暗号技術仕様書

CIPHERUNICORN-E

日本電気株式会社

1 概要

1.1 目的

本仕様書は、64 ビットブロック暗号 CIPHERUNICORN-E の設計方針、設計基準、暗号アルゴリズムの仕様を報告することを目的とする。

1.2 記号の定義

本仕様書では、以下の表記を用いる。

P	：平文1ブロック
C	：暗号文1ブロック
$F^{\{i\}}$	：第i段F関数(i=0,1,…,15)
$L^{\{i\}}$	：第i段L関数(i=0,1,…,8)
$FK^{\{i\}}[j]$	：$F^{\{i\}}$の本流で使用する第j番目32ビット拡大鍵(j=0,1)
$SK^{\{i\}}[j]$	：$F^{\{i\}}$の一時鍵生成部で使用する第j番目32ビット拡大鍵(j=0,1)
$LK^{\{i\}}[j]$	：$L^{\{i\}}$で使用する第j番目32ビット拡大鍵(j=0,1)
$FK^{\{i\}}$	：$F^{\{i\}}$の本流で使用する2つの拡大鍵のまとまり(関数鍵)
$SK^{\{i\}}$	：$F^{\{i\}}$の一時鍵生成部で使用する2つの拡大鍵のまとまり(シード鍵)
wk0	：4ビット一時鍵
wk1,wk2	：8ビット一時鍵
sh[i][j]	：T関数入力番号テーブルのi行j列の要素
‖	：データの連結
∧	：論理積
⊕	：排他的論理和
⊞	：加算(mod 2^{32})
x<<n	：xをnビット左論理シフト
x>>n	：xをnビット右論理シフト

1.3 ビット／バイト／ワードの順序

本仕様書では、big endian 表記を用いる。

Q を 128 ビットデータ(quad word)、
D を 64 ビットデータ(double word)、
W を 32 ビットデータ(word)、
B を 8 ビットデータ(byte)、
E を 1 ビットデータ(bit)

とすると、

$$
\begin{aligned}
Q &= D_0 \parallel D_1 \\
&= W_0 \parallel W_1 \parallel W_2 \parallel W_3 \\
&= B_0 \parallel B_1 \parallel B_2 \parallel \cdots\cdots \parallel B_{15} \\
&= E_0 \parallel E_1 \parallel E_2 \parallel \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \parallel E_{127}
\end{aligned}
$$

2 設計方針、基準

どのような構造を持つブロック暗号に対しても有効である代表的な解読法として、線形解読法と差分解読法がある。これらの解読法は、データ撹拌関数の撹拌の偏りを利用して鍵の情報を推定する。撹拌の偏りは、撹拌処理の最も基本となる処理での撹拌の偏りから生じることが多い。従って、基本となる処理において撹拌の偏りが検出できない構造にすることが望ましい。

我々は、基本となる処理であるラウンド関数において、撹拌の偏りが現れないように設計することとした。撹拌の偏りは、入力と出力の関係を統計的に調べることで調査することにした。

また、設計過程のアルゴリズムを統一的に評価できるように、暗号アルゴリズムをブラックボックスとみなして入力と出力の関係を調べられる共通な評価尺度を設定した。偏りがなく十分撹拌できた状態を以下のように定め、我々が採用した統計的手法を用いて確認することとした。

- 高い確率で成立する入力ビットと出力ビットの関係が存在しない
- 高い確率で成立する出力ビット間の関係が存在しない
- 高い確率で成立する入力ビットの変化と出力ビットの変化の関係が存在しない
- 高い確率で成立する拡大鍵ビットの変化と出力ビットの変化の関係が存在しない
- 高い確率で 0 あるいは 1 となる出力ビットが存在しない

ブロックサイズは DES(Data Encryption Standard)と同じ 64 ビットとし、秘密鍵長は DES よりも長い 128 ビットとする。本暗号は、32 ビットプロセッサ上でより高速に実装できるようにする。

2.1 データ撹拌部

2.1.1 Feistel構造

暗号の基本構造は以下の利点から Feistel 構造を採用する。

- 暗号化と復号がほぼ同じ速度
- ラウンド関数の構造に制約がない
- 解析実績が多い

2.1.2 初期/終期および中間処理

1 段目ラウンド関数への入力、最終段ラウンド関数への入力が既知となり攻撃しやすくなることを防ぐため、また、未知の攻撃に対する防御として、初期/終期およびラウンド関数 2 段おきに 64 ビット幅の関数を付加する。

2.2 ラウンド関数

2.2.1 二重構造

ラウンド関数は二重構造を採用し、一方が解読されてももう一方で安全性を保証できる構造にする。

ラウンド関数は本流と一時鍵生成部より構成する。各々に拡大鍵を入力する(関数鍵とシード鍵)。一時鍵生成部により一時鍵を作成し、本流に合流する。

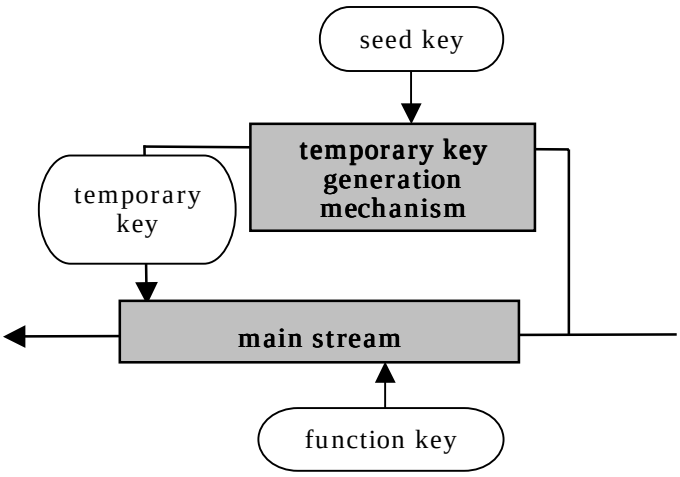


図 2.1 二重構造のラウンド関数

2.2.2 本流

本流は以下の構造とする。

- 一時鍵が固定ならば全単射
- 本流のみでも十分な撹拌を行う

2.2.3 一時鍵生成部

一時鍵生成部は以下の構造とする。

- 一時鍵がとりうる全値域において均等に出力する
- 本流より軽い構造(並列処理が可能な場合を考慮)
- 本流とは異なる構造(構造の違いによる安全性の保証)
- 一時鍵のサイズはシード鍵サイズよりも小さくする
- 一時鍵生成部のみでも十分な撹拌を行う

攻撃者は一時鍵生成部が本流より軽い構造のため一時鍵生成部を最初に攻撃すると考え、仮に一時鍵が既知となってもシード鍵の候補が複数存在し、シード鍵から秘密鍵、シード鍵から関数鍵を推定することが困難になることを期待する。

2.2.4 演算子

基本的に 32 ビットプロセッサでの実装を考え、32 ビットプロセッサ上で高速に処理できる演算子を採用する。また、代数的構造が異なる演算を組み合わせることにより暗号の強度向上が見込めるため、代数的構造の異なる演算を組み合わせて使用する。

2.2.5 演算単位

Truncated differential attack への対策として、演算単位は 8 ビット、32 ビットの 2 種類を使用する。

2.3 換字テーブル

換字テーブルは、8 ビット入出力のテーブル 4 種を組み合わせ使用する。8 ビット入出力テーブルの条件は、以下の通りとする。

- 全単射
- 最大差分確率が 2^{-6}
- 最大線形確率が 2^{-6}
- 代数次数が 7 次
- 入出力多項式の次数大、かつ項数が多い
- 平均拡散ビット数(入力 1 ビットの変化による出力変化ビット数)が 4.0
- 不動点がない

上記条件を満たす換字テーブルの生成法として、GF(2^8)上の逆数関数とアフィン変換の組み合わせを採用する。

GF(2^8)上の逆数関数は、最大差分・線形確率が 2^{-6}(最良)であることが知られており、代数次数は 7 次の全単射関数である。また、入出力多項式の次数は 254 次と大きい。アフィン変換を取り入れることにより、入出力多項式の項数の増加が期待できる。

また、8 ビット入出力テーブル 4 種を組み合わせ使用するため、既約多項式はそれぞれ異なるものを採用することにする。

換字テーブルの生成式を以下に示す。

$$S(x) = \mathrm{matrixA}\{(x + c)^{-1} \bmod g\} + d$$

ここで、

matrixA：GF(2)の 8×8 の全単射行列
c,d　　：8 ビットの定数(0 以外)
g　　：8 次の既約多項式

matrixA, c, d, g を乱数で選択し、上記条件を満たす換字テーブルを検索することにする。

2.4 鍵スケジューラ

鍵スケジューラは以下の構造とする。

- 秘密鍵から拡大鍵への写像が単射である
- どの拡大鍵にも秘密鍵の全情報が影響している
- 秘密鍵と拡大鍵、拡大鍵同士に高い確率で成立する関係がない(鍵関連攻撃への安全性)
- ラウンド関数の構成要素を利用する

3 暗号アルゴリズム

3.1 全体

本暗号 CIPHERUNICORN-E は、データブロック長 64 ビット、秘密鍵長 128 ビットの Feistel 構造の暗号である。

ラウンド数は16段で、ラウンド関数(F 関数)2段おきに64ビット幅の処理(L 関数)を行う。

鍵スケジューラは、秘密鍵を入力とする Feistel 構造である。ダミーループによる撹拌後、撹拌しながら拡大鍵の取り出しを繰り返す。

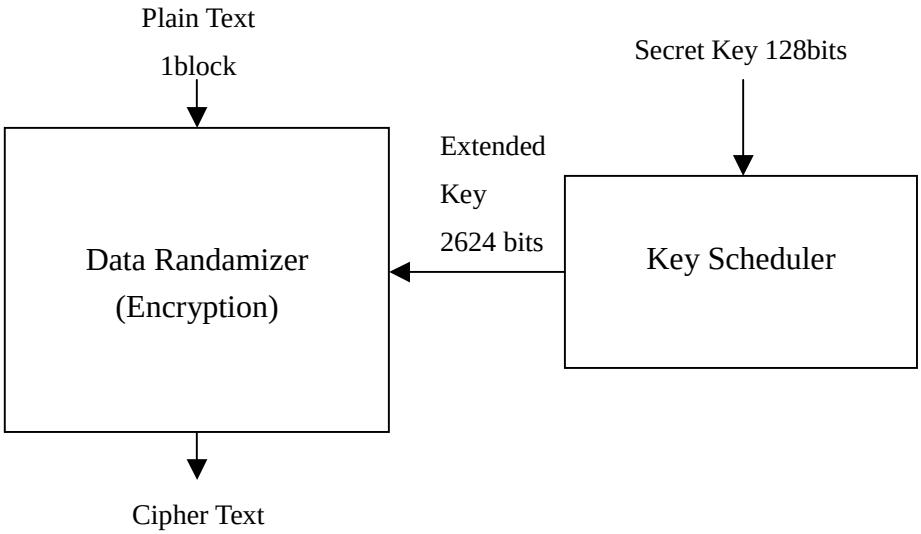


図 3.1 暗号化

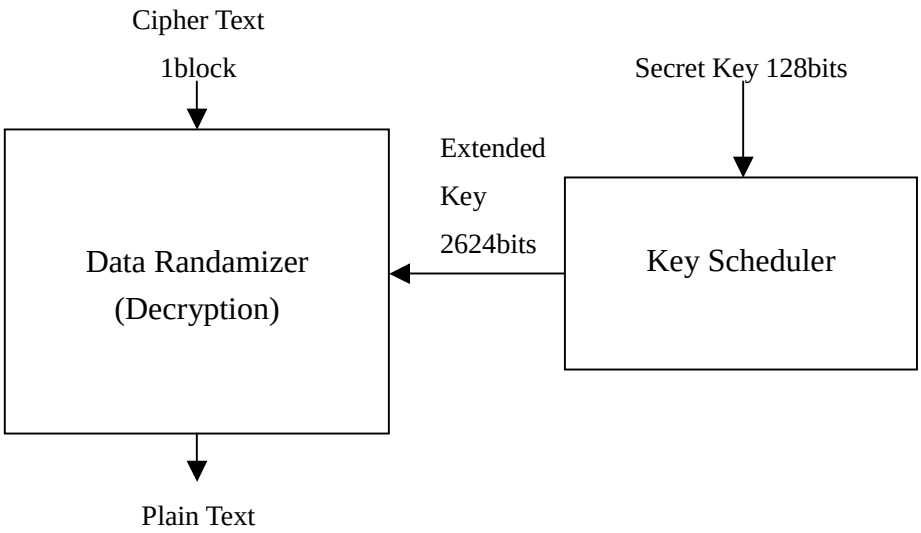


図 3.2 復号

3.2 データ撹拌部

3.2.1 暗号化

[入力] 平文1ブロック $P=P_0 \| P_1$ (64ビット)

F関数本流用拡大鍵 $FK^{\{i\}}=FK^{\{i\}}[0] \| FK^{\{i\}}[1]$ (64ビット: i=0,1,…,15)

F関数一時鍵生成部用拡大鍵 $SK^{\{i\}}=SK^{\{i\}}[0] \| SK^{\{i\}}[1]$ (64ビット: i=0,1,…,15)

L関数用拡大鍵 $LK^{\{j\}}=LK^{\{j\}}[0] \| LK^{\{j\}}[1]$ (64ビット: j=0,1,…,8)

[出力] 暗号文1ブロック $C=C_0 \| C_1$ (64ビット)

[処理] 撹拌は 16段Feistel構造による撹拌と、2段おきの64ビット撹拌のL関数により行う。

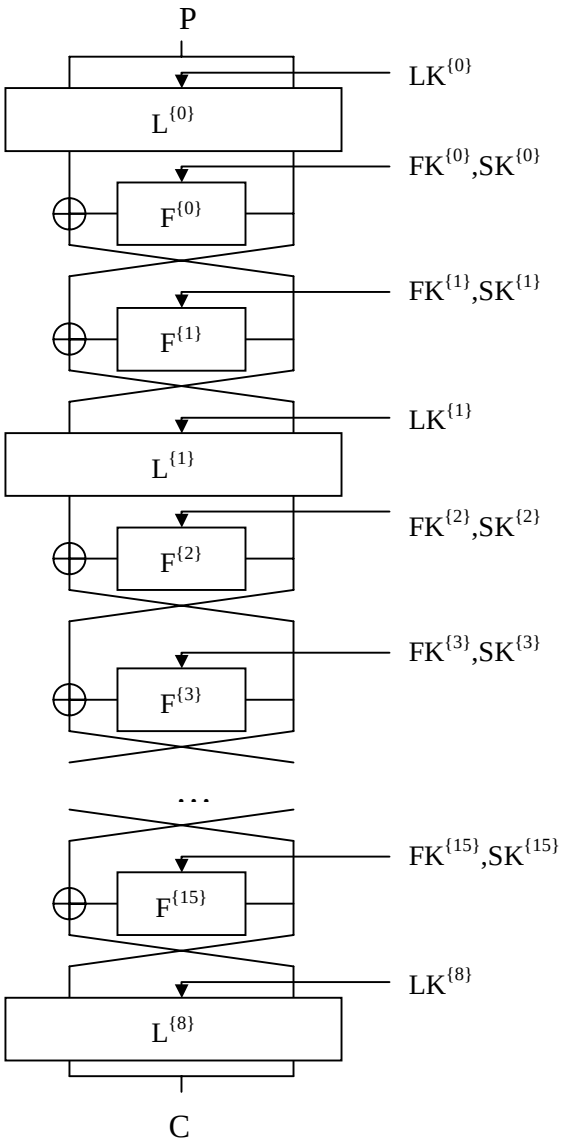


図 3.3 データ撹拌部(暗号化)

3.2.2 復号

[入力] 暗号文1ブロック $C=C_0 \| C_1$ (64ビット)

F関数本流用拡大鍵 $FK^{\{i\}}=FK^{\{i\}}[0] \| FK^{\{i\}}[1]$ (64ビット: i=0,1,…,15)

F関数一時鍵生成部用拡大鍵 $SK^{\{i\}}=SK^{\{i\}}[0] \| SK^{\{i\}}[1]$ (64ビット: i=0,1,…,15)

L関数用拡大鍵 $LK^{\{j\}}=LK^{\{j\}}[0] \| LK^{\{j\}}[1]$ (64ビット: j=0,1,…,8)

[出力] 平文1ブロック $P=P_0 \| P_1$ (64ビット)

[処理] 撹拌は 16段Feistel構造による撹拌と、2段おきの64ビット撹拌のL関数により行う。

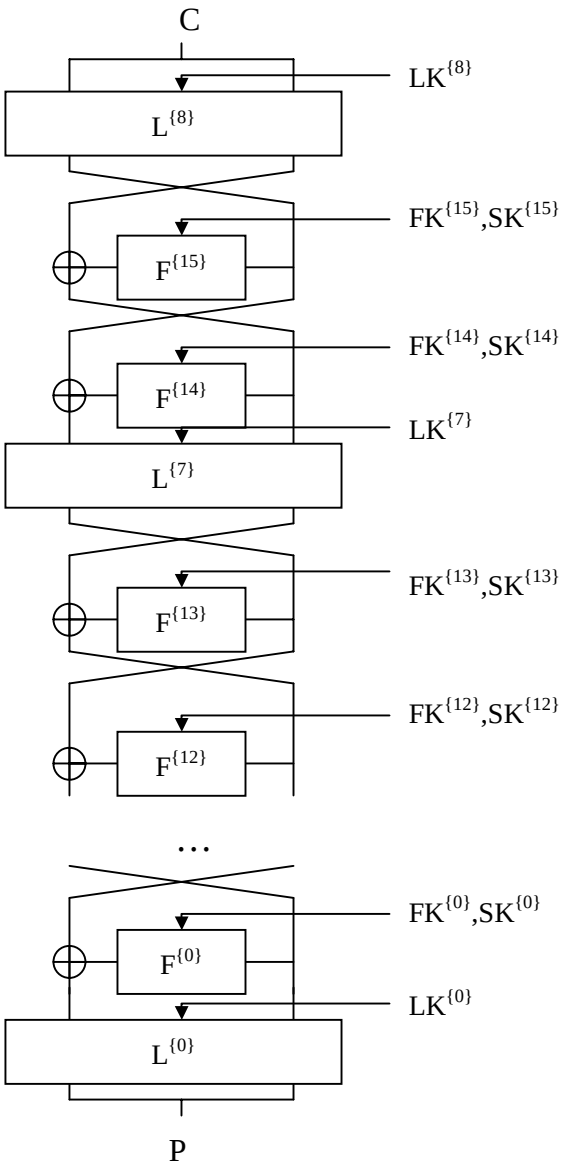


図 3.4 データ撹拌部(復号)

3.3 L関数

[入力] 入力データ $X=X_L \| X_R$ (64ビット)

$L^{\{i\}}$関数用拡大鍵$LK^{\{i\}}=LK^{\{i\}}[0] \| LK^{\{i\}}[1]$ (64ビット)

[出力] 出力データ $Z=Z_L \| Z_R$ (64ビット)

[処理] L関数は、入力データと鍵を以下の式で演算する。

暗号化と復号では、鍵の順序が逆である。

$$Z_L=X_L \oplus (X_R \wedge LK^{\{i\}}[1]) \oplus (X_L \wedge LK^{\{i\}}[0] \wedge LK^{\{i\}}[1])$$

$$Z_R=X_R \oplus (X_L \wedge LK^{\{i\}}[0]) \oplus (X_R \wedge LK^{\{i\}}[0] \wedge LK^{\{i\}}[1])$$

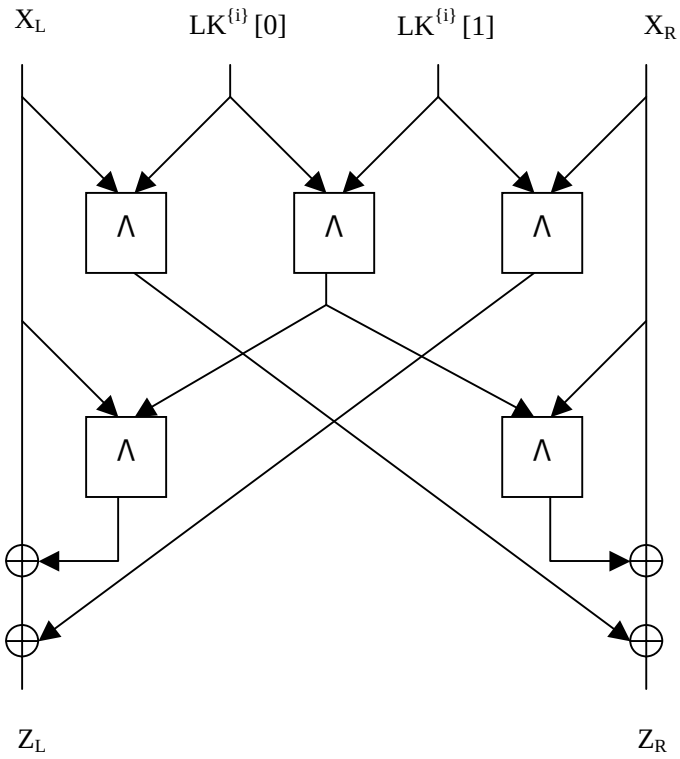


図 3.5 L 関数

3.4 F関数

[入力] 入力データ X (32ビット)

$F^{\{i\}}$関数本流用拡大鍵 $FK^{\{i\}}=FK^{\{i\}}[0] \| FK^{\{i\}}[1]$ (64ビット)

$F^{\{i\}}$関数一時鍵生成部用拡大鍵 $SK^{\{i\}}=SK^{\{i\}}[0] \| SK^{\{i\}}[1]$ (64ビット)

[出力] 出力データ Z (32ビット)

[処理] F関数は、本流と一時鍵生成部から構成される。

入力データ32ビットに$FK^{\{i\}}[0]$を加算した後、本流と一時鍵生成部に分岐する。本流では、入力番号0,1,2,3の順でT関数を実行し、$FK^{\{i\}}[1]$を加算する。以降のT関数は、wk0に基づいてShテーブルを参照し入力番号を決定する。最後の2つの各T関数の手前では、K関数を実行する。本流の32ビット出力が、F関数の出力となる。一時鍵生成部では、$SK^{\{i\}}[0]$を加算した後、Y関数(シフト数は(3,8,16))を実行する。次に入力番号0のT関数を実行する。$SK^{\{i\}}[1]$を加算後、再びY関数(7,9,13)を実行し、入力番号0,1のT関数を実行する。結果の最上位4ビットをwk0、最下位バイトをwk1、下位2バイト目をwk2とする。

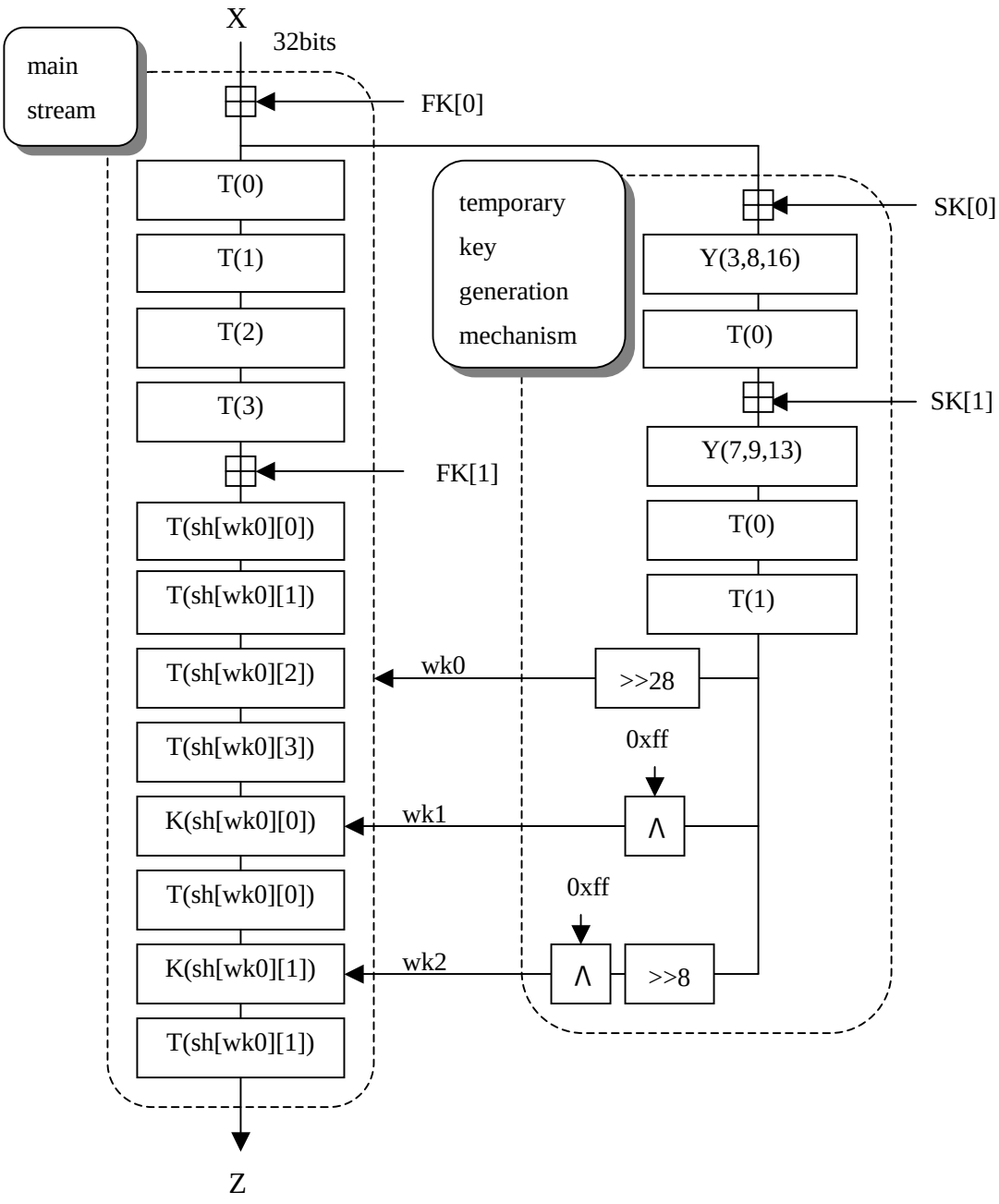

X
32bits
main
stream
FK[0]
T(0)
T(1)
T(2)
T(3)
FK[1]
T(sh[wk0][0])
T(sh[wk0][1])
T(sh[wk0][2])
T(sh[wk0][3])
K(sh[wk0][0])
T(sh[wk0][0])
K(sh[wk0][1])
T(sh[wk0][1])
Z
temporary
key
generation
mechanism
SK[0]
Y(3,8,16)
T(0)
SK[1]
Y(7,9,13)
T(0)
T(1)
wk0
>>28
0xff
wk1
∧
0xff
wk2
∧
>>8

図 3.6 F 関数

3.5 T関数

[入力] 入力データ X＝$X_0 \| X_1 \| X_2 \| X_3$ (32ビット)

入力番号 n (n=0,1,2,3)

[出力] 出力データ Z (32ビット)

[処理] 入力データを1バイトごとに区切り、入力番号に対応する1バイトを換字テーブルの入力値とする。換字テーブルは8ビット入力8ビット出力の4種類S_0,S_1,S_2, S_3とする。入力番号に対応する1バイトは換字テーブルの出力を使用し、それ以外のバイトは換字テーブルの出力を入力データと排他的論理和する。

Z=T(n)

ここで、

$T(0) = S_3(X_0) \| (S_0(X_0) \oplus X_1) \| (S_1(X_0) \oplus X_2) \| (S_2(X_0) \oplus X_3)$

$T(1) = (S_2(X_1) \oplus X_0) \| S_3(X_1) \| (S_0(X_1) \oplus X_2) \| (S_1(X_1) \oplus X_3)$

$T(2) = (S_1(X_2) \oplus X_0) \| (S_2(X_2) \oplus X_1) \| S_3(X_2) \| (S_0(X_2) \oplus X_3)$

$T(3) = (S_0(X_3) \oplus X_0) \| (S_1(X_3) \oplus X_1) \| (S_2(X_3) \oplus X_2) \| S_3(X_3)$

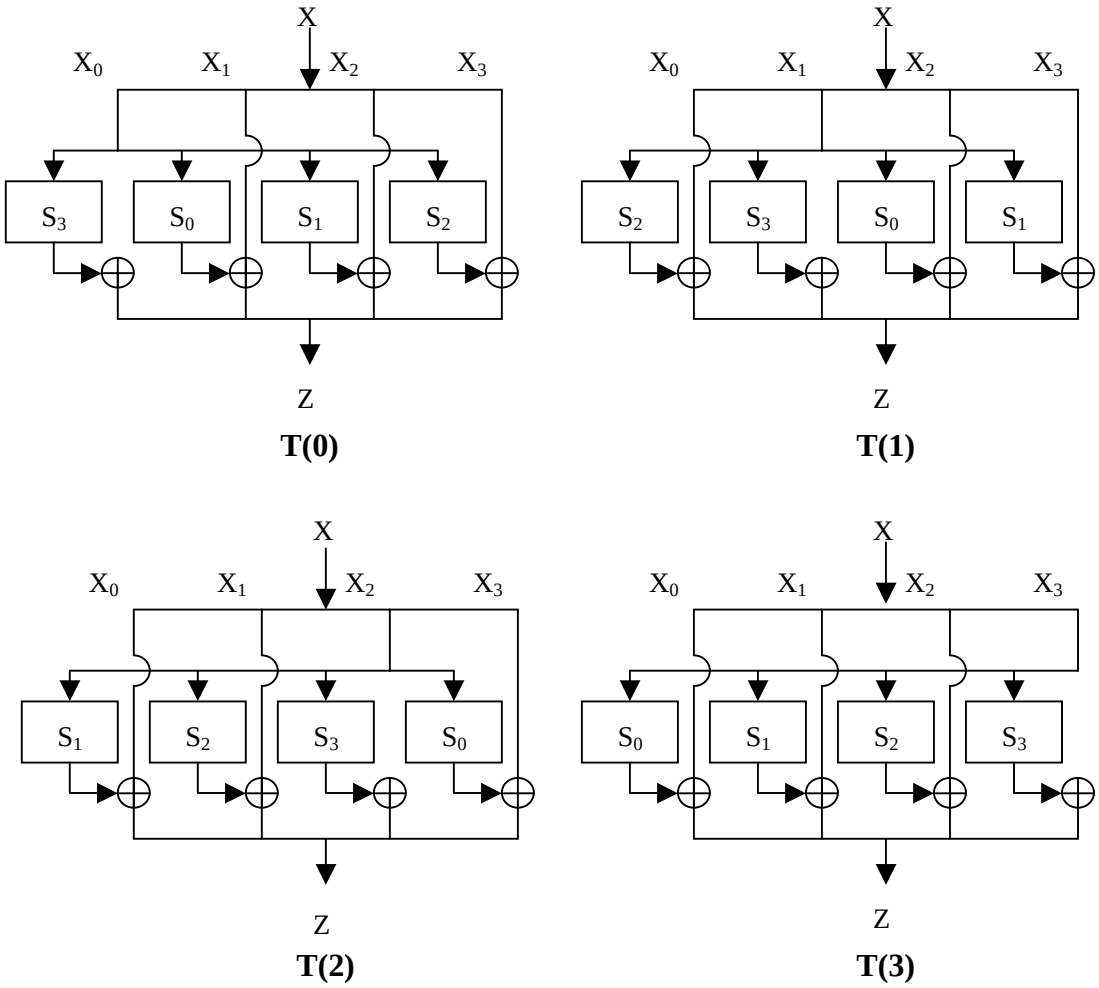


図 3.7 T 関数

3.6 K関数

[入力] 入力データ　X＝X_0 ‖ X_1 ‖ X_2 ‖ X_3 (32ビット)

入力番号　n (n=0,1,2,3)

一時鍵　wk=wk1またはwk2 (8ビット)

[出力] 出力データ　Z (32ビット)

[処理] 入力データを1バイトごとに区切り、入力番号に対応する1バイトに一時鍵を排他的論理和する。

Z=K(n,wk)

ここで、

K(0,wk) ＝ (X_0⊕wk) ‖ X_1 ‖ X_2 ‖ X_3
K(1,wk) ＝ X_0 ‖ (X_1⊕wk) ‖ X_2 ‖ X_3
K(2,wk) ＝ X_0 ‖ X_1 ‖ (X_2⊕wk) ‖ X_3
K(3,wk) ＝ X_0 ‖ X_1 ‖ X_2 ‖ (X_3⊕wk)

3.7 換字テーブル

[入力] 入力データ X (8ビット)

[出力] 出力データ Z (8ビット)

[処理] 換字テーブルS_nの入力データ位置のデータを出力する。

$$Z = S_n(X) \qquad n=0,1,2,3$$

4種の換字テーブルの生成式は以下の通りである。

$$S_n(x) = \mathrm{matrixA}\{(x + c)^{-1} \bmod g\} + d$$

表 3.1 換字テーブルのパラメータ

S_n	matrixA	c	g	d
S_0	{0x23, 0x4e, 0x9c, 0xb1, 0x49, 0xd8, 0xc6, 0xe4}	233	0x11d	28
S_1	{0x7e, 0x2a, 0xef, 0x52, 0x34, 0xa2, 0x70, 0xd7}	26	0x165	171
S_2	{0x32, 0x04, 0x8f, 0x83, 0x89, 0x67, 0xcf, 0x3b}	43	0x14d	155
S_3	{0x34, 0x20, 0xba, 0xd0, 0x66, 0xd7, 0xb2, 0xa8}	200	0x171	47

ここで、S_0 の matrixA＝{0x23, 0x4e, 0x9c, 0xb1, 0x49, 0xd8, 0xc6, 0xe4}は以下の GF(2) の 8×8 行列を示す。

$$\mathbf{matrixA} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

また、S_0 の既約多項式 g＝0x11d＝$100011101_{(2)}$は以下の多項式であることを示す。

$$g = x^8 + x^4 + x^3 + x^2 + 1$$

表 3.2 換字テーブル S_0

$S_0(0)=149,\ S_0(1)=111,\cdots\cdots\cdots\cdots,\ S_0(255)=92$

149	111	237	155	21	85	108	76	236	75	193	84	22	138	89	55
51	145	13	153	148	163	86	59	204	175	91	117	126	70	144	10
248	146	201	0	97	208	23	214	147	234	66	65	226	57	210	224
172	40	154	87	178	235	135	220	110	121	96	8	9	53	241	105
143	169	182	139	112	16	183	67	233	39	197	74	166	218	231	242
161	159	192	37	177	228	47	119	14	18	244	56	3	195	239	219
33	167	26	180	54	61	58	222	4	30	191	34	107	249	142	150
95	42	124	25	232	181	120	93	5	68	6	48	129	41	104	73
188	165	212	160	250	141	123	216	94	238	81	202	7	122	196	17
207	102	184	189	243	72	206	12	200	225	164	176	247	1	2	254
71	185	229	187	251	137	69	168	50	24	171	173	158	221	127	27
252	114	152	82	209	38	203	128	215	213	36	174	134	179	90	118
80	246	253	125	29	44	15	227	98	205	255	77	198	194	133	130
79	103	78	49	19	140	109	211	223	63	64	151	62	217	170	83
136	45	115	199	20	46	190	240	132	28	162	230	131	106	32	88
157	31	43	156	113	186	35	101	52	60	11	100	116	245	99	92

表 3.3 換字テーブル S_1

$S_1(0)=174,\ S_1(1)=255,\cdots\cdots\cdots\cdots,\ S_1(255)=53$

174	255	161	109	254	40	95	67	33	124	133	58	224	238	129	56
137	57	169	87	221	220	163	84	14	239	171	138	74	192	66	104
8	250	43	115	126	88	212	103	62	82	143	4	117	226	28	155
65	156	139	183	235	125	217	116	111	237	157	68	160	184	213	172
170	132	73	2	1	232	92	249	136	106	175	5	9	140	38	191
50	251	85	12	27	48	46	52	145	78	168	159	100	188	16	227
26	198	244	205	178	72	142	162	51	246	241	128	194	177	122	20
144	49	83	166	247	225	11	7	102	242	185	18	150	165	121	98
93	197	70	151	75	118	202	216	108	207	15	112	99	35	101	69
86	61	79	110	13	218	149	6	134	29	36	131	181	154	180	230
77	193	164	17	211	3	209	105	94	206	44	19	60	123	10	31
130	195	76	208	54	252	219	203	199	39	189	80	167	90	32	30
233	64	245	182	120	231	127	47	22	135	55	114	234	41	21	81
173	223	23	253	153	25	45	248	97	179	186	119	200	146	187	210
0	228	24	190	141	236	63	201	96	113	240	147	229	91	107	214
89	59	152	215	176	204	243	148	42	158	71	34	222	37	196	53

表 3.4 換字テーブル S_2

$S_2(0)=37$, $S_2(1)=34$,……, $S_2(255)=124$

37	34	162	132	134	220	91	143	41	45	229	247	98	178	68	56
212	97	70	15	58	72	216	208	14	96	214	217	133	179	28	154
120	123	83	100	235	3	230	160	193	245	164	155	255	175	79	148
227	219	23	95	111	11	87	104	163	203	189	29	156	173	211	64
157	53	196	89	81	4	84	16	192	74	13	181	20	184	57	183
90	119	93	207	38	131	94	60	116	1	213	122	5	101	144	117
75	46	8	172	170	152	231	210	66	54	10	187	128	204	12	102
243	115	137	147	159	233	59	221	253	112	165	198	105	222	234	153
43	201	121	180	86	205	225	242	182	55	63	232	254	44	9	21
136	65	114	31	40	49	0	36	169	22	249	35	62	17	174	248
158	151	24	50	176	108	67	127	150	18	2	168	194	171	195	145
99	25	80	224	33	200	197	118	161	61	142	77	190	209	48	139
238	206	42	125	239	237	52	223	88	167	26	130	76	191	7	71
215	27	126	6	251	51	241	129	135	246	244	146	32	177	73	82
226	110	78	186	240	141	166	69	107	85	103	149	250	109	202	19
113	140	138	39	185	228	106	47	252	199	188	92	218	30	236	124

表 3.5 換字テーブル S_3

$S_3(0)=24$, $S_3(1)=252$,……, $S_3(255)=34$

24	252	144	121	17	42	77	127	2	35	173	21	129	58	105	113
112	229	185	189	76	204	209	87	5	96	82	99	133	140	66	64
192	107	194	220	16	68	183	171	219	51	92	13	152	86	135	123
98	174	103	156	157	59	145	155	158	8	231	132	83	49	23	32
85	69	251	36	233	238	222	149	37	248	26	18	125	11	137	253
79	52	56	95	241	187	44	167	124	102	227	115	212	142	154	93
247	211	33	28	67	10	147	225	215	210	246	160	131	73	65	57
1	182	180	199	207	126	216	224	61	81	202	196	146	188	119	128
50	30	91	161	89	12	195	74	235	223	226	172	245	7	218	159
242	217	208	38	163	45	39	4	62	136	104	179	88	197	6	0
141	190	243	214	109	162	60	165	198	228	221	164	106	101	203	236
143	48	110	80	176	78	234	181	97	84	20	70	29	168	27	72
71	90	255	19	254	114	25	230	47	43	100	178	40	41	249	186
150	205	184	201	139	75	54	22	63	244	108	175	46	169	240	153
151	116	122	232	166	117	14	94	111	206	237	177	200	31	170	120
213	53	148	15	55	239	3	191	134	250	193	9	130	118	138	34

3.8 Shテーブル

[入力] 入力データ wk0 (4ビット)

列番号 i (i=0,1,2,3)

[出力] 出力データ n (n=0,1,2,3)

[処理] T関数入力番号テーブルShの第wk0行、第i列のデータを出力する。

n ＝ Sh[wk0][i]　　wk0=0,1,…,15、i=0,1,2,3

Shテーブルは以下の条件にあてはまるものにした。

(1) 行ベクトル(Sh[wk0][0],Sh[wk0][1],Sh[wk0][2],Sh[wk0][3])の各要素が、0,1,2,3 の順列となるもの

(2) 行ベクトル(Sh[wk0][0],Sh[wk0][1],Sh[wk0][2],Sh[wk0][3])がベクトル(0,1,2,3)の巡回にならないもの

(3) 列ベクトル(Sh[0][i],Sh[1][i],...,Sh[15][i])の要素 n が等確率で出現する

(4) (1)(2)(3)の条件で残った 16 通りの行ベクトル(Sh[wk0][0],Sh[wk0][1],Sh[wk0][2],Sh[wk0][3])について、wk0 にハミングウェイト 1 の差分を与えた時に、Sh[wk0][i]に差分が出る確率がなるべく 1/2 になるように行番号を決定する

例えば、wk0 の差分が $0001_{(2)}$である第 0 行と第 1 行の要素(0,2,1,3)と(0,2,3,1)について各列毎の差分を見ると、第 0 列と第 1 列には差分がなく、第 2 列と第 3 列には差分がある

表 3.6 Sh テーブル

$$Sh = \begin{pmatrix} 0 & 2 & 1 & 3 \\ 0 & 2 & 3 & 1 \\ 0 & 3 & 1 & 2 \\ 0 & 3 & 2 & 1 \\ 1 & 0 & 3 & 2 \\ 1 & 2 & 0 & 3 \\ 1 & 3 & 0 & 2 \\ 3 & 1 & 0 & 2 \\ 3 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 2 & 0 & 3 & 1 \\ 3 & 0 & 2 & 1 \\ 1 & 3 & 2 & 0 \\ 2 & 1 & 0 & 3 \\ 2 & 1 & 3 & 0 \\ 3 & 1 & 2 & 0 \end{pmatrix}$$

3.9 Y関数

[入力] 入力データ X (32ビット)

[定数] 3種類の定数の組 Const[i][j]　　(i=0,1　j=0,1,2)

[出力] 出力データ Z (32ビット)

[処理] 入力データを定数の値だけシフトした後、もとの入力データと加算する。その加算結果をもとのデータとしてさらに2回繰り返す。

W0=X＋(X<<Const[i][0])

W1=W0+(W0<<Const[i][1])

Z=W1+(W1<<Const[i][2])

ただし、(Const[0][0],Const[0][1],Const[0][2])=(3,8,16)

(Const[1][0],Const[1][1],Const[1][2])=(7,9,13)

3種×2組の定数は以下の条件(優先順)で選択した。

(1) Const[1]の選択

F 関数一時鍵生成部の $SK^{[i]}$[1]加算後において

(A) 定数は、小さいものから順に並べる。

(B) ハミングウェイト 1 の差分を与えたときに、Y 関数(Const[1])直後の T 関数の入力となる 8 ビットそれぞれに差分が出る確率がなるべく 1/2 になるもの

(C) ハミングウェイト 2 の差分を与えたときに、Y 関数(Const[1])直後の T 関数の入力となる 8 ビットそれぞれに差分が出る確率がなるべく 1/2 になるもの

(D) ハミングウェイト 3 の差分を与えたときに、Y 関数(Const[1])直後の 2 つの T 関数の入力となる 16 ビットそれぞれに差分が出る確率がなるべく 1/2 になるもの

(E) ハミングウェイト 4 の差分を与えたときに、Y 関数(Const[1])直後の 2 つの T 関数の入力となる 16 ビットそれぞれに差分が出る確率がなるべく 1/2 になるもの

(2) Const[0]の選択

上記(1)で残った Const[1]を使用する。また $SK^{[i]}$[0]=0 とする。このとき、F 関数一時鍵生成部の $SK^{[i]}$[0]加算後において

(A) 定数は、小さいものから順に並べる。

(B) ハミングウェイト 1 の差分を与えたときに、Y 関数(Const[0])直後の T 関数の入力となる 8 ビットそれぞれに差分が出る確率がなるべく 1/2 になるもの

(C) ハミングウェイト 1 の差分を与えたときに、一時鍵生成部の出力ビット

それぞれに差分が出る確率がなるべく 1/2 になるもの
(D) ハミングウェイト 2 の差分を与えたときに、一時鍵生成部の出力ビット
それぞれに差分が出る確率がなるべく 1/2 になるもの
(E) ハミングウェイト 3 の差分を与えたときに、一時鍵生成部の出力ビット
それぞれに差分がでる確率がなるべく 1/2 になるもの
(F) Y 関数中の加算を排他的論理和とみなし、一時鍵生成部のどの出力ビット
にもなるべく多くの一時鍵生成部の入力ビットが出現するもの

3.10 鍵スケジューラ

[入力] 秘密鍵 Secret Key (128ビット)

[出力] F関数本流用拡大鍵：$FK^{\{i\}}$ (i=0,1,…,15) (64ビット×16段)

F関数一時鍵生成部用拡大鍵：$SK^{\{i\}}$ (i=0,1,…,15) (64ビット×16段)

L関数用拡大鍵：$LK^{\{j\}}$ (j=0,1,…,8) (128ビット×9)

[処理] 鍵スケジューラは複数のST関数(図 3.9)により構成される。

基本構造は、ST関数を4回ループ後、ST関数を32回繰り返しながら、拡大鍵を生成する。

生成する順序は、図 3.8、図 3.10、図 3.11に記述する。途中、$FK^{\{6\sim9\}}$[1]は、鍵スケジューラが単射な写像となるように、取り出している。

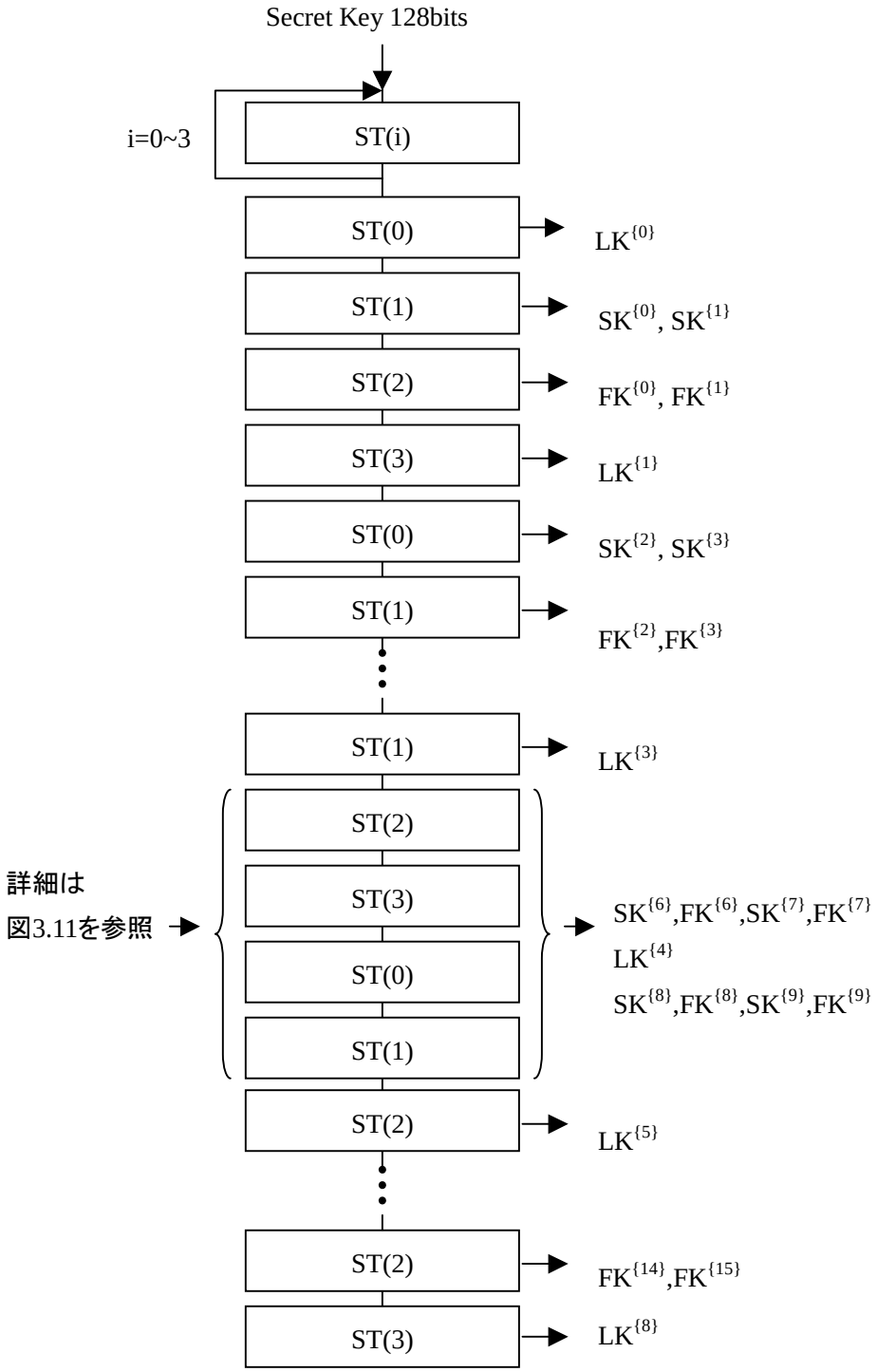

Secret Key 128bits
i=0~3
ST(i)
ST(0)
LK{0}
ST(1)
SK{0}, SK{1}
ST(2)
FK{0}, FK{1}
ST(3)
LK{1}
ST(0)
SK{2}, SK{3}
ST(1)
FK{2},FK{3}
ST(1)
LK{3}
詳細は
図3.11を参照
ST(2)
ST(3)
ST(0)
ST(1)
SK{6},FK{6},SK{7},FK{7}
LK{4}
SK{8},FK{8},SK{9},FK{9}
ST(2)
LK{5}
ST(2)
FK{14},FK{15}
ST(3)
LK{8}

図 3.8 鍵スケジューラ

3.11 ST関数

[入力] 入力データ $X = X_0 \| X_1 \| X_2 \| X_3$ (128ビット)

入力番号 n (n=0,1,2,3)

[出力] 出力データ $Z = Z_0 \| Z_1 \| Z_2 \| Z_3$ (128ビット)

[処理] T関数を用いたFeistel構造の入れ子構造である。T関数の入力番号はmodulo 4とする。

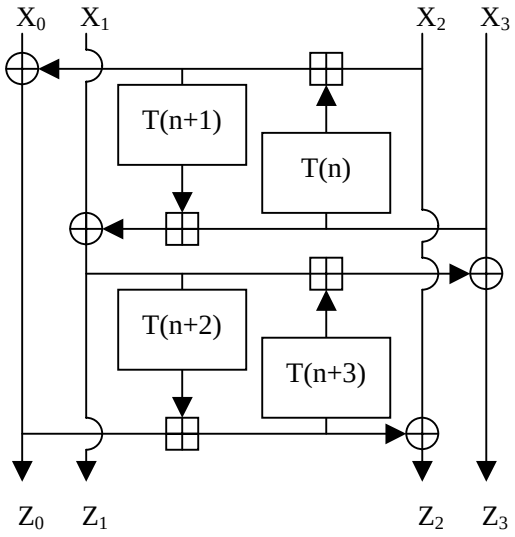
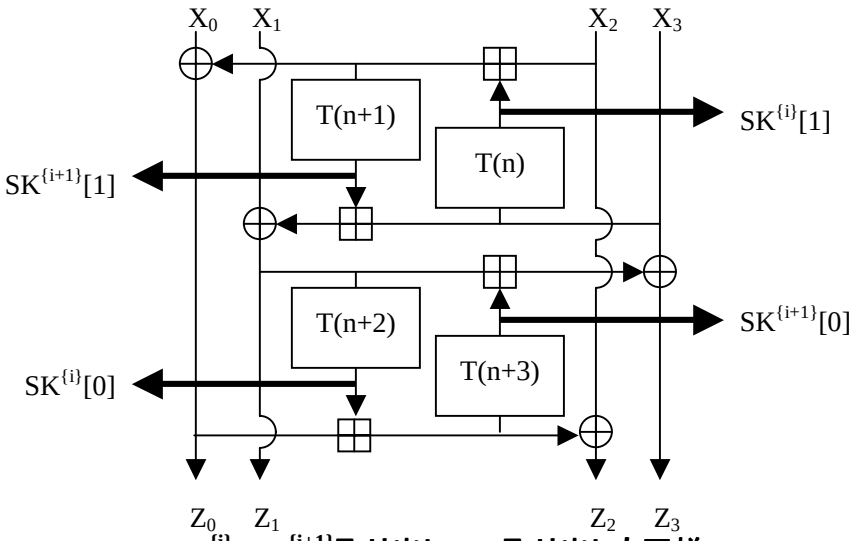


図 3.9 ST 関数



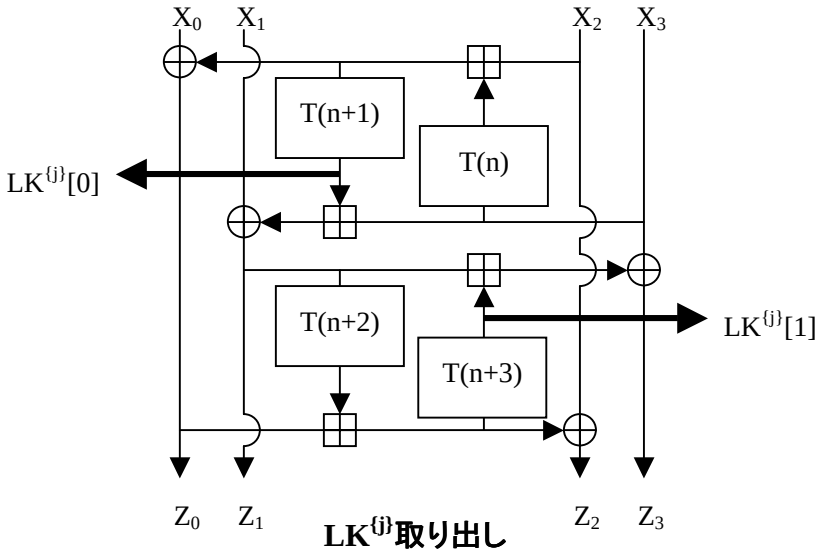


図 3.10 拡大鍵取り出し(1)

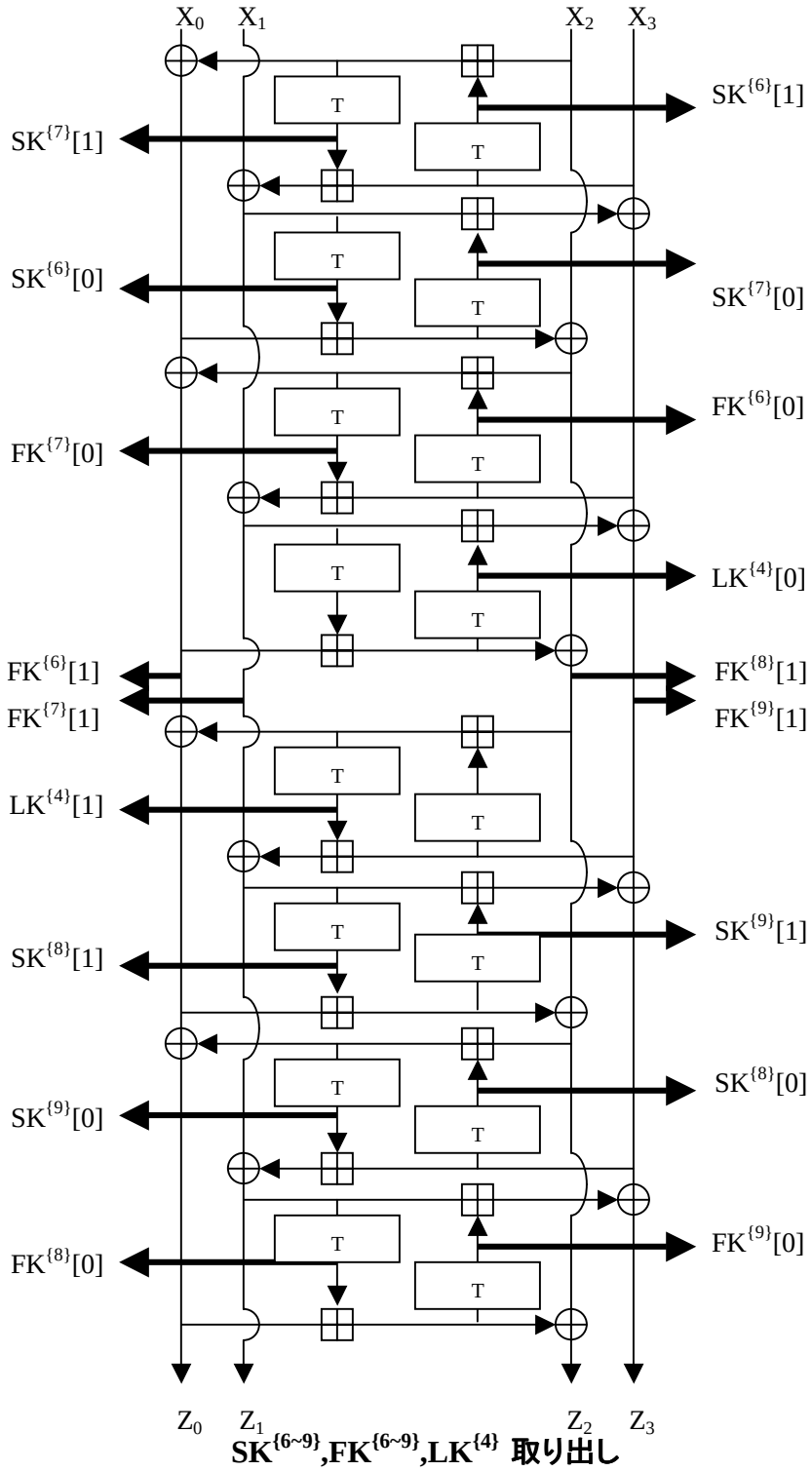


図 3.11 拡大鍵取り出し(2)

参考文献

[1] M. Matsui, “Linear Cryptanalysis Method for DES Cipher,” EUROCRYPT'93, LNCS765, pp.386-397, Springer-Verlag, 1994.

[2] E. Biham and A. Shamir, “Differential Cryptanalysis of DES-like Cryptosystems (Extended Abstract),” proceeding of CRYPTO'90, pp.2-21, 1990.

[3] T. Jakobsen and L.R. Knudsen, “The Interpolation Attack on Block Ciphers,” FSE'97, LNCS1267, pp.28-40, Springer-Verlag, 1997.

[4] L.R. Knudsen and T.A. Berson, “Truncated Differentials of SAFER,” FSE'96, LNCS1039, pp.15-25, Springer-Verlag, 1996.

[5] E. Biham, “New Types of Cryptanalytic Attacks Using Related Keys,” EUROCRYPT'93, LNCS765, pp.398-409, Springer-Verlag, 1994.

[6] E. Biham, “On Matsui's Linear Cryptanalysis,” EUROCRYPT'94, LNCS950, pp.341-355, Springer-Verlag, 1994.