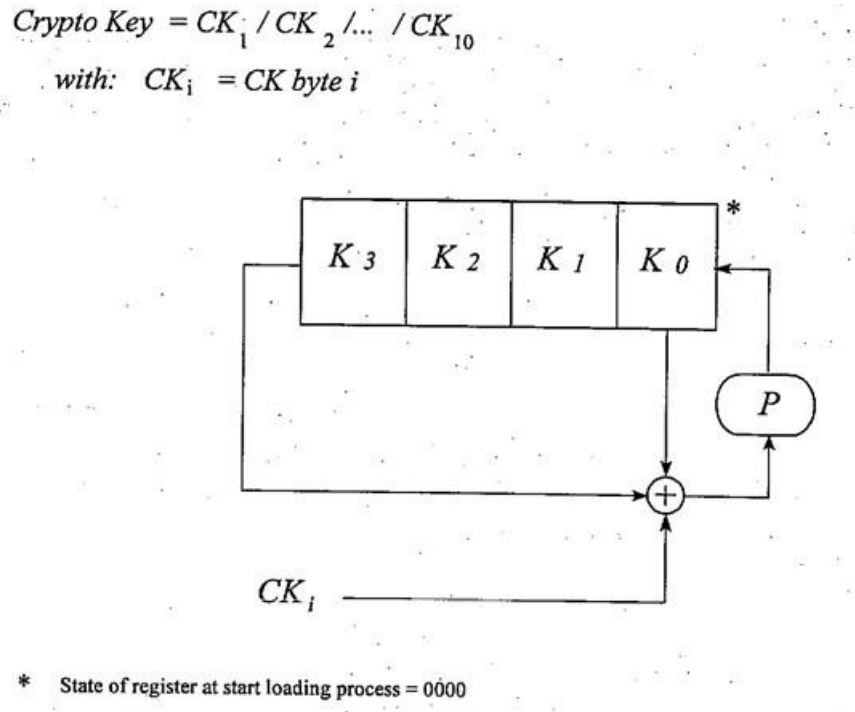


Figure 2: Cipher Key Load Map

High nibble	Lower Nibble Input 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9B	F8	3B	72	75	62	88	22	FF	A6	10	4D	A9	97	C3	7B
1	9F	78	F3	B6	A0	CC	17	AB	4A	41	8D	89	25	87	D3	E3
2	CE	47	35	2C	6D	FC	E7	6A	B8	B7	FA	8B	CD	74	EE	11
3	23	DE	39	6C	1E	8E	ED	30	73	BE	BB	91	CA	69	60	49
4	5F	B9	C0	06	34	2A	63	4B	90	28	AC	50	E4	6F	36	B0
5	A4	D2	D4	96	D5	C9	66	45	C5	55	DD	B2	A1	A8	BF	37
6	32	2B	3E	B5	5C	54	67	92	56	4C	20	6B	42	9D	A7	58
7	0E	52	68	95	09	7F	59	9C	65	B1	64	5E	4F	BA	81	1C
8	C2	0C	02	B4	31	5B	FD	1D	0A	C8	19	8F	83	8A	CF	33
9	9E	3A	80	F2	F9	76	26	44	F1	E2	C4	F5	D6	51	46	07
A	14	61	F4	C1	24	7A	94	27	00	FB	04	DF	1F	93	71	53
B	EA	D8	BD	3D	D0	79	E6	7E	4E	9A	D7	98	1B	05	AE	03
C	C7	BC	86	DB	84	E8	D1	F7	16	21	6E	E5	CB	A3	1A	EC
D	A2	7D	18	85	48	DA	AA	F0	08	C6	40	AD	57	0D	29	82
E	7C	E9	8C	FE	DC	0F	2D	3C	2E	F6	15	2F	AF	E1	EB	3F
F	99	43	13	0B	E0	A5	12	77	5D	B3	38	D9	EF	5A	01	70

Left = higher nibble

Figure 3: Byte permutation lookup table

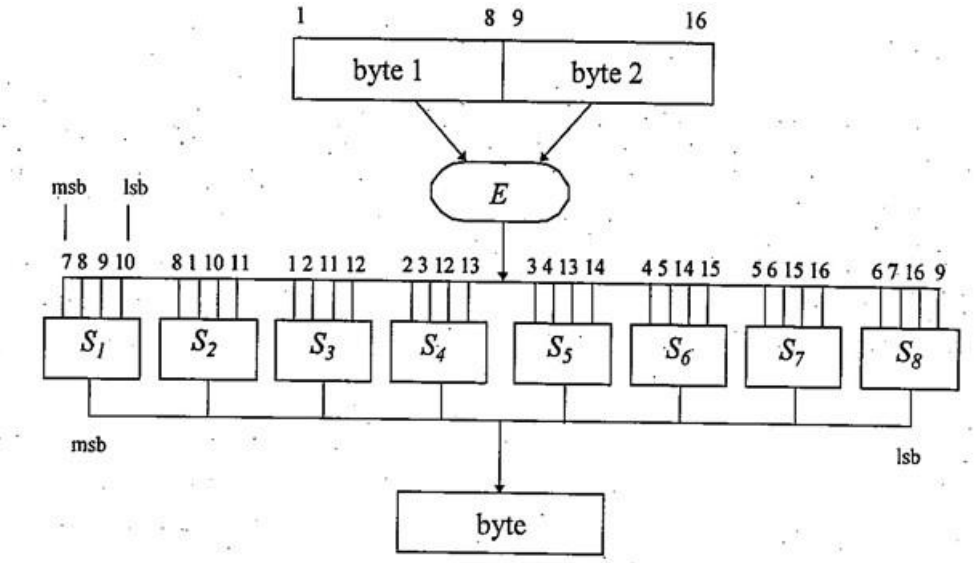


Figure 4: Structure of expander and functions f_1 and f_2

	Value of input nibble															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	0	1	0	0	0	1	1	1	1	1	0	0	1	0	0	1
S_2	1	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1
S_3	0	0	1	1	0	0	1	0	1	1	1	0	1	0	0	1
S_4	1	1	0	1	0	1	1	0	0	0	1	1	0	0	0	1
S_5	0	1	1	0	0	0	1	1	1	1	0	1	0	1	0	0
S_6	1	0	1	0	1	1	0	1	1	0	0	1	0	1	0	0
S_7	1	0	0	1	0	1	1	1	1	0	1	0	0	0	0	1
S_8	0	1	1	0	0	0	0	1	0	1	0	1	1	0	1	1

Figure 5: Truth table of f_1

	Value of input nibble															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	1	1	1	0	0	0	1	0	0	0	1	1	1	0	0	1
S_2	1	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1
S_3	0	1	0	0	1	0	0	1	0	0	1	1	0	1	1	1
S_4	0	0	1	1	1	0	0	1	1	1	0	1	0	1	0	0
S_5	1	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1
S_6	1	0	1	0	0	0	0	1	1	0	0	1	0	1	1	1
S_7	0	1	0	1	1	0	0	0	1	0	0	1	1	1	1	0
S_8	0	1	1	0	1	0	1	1	1	0	1	0	0	0	0	1

Figure 6: Truth table of f_2

6 TEA2 ALGORITHM DESCRIPTION

6.1 TEA2 Functional Components

6.1.1 Summary of Components

The cryptographic algorithm TEA2 consists of the following functional components as depicted in Figure 7:

- A set of eight 8-bit wide shift registers, called Output Register (clause 6.1.3).
- A set of ten 8-bit wide shift registers, called Cipher Key Register (clause 6.1.4).
- A byte permutation function *P* (byte substitution) (clause 6.1.5).
- Two functions *E* to expand the 16-bit output of two 8-bit registers to 32 bits word (clause 6.1.6).
- Two non-linear functions f_1 and f_2 to compute a byte from the expanded 32-bit word (clause 6.1.7).
- An 8-bit permutation function *BP* (wire crossing) (clause 6.1.8).
- Five 8-bit wide XOR functions to combine section outputs (bytes) and functions in the feedback of the Cipher key register and the output register.

6.1.2 Notation

The symbol $\oplus$ denotes the bytewise addition modulo two (exclusive or); i.e. msb of byte *x* is added to msb of byte *y*, ..., lsb of byte *x* to lsb of byte *y*.

x (i)1	*x(i)2*	*x(i)1* $\oplus$ *x(i)2*
0	0	0
0	1	1
1	0	1
1	1	0
x(i) is bit *i* of byte *x* *y(i)* is bit *i* of byte *y* *i = 0..., 7*		

6.1.3 Output Register

The output register is denoted by R_0, R_1..., R_6, and R_7, and functions as an 8-bit wide shift register. The output bytes of sections R_0 up to R_5 are also used as input for the other functional components of TEA2.

The output of R_7 is added to the outputs of the other functional components and returned to the first section R_0 as depicted in Figure 7.

Each 19 steps the byte output of R_7 is used as key byte for encryption or decryption. Before the process as described above can take place the output register is loaded with an Initialization Vector (see clause 6.2.3) and initialized (see clause 6.2.4).

6.1.4 Cipher Key Register

The Cipher key (CK) register, denoted by K_0, K_1, ..., K_8, and K_9, is a ten section 8-bit wide shift register. The bytes of a CK are loaded into the CK register as shown in Figure 8 and before the Initialization Vector is loaded into the Output Register. During initialization and the generation of Cipher bytes the Cipher key mixing is done, according to the irreducible polynomial $X^{10} + X^3 + 1$ and the permutation function P, as follows:

with K'_i denoting the next byte value (i.e. after one step) of the register section K_i:

$K'_0 = P\ (K_9 \oplus K_2)$

$K'_1 = K_0$

$K'_2 = K_1$

$K'_3 = K_2$

$K'_4 = K_3$

$K'_5 = K_4$

$K'_6 = K_5$

$K'_7 = K_6$

$K'_8 = K_7$

$K'_9 = K_8$

The series of the mixed and permutated CK bytes is also offered to the feedback circuit of the output register.

6.1.5 Byte Permutation Function

The CK byte permutation function P is an arbitrary permutation on 256 bytes. The look-up table for this permutation is given in Figure 9. The input to the function is two 4-bit nibbles, one higher nibble and one lower nibble. In the output, the left 4-bit nibble is the higher one, e.g. P (37) = A1.

6.1.6 Expander E

The expander function converts a two-byte input to a 32-bit word, i.e. eight 4-bit nibbles for the non-linear function f_1 and f_2. The structure of expander E and functions f_1 and f_2 is shown in Figure 10.

In this figure, bits 1-8 are the bits of R_1, respectively R_4. Bits 9-16 are the bits of R_0, respectively R_3. Bits 1 and 9 are the most significant bits. The 4-bit nibble inputs to the S_i boxes are the result of reading R_1 R_0 (for the input of f_1), respectively R_4 R_3 (for the input of f_2). The bit left is the msb of each 4-bit nibble.

6.1.7 Nonlinear Function f_1

The non-linear functions f_1 and f_2 have the structure shown in Figure 10.

Each of the boxes S_1 to S_8 receives the nibble input concerned from the expander E and computes a bit for the output byte. S_1 is the most significant bit in the output byte, and S_8 is the least significant one.

The functions f_1 and f_2 are given in the truth tables Figure 11 and Figure 12, respectively. The hexadecimal value of the input nibble for each Si is the one given in the top row of these figures. The computed output bit for the corresponding S_i can be read in the column below the input nibble row.

6.1.8 8-bit Permutation BP

The permutation BP is a so-called wire crossing, i.e. only one permutation with a fixed pattern. If the eight bits of R_5 are numbered 12345678, the order of the bits after BP becomes 48572136. The left bit in both bytes is the most significant and the right bit is the least significant.

6.1.9 Feedback of output register

The results of the functional components *BP*, f_1, f_2 and CK byte permutation *P* are added in the feedback path of the output register and affect the operation (i.e. operation after loading the CK and Initialization vector) as follows:

- with R'_i denoting the next byte value (i.e. after one step) of the output register section R_i:

  - $R'_0 = R_7 \oplus BP(R_5) \oplus R_2 \oplus f_1 [E (R_1, R_0)] \oplus P_{Out}$
    $R'_1 = R_0$
  - $R'_2 = R_1$
  - $R'_3 = R_2 \oplus f_2 [E (R_4, R_3)]$
  - $R'_4 = R_3$
  - $R'_5 = R_4$
  - $R'_6 = R_5$
  - $R'_7 = R_6$

6.2 Key Stream Generation

6.2.1 Summary

The algorithm consists of four main phases: the CK loading, the IV loading, the run-up and the key byte generation proper.

During the CK loading, the initial state of the Cipher key register is determined. Next, the initial state of the output register is determined by loading of the Initialization Vector as described in clause 6.2.3. Thereafter, a run-up is carried out during which the initial contents of the Output Register (converted and adapted IV) and the CK Register (CK) are changed by the effect of the *BP, E,* f_1, f_2 and *P* functions (see clause 6.2.4).

After the run-up is completed, each output cycle produces a key byte from the key byte stream as specified in clause 6.2.5. At any point during the run-up and the generation of the key byte stream, the state of the algorithm is determined by the states of the output and the CK registers.

6.2.2 CK loading

The CK loading depends solely on the 80-bit Cipher Key. The CK bytes are loaded as depicted in the Load Map shown in Figure 8. The CK bytes C_1, C_2, ..., C_{10} are shifted into the register from K_0 to K_9.

6.2.3 IV loading

The output register is first loaded with a 29-bit Initialization Vector. This IV is converted to a 32-bit word by taking the three most significant bits as zeroes, and the remaining 29 bits as the given IV. For instance, if the IV is the binary value:

11010 00011010 11100010 00000110

the 32-bit word becomes:

00011010 00011010 11100010 00000110.

The least significant byte of that 32-bit word is loaded into R_3, the next byte into R_4, the next one into R_5, and, finally, the most significant (the padded one) becomes the R_6 initial value. The cells R_0, R_1, R_2 and R_7 are loaded in the same order with the 32-bit word XORed with the constant mask **5A6E3278**. This gives the following initialization assignments, with the 32-bit word as a four-byte number $F_1F_2F_3F_4$:

$R_7 = F_1 \oplus 5A$

$R_6 = F_1$

$R_5 = F_2$

$R_4 = F_3$

$R_3 = F_4$

$R_2 = F_2 \oplus 6E$

$R_1 = F_3 \oplus 32$

$R_0 = F_4 \oplus 78$

Using the IV, mentioned in the example above, this produces:

1 8

01000000	00011010	00011010	11100010
00000110	01110100	11010000	01111110

for the:

$R_7, ..., R_4,$ and $R_3, ..., R_0$ bytes.

The eight bits of each R_i numbered from 1 to 8 are loaded into the register locations R_i (7), R_i (6), ..., R_i (0) respectively.

6.2.4 Run-up

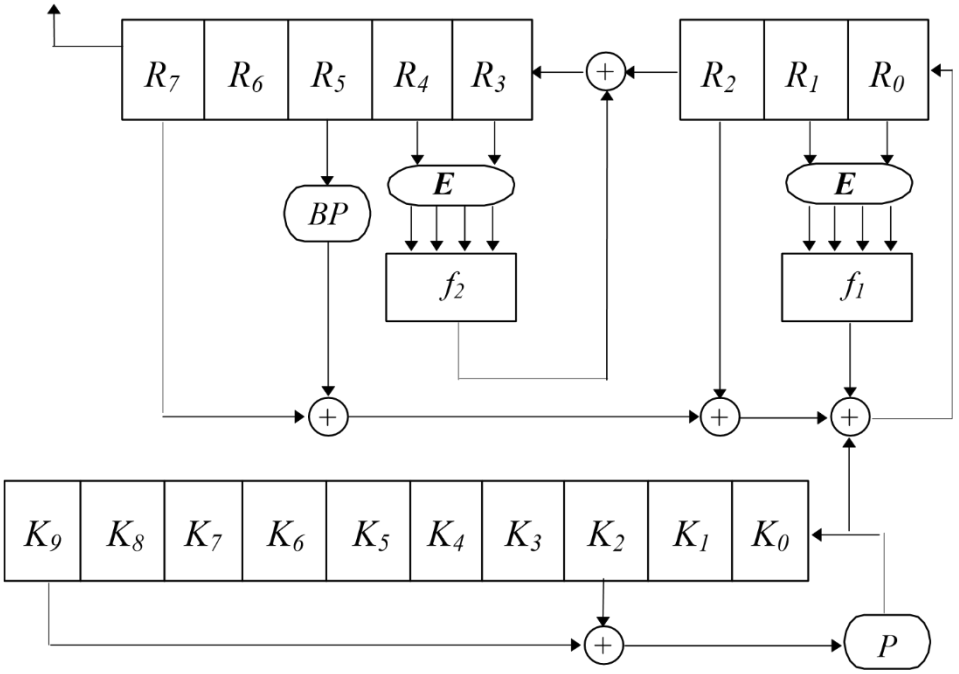
After the IV load into the output register all functional components of TEA2 become operational and 50 initializing steps are performed to bring the algorithm in the CK and IV dependent starting point from which the generation of key bytes can start. For run-up and key byte generation a step is defined as applying the formulae in clauses 6.1.4 and 6.1.9 once.

6.2.5 Key byte generation

After the run-up cycle one step is made to produce the first key byte. Successive key bytes are generated each 19 steps.

More precisely, the keystream generator, being byte orientated, generates 8 output bits at a time. Keystream bytes are generated by stepping the algorithm 19 times and then taking the value held in the Output Register stage R_7. The 8 bits in this register are then used, most-significant bit first, as the next 8 bits of the keystream.

6.3 Figures of TEA2 Algorithm



$R_0....R_7$	Output Register (eight 8-bit wide registers)
$K_0....K_9$	Cipher Key load register (ten 8-bit wide registers)
E	Expansion from 16 to 32 bits
f_1, f_2	Non-linear function: each eight functions of 4 bits
BP	Permutation of 8 bits (wire crossing)
P	Permutation on 2^8 elements (byte substitution)

Figure 7: Functional components of TEA2

CK = C_1 / C_2 / C_3 / C_4 / C_5 / C_6 / C_7 / C_8 / C_9 / C_{10}

with: C_i = CK byte i

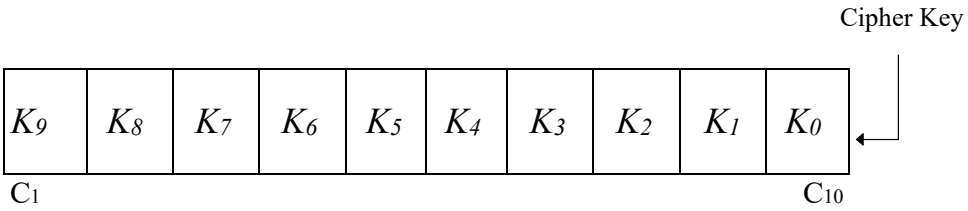


Figure 8: Cipher Key Load Map

Hg		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
h	0	62	DA	FD	B6	BB	9C	D8	2A	AB	28	6E	42	E7	1C	78	9E
e	1	FC	CA	81	8E	32	3B	B4	EF	9F	8B	DB	94	0F	9A	A2	96
r	2	1B	7A	FF	AA	C5	D6	BC	24	DF	44	03	09	0B	57	90	BA
	3	7F	1F	CF	71	98	07	F8	A1	60	F7	52	8D	E5	D7	69	87
n	4	14	ED	92	EB	B3	2F	E9	3D	C6	50	5A	A7	45	18	11	C4
i	5	CE	AC	F4	1D	82	54	3E	49	D5	EE	84	35	41	3A	EC	34
b	6	17	E0	C9	FE	E8	CB	E6	AE	68	E2	6B	46	C8	47	B2	E3
b	7	97	10	0E	B8	76	5B	BE	F5	A6	3C	8F	F6	D1	AF	C0	5E
l	8	7E	CD	7C	51	6D	74	2C	16	F2	A5	65	64	58	72	1E	F1
e	9	04	A8	13	53	31	B1	20	D3	75	5F	A4	56	06	8A	8C	D9
	A	70	12	29	61	4F	4C	15	05	D2	BD	7D	9B	99	83	2B	25
i	B	D0	23	48	3F	B0	2E	0D	0C	C7	CC	B7	5C	F0	BF	2D	4E
n	C	40	39	9D	21	37	77	73	4B	4D	5D	FA	DE	00	80	85	6F
p	D	22	91	DC	26	38	E4	4A	79	6A	67	93	F3	FB	19	A0	7B
u	E	F9	95	89	66	B9	D4	C1	DD	63	33	E1	C3	B5	A3	C2	27
t	F	0A	88	A9	1A	6C	43	EA	AD	30	86	36	59	08	55	01	02

Left = higher nibble

Figure 9: Byte permutation lookup table

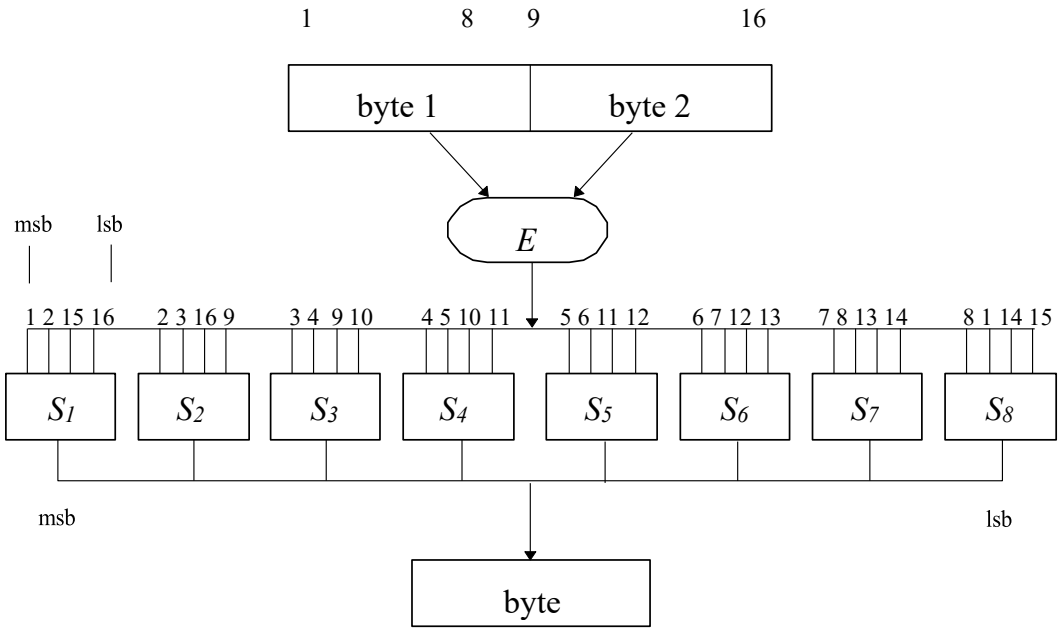


Figure 10: Structure of expander and functions f_1 and f_2

Value of Input Nibble

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	1	1	0	1	0	0	0	1	0	1	1	0	0	0	1	1
S_2	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0
S_3	1	0	1	1	0	0	1	0	1	1	0	0	1	0	0	1
S_4	0	0	1	0	1	0	0	1	1	1	0	0	1	1	1	0
S_5	0	1	1	0	1	0	1	1	1	0	0	0	1	1	0	0
S_6	0	0	0	1	0	0	1	1	0	1	1	0	1	1	0	1
S_7	1	0	1	0	0	1	1	1	0	1	1	0	0	0	0	1
S_8	1	0	0	1	1	1	1	0	1	0	1	0	0	1	0	0

Figure 11: Truth table of f_1

Value of input nibble

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	1	0	0	0	1	0	1	1	0	0	1	1	0	1	1	0
S_2	0	1	0	0	1	1	0	1	1	0	0	1	0	0	1	1
S_3	0	0	0	1	0	1	1	1	0	1	1	0	1	1	0	0
S_4	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1
S_5	0	1	1	1	1	0	0	1	1	1	0	0	0	1	0	0
S_6	1	0	0	1	0	0	1	1	0	1	0	0	1	1	0	1
S_7	1	0	0	0	0	1	0	1	1	1	1	0	1	0	0	1
S_8	0	1	0	1	0	0	0	1	0	1	1	0	1	0	1	1

Figure 12: Truth table of f_2

7 TEA3 ALGORITHM DESCRIPTION

7.1 TEA3 Functional Components

7.1.1 Summary of Components

The cryptographic algorithm TEA3 consists of the following functional components as depicted in Figure 13:

- An Output Register comprising a set of eight 8-bit wide shift register stages (clause 7.1.3).
- A Cipher Key Register comprising a set of ten 8-bit wide shift register stages (clause 7.1.4).
- A byte permutation function P (byte substitution) (clause 7.1.5).
- Two instances of a function E which expands the 16 bits formed from the concatenation of two 8-bit registers to 32 bits (clause 7.1.6).
- Two nonlinear functions f_1 and f_2 which each compute one byte from the 32-bit output of the expander function E (clause 7.1.7).
- An 8-bit permutation function BP (wire crossing) (clause 7.1.8).
- Five 8-bit wide XOR functions to combine register stage outputs and function outputs in the feedbacks of the Cipher Key Register and the Output Register.

7.1.2 Notation

The symbol ⊕ denotes the bytewise addition modulo two (exclusive or); i.e. msb of byte x is added to msb of byte y, ..., lsb of byte x to lsb of byte y.

x (i)1	*x(i)2*	*x(i)1 ⊕ x(i)2*
0	0	0
0	1	1
1	0	1
1	1	0
x(i) is bit *i* of byte *x* *y(i)* is bit *i* of byte *y* *i=0…..,7*		

7.1.3 Output Register

The Output Register comprises eight stages denoted by bytes R_0 .....R_7 and functions as an eight stage 8-bit wide shift register. Stages R_1 and R_2, and R_4, R_5 and R_6 are also used as input for the other functional components of TEA3. Stage R_7 is added to the outputs of the other functional components and returned to the first stage R_0 as defined below and as depicted in Figure 13.

The outputs of the functional components *BP*, *E*, f_1, f_2, and the Cipher Key Register (K_{out}) are added in the feedback paths of the Output Register and define its operation as follows:

$R'_0 = R_7 \oplus BP\,[R_4] \oplus f_2\,(E\,[R_2,, R1]) \oplus K_{out}$

$R'_1 = R_o$

$R'_2 = R_1$

$R'_3 = R_2$

$R'_4 = R_3$

$R'_5 = R_4 \oplus f_1\,(E\,[R_6, R_5])$

$R'_6 = R_5$

$R'_7 = R_6$

with R'_i denoting the next byte value (i.e. after one step) of the Output Register stage R_i, and K_{out} denoting the output from the Cipher Key Register block (i.e. $K_{out} = K_9 \oplus P\,[K_7 \oplus K_2]$).

Before the process as described above can take place the Output Register is loaded with an Initialization Vector (see clause 7.2.3). Following loading it is initialized (see clause 7.2.4).

Once initialized, the algorithm is repeatedly stepped 19 times and the resultant contents of byte R_7 following each 19/h step is used as a keystream byte for encryption or decryption.

7.1.4 Cipher Key Register

The Cipher Key Register comprises ten stages denoted by bytes K_0 .....K_9 and functions as a ten stage 8-bit wide shift register. Stages K_2 and K_7 are also used as input for the other functional components of TEA3. Stage K_9 is added to the outputs of the other functional components and returned to the first stage K_0 as defined below and as depicted in Figure 13.

During run-up and the generation of keystream bytes the Cipher Key Register is re-circulated as follows (*P* is the permutation function):

$K'_0 = K_{out} = K_9 \oplus P\,(K_7 \oplus K_2)$

$K'_1 = K_0$

$K'_2 = K_1$

$K'_3 = K_2$

$K'_4 = K_3$

$K'_5 = K_4$

$K'_6 = K_5$

$K'_7 = K_6$

$K'_8 = K_7$

$K'_9 = K_8$

with K'_i denoting the next byte value (i.e. after one step) of the register section K_i, K_{out} is an input to the feedback circuit of the Output Register.

7.1.5 Byte Permutation Function P

The byte permutation function *P* is a fixed random-like permutation on 256 bytes. The look-up table for this permutation is given in Figure 15. The input to the function is two 4-bit nibbles, one higher nibble and one lower nibble. In the output, the left 4-bit nibble is the higher one, e.g. P (27) = 5B. In this example '2' is the higher input nibble, '7' is the lower input nibble, '5' is the higher output nibble and 'B' is the lower output nibble. The example is indicated by the shaded cell in Figure 15.

7.1.6 Expander E

The expander function converts a two-byte input to a 32-bit word. The 32-bits are subdivided into eight 4-bit nibbles for the nonlinear functions f_1 and f_2. The structure of expander *E* and functions f_1 and f_2 is shown in Figure 16.

In this figure, bits 1-8 are the bits of R_6 (R_2). Bits 9-16 are the bits of R_5 (R_1). Bits 1 and 9 are the most significant bits. The 4-bit nibble inputs to the Si boxes are the result of reading R_6 R_5 (for the input of f_1), and R_2 R_1 (for the input of f_2). The left bit is the msb of each 4-bit nibble.

7.1.7 Nonlinear Function f_1

The nonlinear functions f_1 and f_2 have the structure shown in Figure 16.

Each of the boxes S_1 to S_8 receives the nibble input concerned from the expander *E* and computes a bit for the output byte. S_1 is the most significant bit in the output byte, and S_8 is the least significant one.

The functions f_1 and f_2, are given in Figure 17 and Figure 18, respectively. The hexadecimal value of the input nibble for each S_i is the one given in the top row of these figures. The computed output bit for the corresponding S_i can be read in the column below.

7.1.8 8-bit Permutation BP

The permutation *BP* is a simple wire-crossing, i.e. a reordering of the bits within the byte. If the eight bits of R_4 are numbered 12345678, the order of the bits after *BP* becomes 38467215. The left bit (bit 1) in both bytes is the most significant and the right bit (bit 8) is the least significant.

7.2 Keystream Generation

7.2.1 Summary

The algorithm consists of three main phases: the CK and the IV loading, the run-up and the keystream generation proper.

During the loading phase the Cipher Key is used to set the initial state of the Cipher Key Register (clause 7.2.2), and the Initialization Vector is used to set the initial state of the Output Register (clause 7.2.3).

Following the initializations of the two registers a run-up phase is carried out to complete the initializations of the algorithm (see clause 7.2.4).

After the run-up is completed, each output cycle produces 8 bits of keystream as specified in clause 7.2.5. At any point during the run-up and the generation of the keystream, the state of the algorithm is determined by the states of the Output Register and the Cipher Key Register.

7.2.2 CK loading

The 80-bit Cipher Key is used to initialize the ten stage Cipher Key Register as depicted in the Load Map shown in Figure 14. The CK bytes C1, C2…, 10, starting with the most significant byte (C1), are shifted into the Cipher Key Register Ki, entering through stage K0. Note that the feedback function is not enabled at this time.

I.e. the bytes of CK are initially shifted into the register such that:

$$K_{10-i} = C_i, \qquad i = 1 \ldots 10$$

7.2.3 IV loading

The 29-bit IV is converted to a 32-bit word (four bytes) by adding three '0' bits to the most significant end. For instance, if the IV is the binary value:

11010 00011010 11100010 00000110

the 32-bit word becomes:

00011010 00011010 11100010 00000110.

The resultant 32-bit IV is used to initialize the Output Register *R*. The least significant byte of that 32-bit word is loaded into R_3, the next byte into R_4, the next one into R_5, and, finally, the most significant into R_6. The stages R_0, R_1, R_2 and R_7 are loaded in the same order with the 32-bit word XORed with the hexadecimal constant C43A7D51. So, taking the (adjusted) IV as IV1IV2IV3IV4, we have the following initializations of *R*:

$R_7 = IV_1 \oplus$ C4

$R_6 = IV_1$

$R_5 = IV_2$

$R_4 = IV_3$

$R_3 = IV_4$

$R_2 = \oplus$ 3A

$R_1 = IV_1 \oplus$ 7D

$R_0 = IV_4 \oplus$ 51

Using the IV, mentioned in the example above, we get:

1 8

11011110 00011010 00011010 11100010 00000110 00100000 10011111 01010111

for the R_7. ... .. R_0 bytes.

The eight bits of each R_i numbered from 1 to 8 are loaded into the register locations R_i (7), R_i (6), ..., R_i (0) respectively.

7.2.4 Run-up

After the CV and IV have been loaded into their respective registers all functional components of the TEA3 become operational and 32 steps are performed to bring the algorithm to a state where the generation of keystream can start.

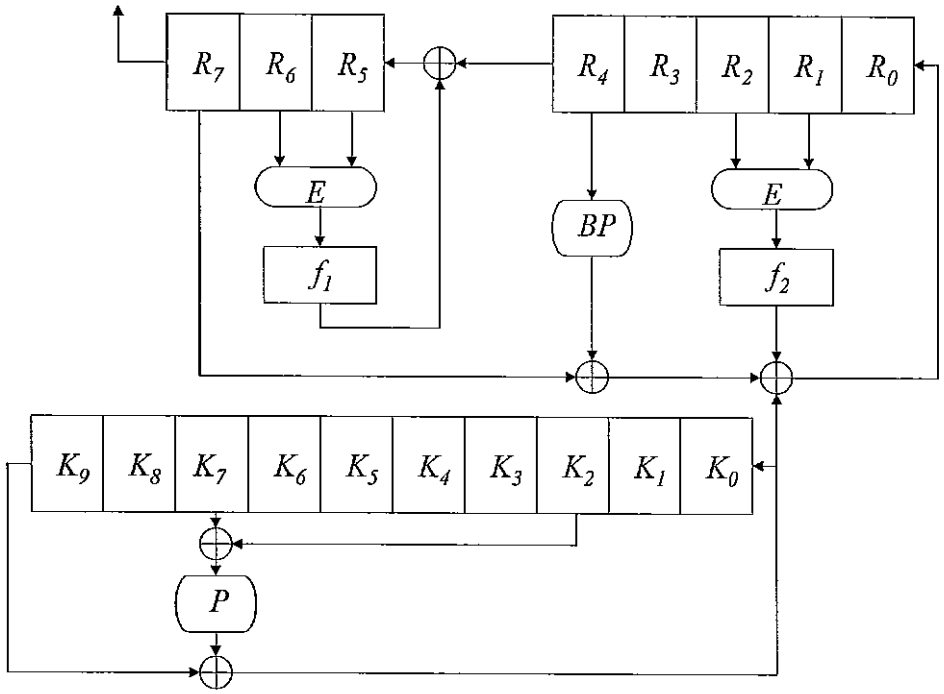
For run-up and keystream generation a step is defined as applying the formulae in clauses 7.1.4 and 7.1.9 once.

7.2.5 Key byte generation

The keystream generator, being byte orientated, generates 8 output bits at a time.

Keystream bytes are generated by stepping the algorithm 19 times and then taking the value held in the Output Register stage R_7. The 8 bits in this register are then used, most-significant bit first, as the next 8 bits of the keystream.

7.3 Figures of TEA3 Algorithm



$R_0....R_7$	Output Register (eight 8-bit wide register stages)
$K_0....K_9$	Crypto Key Register (ten 8-bit wide register stages)
E	Expansion from 16 to 32 bits
f_1, f_2	Nonlinear function: each eight functions of 4 bits
BP	Permutation of 8 bits (bit re-ordering)
P	Permutation on 2^8 elements (byte substitution)

Figure 13: Functional components of TEA3

$CK = C_1 / C_2 / C_3 / C_4 / C_5 / C_6 / C_7 / C_8 / C_9 / C_{10}$

with: C_i = CK byte i

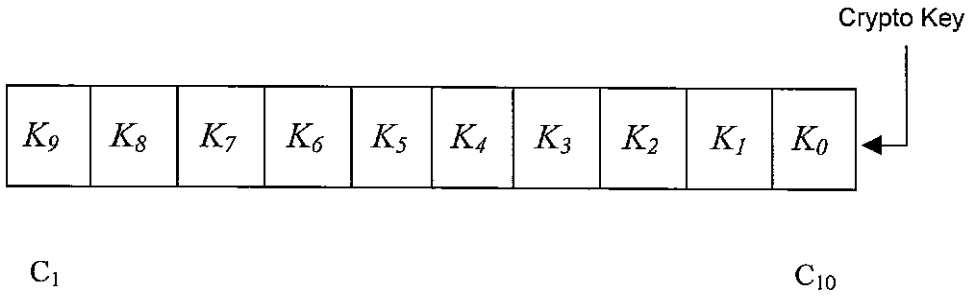


Figure 14: Cipher Key Load Map

Lower Nibble Input

Higher Nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7D	BF	7B	92	AE	7C	F2	10	5A	0F	61	7A	98	76	07	64
1	EE	89	F7	BA	C2	02	0D	E8	56	2E	CA	58	C0	FA	2A	01
2	57	6E	3F	4B	9C	DA	A6	5B	41	26	50	24	3E	F8	0A	86
3	B6	5C	34	E9	06	88	1F	39	33	DF	D9	78	D8	A8	51	B2
4	09	CD	A1	DD	8E	62	69	4D	23	2B	A9	E1	53	94	90	1E
5	B4	3B	F9	4E	36	FE	B5	D1	A2	8D	66	CE	B7	C4	60	ED
6	96	4F	31	79	35	EB	8F	BB	54	14	CB	DE	6B	2D	19	82
7	80	AC	17	05	FF	A4	CF	C6	6F	65	E6	74	C8	93	F4	7E
8	F3	43	9F	71	AB	9A	0B	87	55	70	0C	AD	CC	A5	44	E7
9	46	45	03	30	1A	EA	67	99	DB	4A	42	D7	AA	E4	C2	D5
A	F0	77	20	C3	3C	16	B9	E2	EF	6C	3D	1B	22	84	2F	81
B	1D	B1	3A	E5	73	40	D0	18	C7	6A	9E	91	48	27	95	72
C	68	0E	00	FC	C5	5F	F1	F5	38	11	7F	E3	5E	13	AF	37
D	E0	8A	49	1C	21	47	D4	DC	B0	EC	83	28	B8	F6	A7	C9
E	63	59	BD	32	85	08	BE	D3	FD	4C	2C	FB	A0	C1	9D	B3
F	52	8C	5D	29	6D	04	BC	25	15	8B	12	9B	D6	75	A3	97

Higher Nibble Input

Left = higher nibble

Figure 15: Byte permutation lookup table

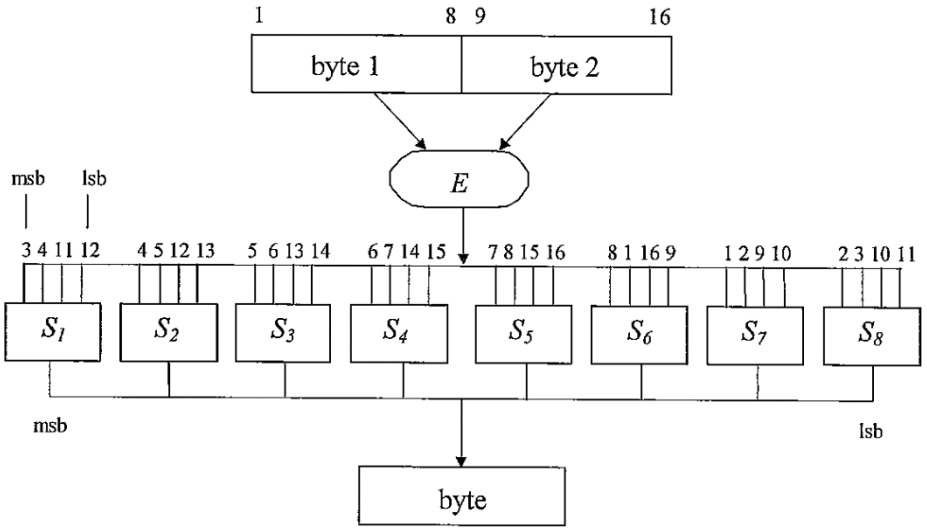


Figure 16: Structure of expander and functions f_1 and f_2

Value of input nibble

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	1	1	0	0	1	0	0	1	0	1	1	1	0	1	0	0
S_2	1	1	0	0	1	0	0	1	1	0	1	1	0	0	1	0
S_3	1	0	0	1	0	0	1	1	0	1	0	0	1	1	0	1
S_4	1	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1
S_5	0	0	1	0	0	0	1	1	1	0	0	1	1	1	1	0
S_6	0	0	1	1	0	1	1	0	1	1	1	0	1	0	0	0
S_7	1	0	1	1	0	1	1	0	0	0	1	0	0	1	0	1
S_8	0	0	0	1	1	0	1	0	1	0	1	1	1	0	0	1

Figure 17: Truth table of f_1

Value of input nibble

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	1	1	0	0	0	1	1	0	0	0	1	0	1	1	1	0
S_2	0	0	1	0	1	0	1	1	1	0	0	1	1	1	0	0
S_3	0	0	1	1	0	1	1	0	1	1	1	0	1	0	0	0
S_4	0	1	1	1	0	0	1	1	1	0	0	1	0	1	0	0
S_5	0	0	1	1	0	0	0	1	1	1	0	1	0	1	1	0
S_6	0	0	1	1	0	0	1	0	1	1	1	0	1	0	0	1
S_7	1	0	0	0	0	1	1	0	1	1	1	0	0	1	0	1
S_8	1	1	1	0	0	1	0	1	0	1	0	0	1	0	0	1

Figure 18: Truth table of f_2

8 TEA4 ALGORITHM DESCRIPTION

8.1 TEA4 Functional Components

8.1.1 Summary of Components

The cryptographic algorithm TEA4 consists of the following functional components as depicted in Figure 19:

- An Output Register comprising a set of eight 8-bit wide shift register stages (clause 8.1.3).
- A Cipher Key Register comprising a set of seven 8-bit wide shift register stages (clause 8.1.4).
- A byte permutation function *P* (byte substitution) (clause 8.1.5).
- Two instances of a function *E* which expands the 16 bits formed from the concatenation of two 8-bit registers to 32-bits (clause 8.1.6).
- Two nonlinear functions f_1 and f_2 which each compute one byte from the 32-bit output of the expander function E (clause 8.1.7).
- An 8-bit permutation function *BP (wire* crossing) (clause 8.1.8).
- Six 8-bit wide XOR functions to combine register stage outputs and function outputs in the feedbacks of the Cipher Key Register and the Output Register.

8.1.2 Notation

The symbol ⊕ denotes the bytewise addition modulo two (exclusive or); i.e. msb of byte *x* is added to msb of byte *y*, ..., lsb of byte *x* to lsb of byte *y*.

x (i)1	*x(i)2*	*x(i)1 ⊕ x(i)2*
0	0	0
0	1	1
1	0	1
1	1	0
x(i) is bit *i* of byte *x* *y(i)* is bit *i* of byte *y* *i=0…..,7*		

8.1.3 Output Register

The Output Register comprises eight stages denoted by bytes R_0 .....R_7 and functions as an eight stage 8-bit wide shift register. Stages R_1 and R_2, and R_4, R_5 and R_6 are also used as input for the other functional components of TEA4. Stage R_7 is added to the outputs of the other functional components and returned to the first stage R_0 as defined below and as depicted in Figure 19.

The outputs of the functional components *BP*, *E*, f_1, f_2, and the Cipher Key Register (K_{out}) are added in the feedback paths of the Output Register and define its operation as follows:

$R'_0 = R_7 \oplus BP[R_6] \oplus f_1 (E [R_5, R_4]) \oplus K_{out}$

$R'_1 = R_0$

$R'_2 = R_1$

$R'_3 = R_2$

$R'_4 = R_3 \oplus f_2 (E [R_1, R_0])$

$R'_5 = R_4$

$R'_6 = R_5$

$R'_7 = R_6$

with R'_i denoting the next byte value (i.e. after one step) of the Output Register stage R_i, and K_{out} denoting the output from the Cipher Key Register block (i.e. $K_{out} = K_6 \oplus P\ (K_5 \oplus K_1)$).

Before the process as described above can take place the Output Register is loaded with an Initialization Vector (see clause 3.3). Following loading it is initialized (see clause 8.2.4).

Once initialized, the algorithm is repeatedly stepped 19 times and the resultant contents of byte R_7 following each 19th step is used as a keystream byte for encryption or decryption.

8.1.4 Cipher Key Register

The Cipher Key Register comprises seven stages denoted by bytes $K_o....K_6$ and function as a seven stage 8-bit wide shift register. Stages K_1 and K_5 are also used as input for the other functional components of TEA4. Stage K_6 is added to the outputs of the other functional components and returned to the first stage K_o as defined below and as depicted in Figure 19.

During run-up and the generation of keystream bytes the Cipher Key Register is re-circulated as follows (P is the permutation function):

$K'_0 = K_{out} = K_6 \oplus P\ (K_5 \oplus K_1)$

$K'_1 = K_0$

$K'_2 = K_1$

$K'_3 = K_2$

$K'4 = K3$

$K'_5 = K_4$

$K'_6 = K_5$

with K'_i denoting the next byte value (i.e. after one step) of the register section K_i, K_{out} is an input to the feedback circuit of the Output Register.

8.1.5 Byte Permutation Function P

The byte permutation function P is a fixed random-like permutation on 256 bytes. The look up table for this permutation is given in Figure 21. The inputs to the function are two 4-bit nibbles, one higher nibble and one lower nibble. In the output, the left 4-bit nibble is the higher one, e.g. P (28) = D4. In this example '2' is the higher input nibble, '8' is the lower input nibble, 'D' is the higher output nibble and '4' is the lower output nibble. The example is indicated by the shaded cell in Figure 21.

8.1.6 Expander E

The expander function conve1ts a two-byte input to a 32-bit word. The 32-bits are subdivided into eight 4-bit nibbles for the nonlinear function f_1 and f_2. The structure of expander E and functions f_1 and f_2 is shown in Figure 22.

In Figure 22, bits 1-8 are the bits of $R_5\ (R_1)$. Bits 9-16 are the bits of $R_4\ (R_0)$. Bits 1 and 9 are the most significant bits. The 4-bit nibble inputs to the Si boxes are the result of reading $R_5\ R_4$ (for the input of f_1), and $R_1\ R_0$ (for the input of f_2). The left bit is the msb of each 4-bit nibble.

8.1.7 Nonlinear Function f_1

The nonlinear functions f_1 and f_2 have the structure shown in Figure 22.

Each of the boxes S_1 to S_8 receives the nibble input concerned from the expander E and computes a bit for the output byte. S_1 is the most significant bit in the output byte, and S_8 is the least significant one.

The functions f_1 and f_2, are given in truth tables, Figure 23 and Figure 24, respectively. The hexadecimal value of the input nibble for each S_i is the one given in the top row of these figures. The computed output bit for the corresponding S_i can be read in the column below the input nibble row.

8.1.8 8-bit Permutation BP

The permutation *BP* is a simple wire-crossing, i.e. a reordering of the bits within the byte. If the eight bits of R_4 are numbered 12345678, the order of the bits after *BP* becomes 74638152. The left bit (bit 1) in both bytes is the most significant and the right bit (bit 8) is the least significant.

8.2 KEYSTREAM GENERATION

8.2.1 Summary

The algorithm consists of three main phases: the CK and the IV loading, the run-up and the keystream generation proper.

During the loading phase the Cipher Key is used to set the initial state of the Cipher Key Register (clause 8.2.2), and the Initialization Vector is used to set the initial state of the Output Register (clause 8.2.3).

Following the initializations of the two registers a run-up phase is carried out to complete the initializations of the algorithm (see clause 8.2.4).

After the run-up is completed, each output cycle produces 8 bits of keystream as specified in clause 8.2.5. At any point during the run-up and the generation of the keystream, the state of the algorithm is determined by the states of the Output Register and the Cipher Key Register.

8.2.2 CK loading

The 80-bit Cipher Key is used to initialize the seven stage Cipher Key Register as defined below and as depicted in Figure 20. All stages of the Cipher Key Register K are initially set to 0, and then the Cipher Key bytes are mixed into the register starting with the most significant byte of the key (C_1). The feedback defined in clause 8.1.4 is inoperative at this time.

The bytes of the Cipher Key (CK) are loaded into the key register in the following way:

Step 1: $K_0 = K_1 = K_2 = K_3 = K_4 = K_5 = K_6 = 0$ {Set all stages to 0}

Step 2: {Load the Cipher Key bytes}

For i = 1 To 10 Do

$K'_0 = CK_i \oplus K_6 \oplus P\,(K_5 \oplus K_1)$

$K'_1 = K_0$

$K'_2 = K_1$

$K'_3 = K_2$

$K'_4 = K_3$

$K'_5 = K_4$

$K'_6 = K_5$

where K'_i is denoting the next byte value (i.e. after one step) of the register section K_i.

I.e. During loading the CK_i bytes are added modulo two to K_6 and to the result of substitution by the permutation function P of the modulo two addition of the output of Sections K_1 and K_5. This result is then passed to section K_0.

8.2.3 IV loading

The 29-bit IV is converted to a 32-bit word (four bytes) by adding three '0' bits to the most significant end. For instance, if the IV is the binary value:

11010 00011010 11100010 00000110

the 32-bit word becomes:

00011010 00011010 11100010 00000110.

The resultant 32-bit IV is used to initialize the Output Register *R*. The least significant byte of that 32-bit word is loaded into R_3, the next byte into R_4, the next one into R_5, and the most significant into R_6. The stages R_0, R_1, R_2 and R_7 are loaded in the same order with the 32-bit word XORed with the hexadecimal constant 56DC28E3. So, taking the (adjusted) IV as $IV_1IV_2IV_3IV_4$ gives the following initializations of *R*:

$R_7 = IV_1 \oplus 56$
$R_6 = IV_1$
$R_5 = IV_2$
$R_4 = IV_3$
$R_3 = IV_4$
$R_2 = IV_2 \oplus DC$
$R_1 = IV_3 \oplus 28$
$R_0 = IV_4 \oplus E3$

Using the IV, mentioned in the example above, gives:

1 8

01001100 0001 1010 00011010 11100010 00000110 11000110 11001010 11100101

for the R_7 .... R_0 bytes.

The eight bits of each R_i numbered from 1 to 8 are loaded into the register locations R_i (7), R_i (6), ..., R_i (0) respectively.

8.2.4 Run-up

After the CV and IV have been loaded into their respective registers all functional components of TEA4 become operational and 35 steps are performed to bring the algorithm to a state where the generation of keystream can start.

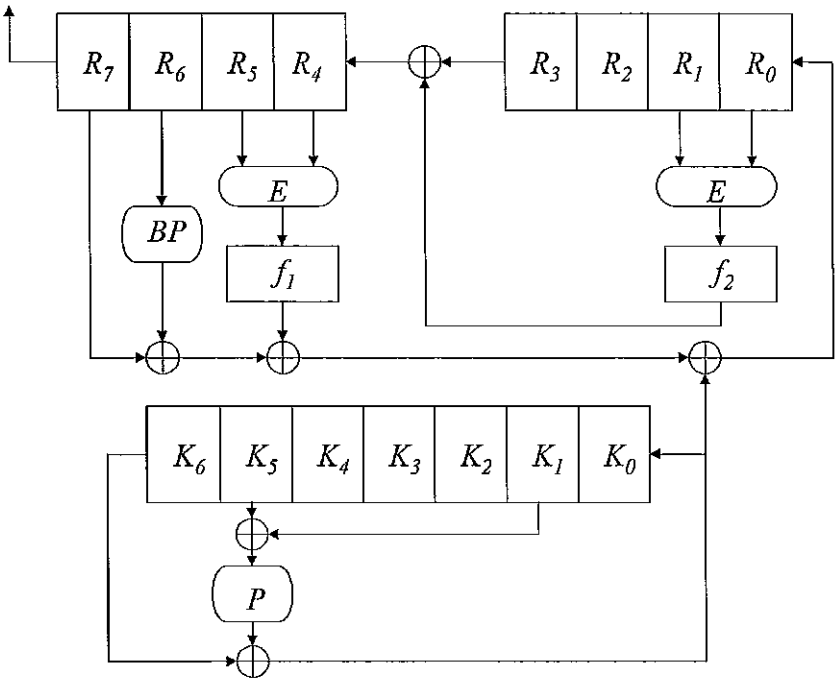
For run-up and keystream generation a step is defined as applying the formulae in clauses 8.1.4 and 8.1.9 once.

8.2.5 Keystream generation

The keystream generator, being byte orientated, generates 8 output bits at a time.

Keystream bytes are generated by stepping the algorithm 19 times and then taking the value held in the Output Register stage R_7. The 8 bits in this register are then used, most-significant bit first, as the next 8 bits of the keystream.

8.3 Figures of TEA4 Algorithm



$R_0....R_7$	Output Register (eight 8-bit wide register stages)
$K_0....K_6$	Crypto Key Register (seven 8-bit wide register stages)
E	Expansion from 16 to 32 bits
f_1, f_2	Nonlinear function: each eight functions of 4 bits
BP	Permutation of 8 bits (bit re-ordering)
P	Permutation on 2^8 elements (byte substitution)

Figure 19: Functional components of TEA4

Crypto Key = $CK_1 ..... CK_{10}$

with CK_i = *CK byte I*

* *State of register at start of keyloading process = 0000000*

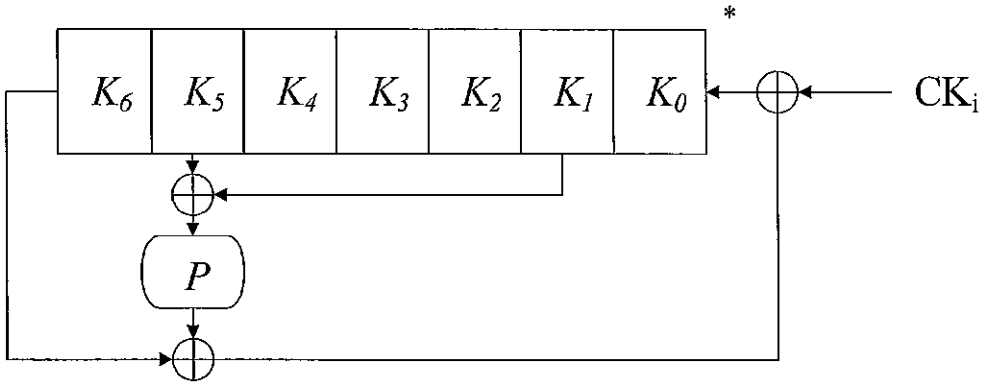


Figure 20: Cipher Key Load Map

		Lower Nibble Input														
H		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
i	0	D2	22	9B	76	EF	8A	EE	C6	5B	04	6B	F1	8B	43	14	13
g	1	00	80	F7	FC	03	B7	A0	AE	C8	A7	AC	82	EC	77	42	E2
h	2	18	51	47	02	52	0A	17	05	D4	7A	78	06	09	0C	62	5E
	3	6E	5F	3F	F4	B4	A1	5C	CC	56	2C	D0	CD	BE	07	3A	D7
n	4	CE	B1	20	CB	2B	33	A8	A6	45	2A	97	DA	7F	4F	55	9D
i	5	4D	44	91	58	92	10	C1	F6	DC	E5	B0	40	B2	0E	2D	9A
b	6	37	67	DF	4E	7B	FF	AF	AB	E7	53	83	8F	0D	48	D1	4B
b	7	49	DE	EA	E9	E1	1B	68	6F	81	32	12	2F	ED	94	8E	B5
l	8	35	7E	[illegible]	A3	E3	BA	95	57	6D	8D	2E	1A	98	EB	65	64
e	9	0B	[illegible]	BD	E0	73	5D	27	39	A4	1D	3E	C4	F9	87	28	E4
	A	9E	1E	99	BC	F5	FE	34	01	41	C9	70	BF	69	AD	DD	46
i	B	E6	15	[illegible]	[illegible]	[illegible]	7D	59	30	6C	84	4C	B9	9C	3B	86	6A
n	C	E8	F8	1C	D6	4A	9F	19	F2	8C	F3	A5	C3	3D	96	C5	D8
p	D	75	C7	36	AA	FB	29	CF	72	25	0F	A9	90	B6	FD	63	BB
u	E	50	88	7C	16	23	D3	FA	93	5A	71	21	B3	31	74	38	66
t	F	C2	CA	61	24	60	89	26	54	F0	B8	85	08	1F	A2	DB	D9

Left = higher nibble

Figure 21: Byte permutation lookup table

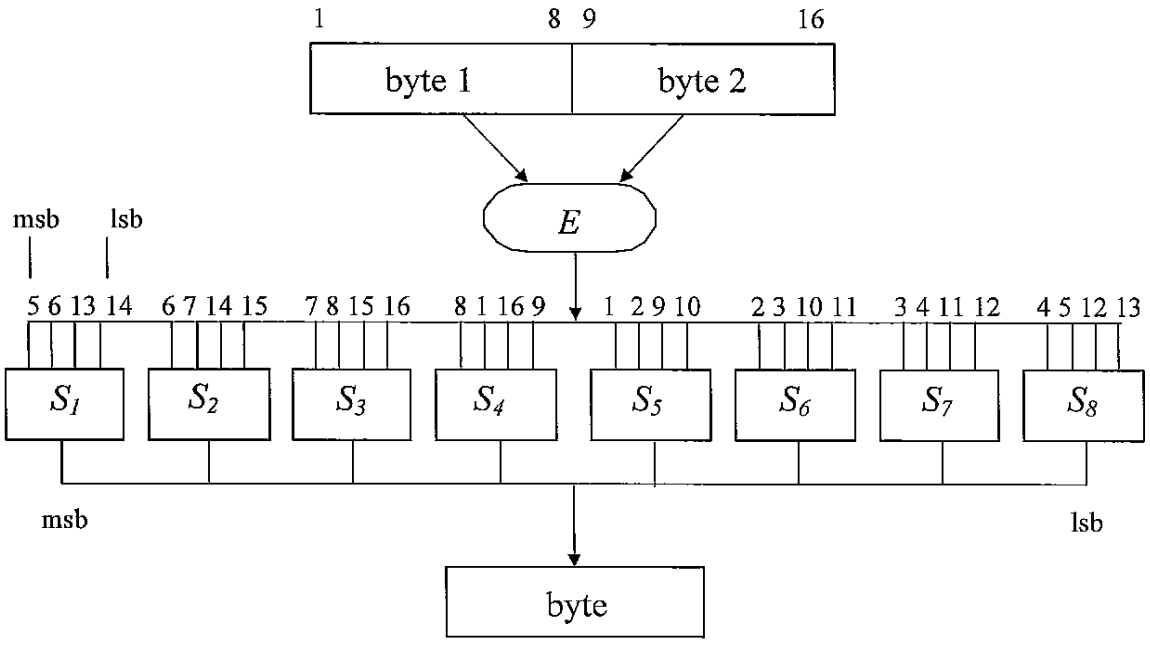


Figure 22: Structure of expander and functions f_1 and f_2

	Value of input nibble															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0
S_2	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	1
S_3	1	1	0	0	1	1	0	1	0	0	0	1	0	1	1	0
S_4	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0
S_5	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1
S_6	0	0	1	0	0	1	0	1	1	0	1	1	0	1	1	0
S_7	0	1	1	1	1	0	0	1	0	0	0	1	1	0	1	0
S_8	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	1

Figure 23: Truth table of f_1

	Value of input nibble															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1	0	0	1	1	0	1	1	0	0	1	0	0	0	1	1	1
S_2	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1
S_3	1	0	1	1	0	1	1	0	1	1	0	0	1	0	0	0
S_4	1	0	0	0	1	1	0	0	0	1	1	0	1	0	1	1
S_5	0	0	1	0	1	0	1	1	1	1	0	0	0	1	1	0
S_6	1	1	1	0	0	1	0	1	1	0	0	0	0	1	1	0
S_7	1	0	1	0	0	1	0	0	0	1	1	0	1	1	0	1
S_8	0	0	0	1	1	0	1	0	1	0	1	1	0	1	1	0

Figure 24: Truth table of f_2

Annex A (informative): Bibliography

- ETSI TS 101 053-2: "Rules for the management of the TETRA standard encryption algorithms; Part 2: TEA2".
- ETSI TS 101 053-3: "Rules for the management of the TETRA standard encryption algorithm TEA3".
- ETSI TS 101 053-4: "Rules for the management of the TETRA standard encryption algorithms; Part 4: TEA4".

History

Document history		
V1.1.1	July 2024	Publication