暗号技術仕様書

CIPHERUNICORN-A

日本電気株式会社

1 概要

1.1 目的

本仕様書は、128 ビットブロック暗号 CIPHERUNICORN-A 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
IK_j	：初期/終期処理で使用する32ビット拡大鍵(j=0,1,...,7)
F^i	：第i段F関数(i=0,1,...,15)
FKa^i,FKb^i	：F^iの本流で使用する32ビット拡大鍵(関数鍵)
SKa^i,SKb^i	：F^iの一時鍵生成部で使用する32ビット拡大鍵(シード鍵)
EK^i	：F^iで使用する4つの拡大鍵FKa^i,SKa^i,FKb^i,SKb^iのまとまり
‖	：データの連結
∧	：論理積
⊕	：排他的論理和
⊞	：加算(mod 2^{32})
⊟	：減算(mod 2^{32})
⊗	：乗算(mod 2^{32})
x>>n	：xをnビット右論理シフト
x<<<n	：xをnビット左巡回シフト

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、

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

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

暗号の入出力仕様は AES(Advanced Encryption Standard)と同じブロックサイズ 128 ビット、秘密鍵長は 128 ビット、192 ビット、256 ビットの 3 種類を使用可能とする。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。

- 暗号化と復号が同じ構造
- 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に初期/終期処理を付加する。

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。

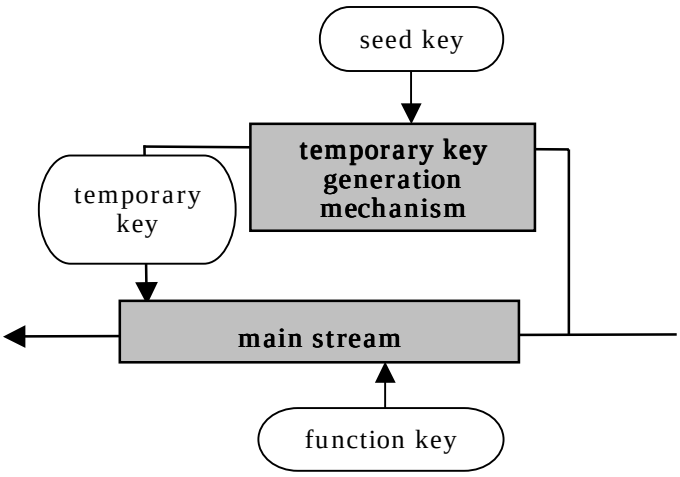

図 2.1 二重構造のラウンド関数

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、64 ビットの 3 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。

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

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-A は、データブロック長 128 ビット、秘密鍵長 128 ビット、192 ビット、256 ビットのいずれかを利用できる Feistel 構造の暗号である。

ラウンド数は 16 段、初期/終期処理では、拡大鍵の加算/減算を行う。

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

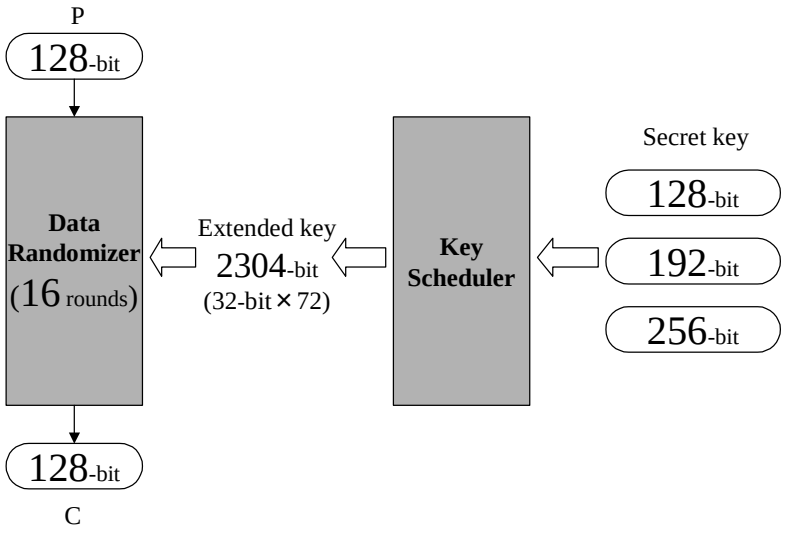

図 3.1 CIPHERUNICORN-A 全体構造

3.2 データ撹拌部

3.2.1 暗号化

[入力] 平文1ブロック $P=P_0 \| P_1 \| P_2 \| P_3$ (128ビット)

F関数用拡大鍵 $EK^i=FKa^i \| SKa^i \| FKb^i \| SKb^i$ (128ビット: i=0,1,...,15)

初期/終期処理用拡大鍵 IK_j (32ビット: j=0,1,...,7)

[出力] 暗号文1ブロック $C=C_0 \| C_1 \| C_2 \| C_3$ (128ビット)

[処理] 平文1ブロックを入力し、初期処理として拡大鍵加算(mod 2^{32})後、16段Feistel構造により撹拌した後、終期処理として拡大鍵減算(mod 2^{32})を行い、暗号文1ブロックを出力する。

for i=0,…,3 do
{
$W^0_i = P_i \boxplus IK_i$
}
for i=0,…,14 do
{
$W^{i+1}_0 \| W^{i+1}_1 = W^i_2 \| W^i_3$
$W^{i+1}_2 \| W^{i+1}_3 = (W^i_0 \| W^i_1) \oplus F^i(W^i_2 \| W^i_3, EK^i)$
}
$W^{16}_2 \| W^{16}_3 = W^{15}_2 \| W^{15}_3$
$W^{16}_0 \| W^{16}_1 = (W^{15}_0 \| W^{15}_1) \oplus F^{15}(W^{15}_2 \| W^{15}_3, EK^{15})$
for i=0,…,3 do
{
$C_i = W^{16}_i \boxminus IK_{i+4}$
}

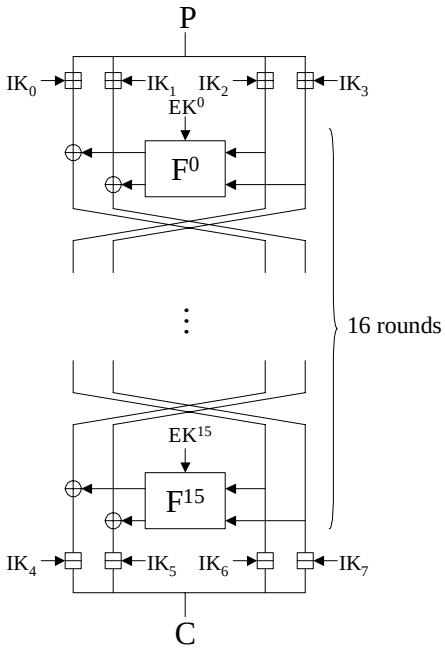

図 3.2 データ撹拌部(暗号化)

3.2.2 復号

[入力] 暗号文1ブロック $C=C_0 \| C_1 \| C_2 \| C_3$ (128ビット)

F関数用拡大鍵 $EK^i=FKa^i \| SKa^i \| FKb^i \| SKb^i$ (128ビット: i=0,1,...,15)

初期/終期処理用拡大鍵 IK_j (32ビット: j=0,1,...,7)

[出力] 平文1ブロック $P=P_0 \| P_1 \| P_2 \| P_3$ (128ビット)

[処理] 暗号文1ブロックを入力し、初期処理として拡大鍵加算(mod 2^{32})後、16段Feistel構造により撹拌した後、終期処理として拡大鍵減算(mod 2^{32})を行い、平文1ブロックを出力する。

for i=0,…,3 do
{
$W^0_i = C_i \boxplus IK_{i+4}$
}
for i=0,…,14 do
{
$W^{i+1}_0 \| W^{i+1}_1 = W^i_2 \| W^i_3$
$W^{i+1}_2 \| W^{i+1}_3 = (W^i_0 \| W^i_1) \oplus F^{15-i}(W^i_2 \| W^i_3, EK^{15-i})$
}
$W^{16}_2 \| W^{16}_3 = W^{15}_2 \| W^{15}_3$
$W^{16}_0 \| W^{16}_1 = (W^{15}_0 \| W^{15}_1) \oplus F^0(W^{15}_2 \| W^{15}_3, EK^0)$
for i=0,…,3 do
{
$P_i = W^{16}_i \boxminus IK_i$
}

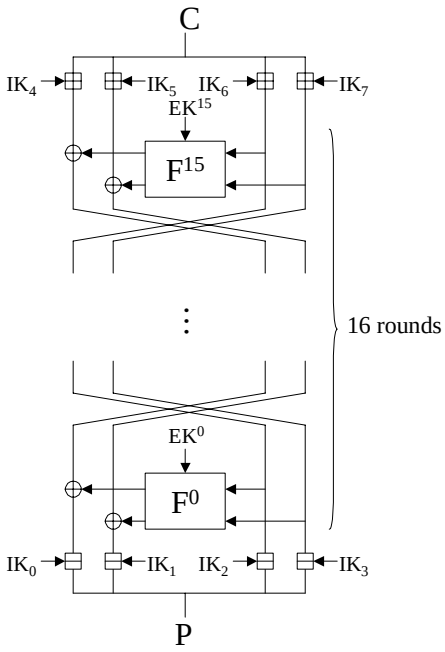

図 3.3 データ撹拌部(復号)

3.3 F関数

[入力] 入力データ $X = X_l \| X_r$ (64ビット)

F^i関数用拡大鍵 $EK^i = FKa^i \| SKa^i \| FKb^i \| SKb^i$ (128ビット)

[定数] 乗算定数 Const0＝0x7e167289, Const1＝0xfe21464b

[出力] 出力データ $Y = Y_l \| Y_r$ (64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。

本流では入力データ64ビットと拡大鍵FKa^i, FKb^iを加算し、A3関数、定数乗算、T_n関数を実行する。一時鍵生成部では、拡大鍵SKa^i, SKb^iと入力データ64ビットを加算し、定数乗算、T_n関数を通過後、一時鍵を取り出す。一時鍵を本流のデータ撹拌に使用し、出力データ64ビットを生成する。

$$WK^0_0 = SKa^i \boxplus X_r$$
$$WK^0_1 = SKb^i \boxplus X_l$$
$$WK^1_0 = WK^0_0 \otimes \mathrm{Const0}$$
$$WK^1_1 = WK^0_1 \oplus T_0(WK^1_0)$$
$$WK^2_1 = WK^1_1 \otimes \mathrm{Const1}$$
$$WK^2_0 = WK^1_0 \oplus T_0(WK^2_1)$$
$$WK^3_0 = WK^2_0 \otimes \mathrm{Const1}$$
$$WK^3_1 = WK^2_1 \oplus T_0(WK^3_0)$$
$$WK^4_1 = WK^3_1 \otimes \mathrm{Const0}$$
$$WK^4_0 = WK^3_0 \oplus T_0(WK^4_1)$$
$$WK^5_1 = WK^4_1 \oplus T_1(WK^4_0)$$
$$WK^5_0 = WK^4_0 \oplus T_1(WK^5_1)$$
$$WX^0_0 = FKa^i \boxplus X_l$$
$$WX^0_1 = FKb^i \boxplus X_r$$
$$WX^1_0 \| WX^1_1 = A3(WX^0_0 \| WX^0_1)$$
$$WX^2_0 = WX^1_0 \otimes \mathrm{Const0}$$
$$WX^2_1 = WX^1_1 \oplus T_0(WX^2_0)$$
$$WX^3_1 = WX^2_1 \otimes \mathrm{Const1}$$
$$WX^3_0 = WX^2_0 \oplus T_0(WX^3_1)$$
$$WX^4_1 = WX^3_1 \oplus T_1(WX^3_0)$$
$$WX^4_0 = WX^3_0 \oplus T_1(WX^4_1)$$
$$WX^5_1 = WX^4_1 \oplus T_2(WX^4_0)$$
$$WX^5_0 = WX^4_0 \oplus T_2(WX^5_1)$$
$$WX^6_1 = WX^5_1 \oplus T_3(WX^5_0)$$

$WX^6_0 = WX^5_0 \oplus T_3(WX^6_1)$

$k = (WK^5_1 >> 2) \wedge 0x3$

$WX^7_1 = WX^6_1 \oplus T_k(WX^6_0)$

$k = WK^5_1 \wedge 0x3$

$WX^7_0 = WX^6_0 \oplus T_k(WX^7_1)$

$Y_l = WX^7_0 \oplus WK^5_0$

$Y_r = WX^7_1 \oplus WK^5_0$

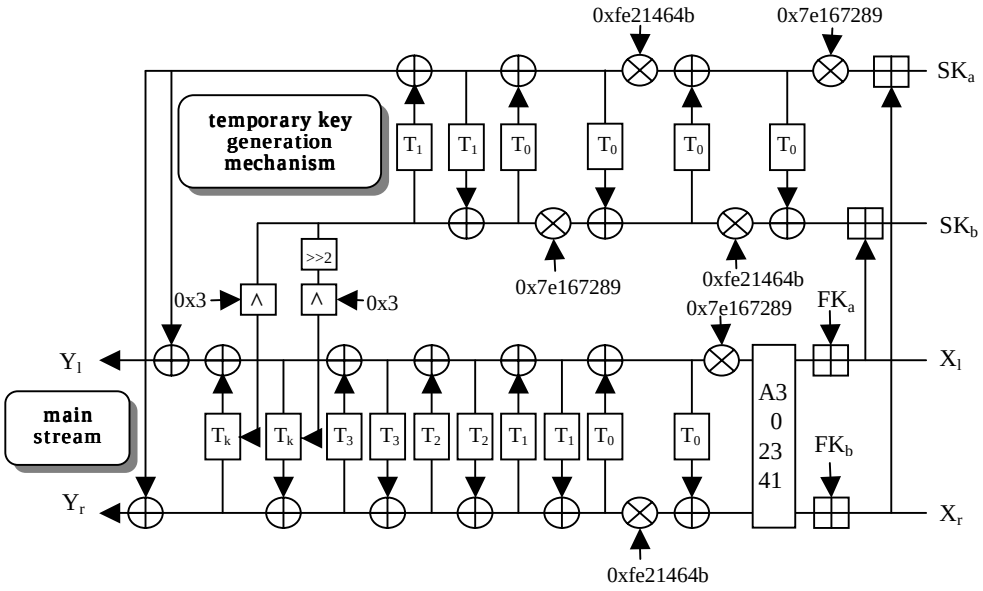

図 3.4 F 関数

3.4 A3関数

[入力] 入力データ X (64ビット)

[定数] 3種類の定数 const0=0, const1=23, const2=41

定数の決定基準

- 一つは0(性能考慮)
- 入力1ビットを求める逆演算に必要な出力ビットが最大(43)になる
- 出力を32ビットづつに分けたときのそれぞれ下位3バイトに入力64ビットがすべて出現する(以降の処理との関連)

上記基準を満たす定数を総当りで検索した結果、6組存在した。それらの組のなかで唯一両方が素数のものが存在したのでそれに決定した。

[出力] 出力データ Y (64ビット)

[処理] 入力データを指定した左巡回シフト数だけシフトした後、3つのデータの排他的論理和を求め、出力する。

$$Y = (X <<< const0) \oplus (X <<< const1) \oplus (X <<< const2)$$

3.5 定数乗算

[入力] 入力データ X (32ビット)

[定数] 乗算定数 Const0＝0x7e167289, Const1＝0xfe21464b

乗算定数の決定基準

- 奇数(全単射性の保持)
- ハミングウェイトが16(定数ビット数の1/2)
- T_0関数入力上位8ビットに入力データXの全ビットが影響する
- 乗算を「シフト＋排他的論理和」に置き換え3ビット以下の差分を与えたとき、T_0関数の入力(上位8ビット)差分が0または0xffにならない

上記基準を満たす定数を検索した結果、4種類の定数が存在した。それらの定数を使い実際に乗算し、入力差分と出力差分の間に高確率で成立する関係が少ない上記2種類に決定した。

[出力] 出力データ Y (32ビット)

[処理] 乗算定数と入力データを乗算し、結果を出力する。

$$Y = X \otimes Constn \qquad n=0,1$$

3.6 Tn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　Y (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、上位からS_0,S_1,S_2,S_3とする。

T_k関数の場合、kは0,1,2,3のいずれかが入力として与えられる。

$$Y = S_0(X_n) \| S_1(X_n) \| S_2(X_n) \| S_3(X_n)$$

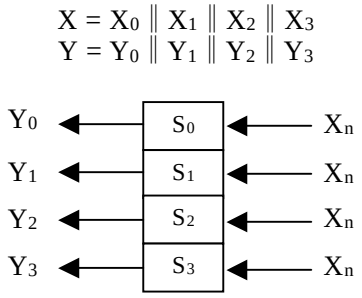

図 3.5 Tn 関数

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 Y (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。

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

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 鍵スケジューラ

[入力] 秘密鍵 $M = M_0 \| M_1 \| \ldots\ldots \| M_{LINE-1}$ (LINE=4(128ビット),
LINE=6(192ビット),
LINE=8(256ビット))

[出力] F関数用拡大鍵：EK^i (i=0,1,...,15) (128ビット×16段)
初期/終期処理用拡大鍵：IK_j (j=0,1,...,7) (32ビット×8)

[処理] 鍵スケジューラは複数のMT関数により構成される。
基本構造は、秘密鍵長により段数が変化する3回のダミーループ後、MT関数16段を9回繰り返す。1回の処理により、32ビット拡大鍵を8個生成する。ダミーループのMT関数段数は、秘密鍵128ビット、192ビット、256ビットのとき、それぞれ4段、6段、8段である。互換性はない。
生成した拡大鍵は図 3.12のように使用する。

cnt = 0
n = 16+2;
$W_i=M_i$ (i = 0,…,LINE-1)
for i = 0,..,2 do
{
  for j = 0,..,LINE-1 do
  {
    $W_j \| W_{(j+1)\%LINE} = MT(W_j \| W_{(j+1)\%LINE})$
  }
}
for i = 0,..,(16+2)/2-1 do
{
  for j = i*16 ,.., i*16+8-1 do
  {
    $W_{j\%LINE} \| W_{(j+1)\%LINE} = MT(W_{j\%LINE} \| W_{(j+1)\%LINE})$
  }
  for j = i*16+8 ,.., i*16+16-1 do
  {
    $W_{j\%LINE} \| W_{(j+1)\%LINE} = MT(W_{j\%LINE} \| W_{(j+1)\%LINE})$
    $WK[cnt++] = W_{(j+1)\%LINE}$
  }
}

IK_0 = WK[0];

IK_1 = WK[n];

IK_2 = WK[n*2];

IK_3 = WK[n*3];

IK_4 = WK[n -1];

IK_5 = WK[n*2-1];

IK_6 = WK[n*3-1];

IK_7 = WK[n*4-1];

for i = 0,.., 16-1 do

{

FKa^i = WK[1+i];

SKa^i = WK[n +1+i];

FKb^i = WK[n*2+1+i];

SKb^i = WK[n*3+1+i];

}

$M(128\text{-bit}) = M_0 \| M_1 \| M_2 \| M_3$ (M_n:32-bit)

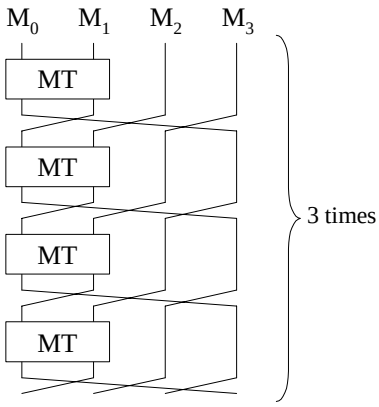

図 3.6 鍵スケジューラ ダミーループ（秘密鍵 128 ビット）

$M(192\text{-bit}) = M_0 \| M_1 \| M_2 \| M_3 \| M_4 \| M_5$ (M_n:32-bit)

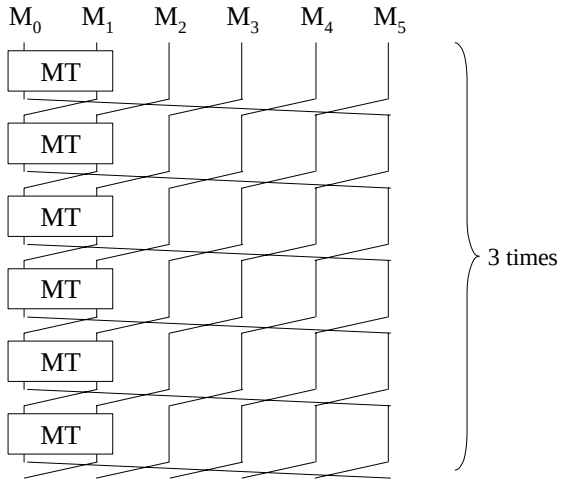

図 3.7 鍵スケジューラ ダミーループ（秘密鍵 192 ビット）

$M(256\text{-bit}) = M_0 \| M_1 \| M_2 \| M_3 \| M_4 \| M_5 \| M_6 \| M_7$ (M_n:32-bit)

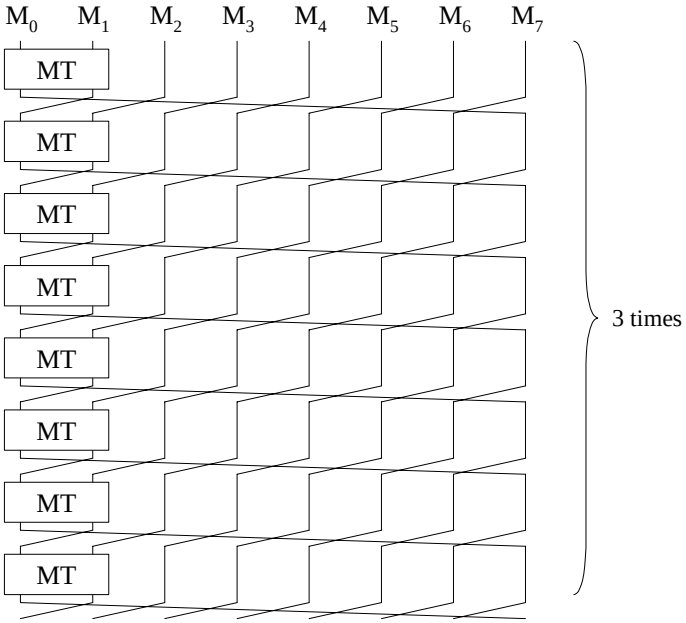

図 3.8 鍵スケジューラ ダミーループ（秘密鍵 256 ビット）

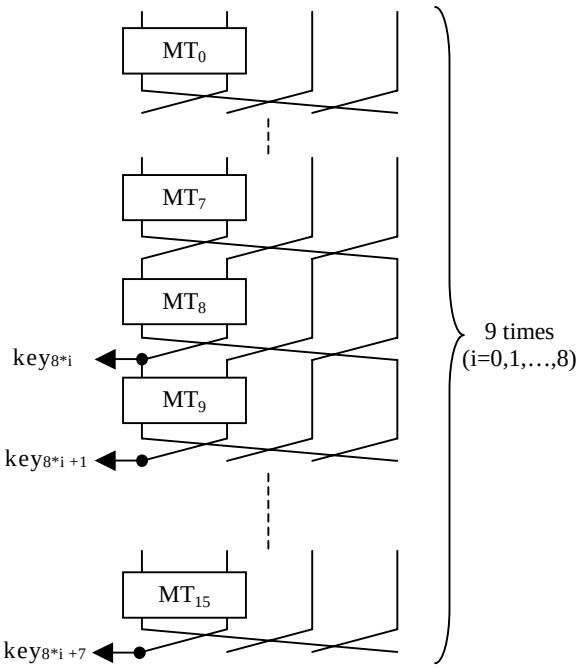

図 3.9 鍵スケジューラ 拡大鍵取り出し（秘密鍵 128 ビット）

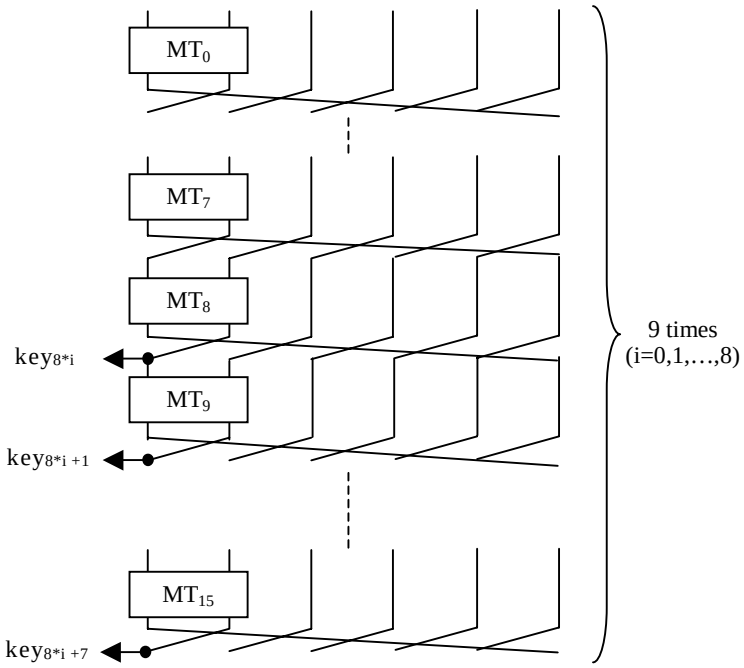

図 3.10 鍵スケジューラ 拡大鍵取り出し（秘密鍵 192 ビット）

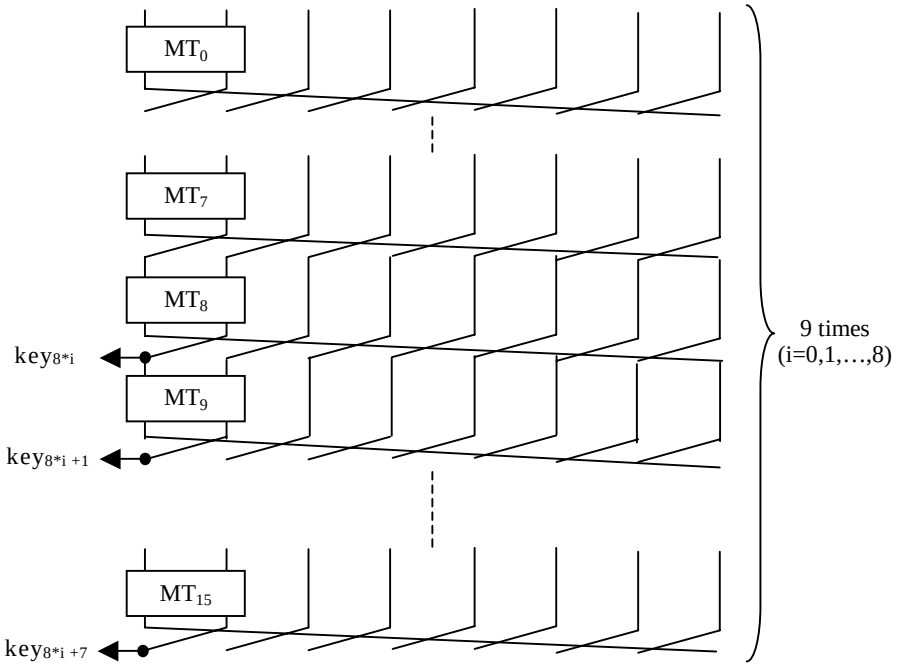

図 3.11 鍵スケジューラ 拡大鍵取り出し（秘密鍵 256 ビット）

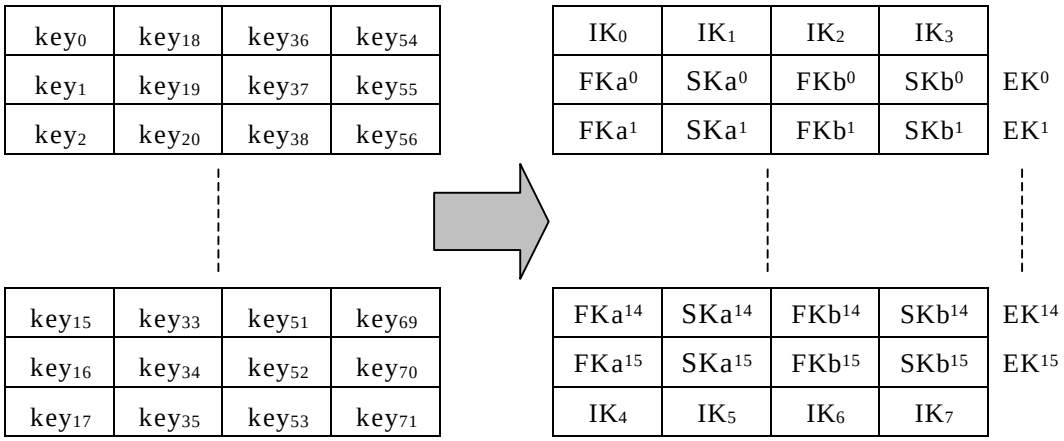
key0 key18 key36 key54
key1 key19 key37 key55
key2 key20 key38 key56
key15 key33 key51 key69
key16 key34 key52 key70
key17 key35 key53 key71
IK0 IK1 IK2 IK3
FKa0 SKa0 FKb0 SKb0 EK0
FKa1 SKa1 FKb1 SKb1 EK1
FKa14 SKa14 FKb14 SKb14 EK14
FKa15 SKa15 FKb15 SKb15 EK15
IK4 IK5 IK6 IK7

図 3.12 拡大鍵の対応

3.9 MT関数

[入力] 入力データ $X = X_0 \| X_1$ (64ビット)

[定数] 乗算定数 Const＝0x01010101

乗算定数の決定基準

- 奇数(全単射性の保持)
- 入力32ビットデータを乗算出力の上位8ビット(T_0関数の入力)に集める
- 上記2条件を満たすものでハミングウェイトが最小

上記基準を満たす定数として上記定数に決定した。

[出力] 出力データ $Y = Y_0 \| Y_1$ (64ビット)

[処理] 入力データの上位32ビットと定数Constとの乗算後、T_0関数の入力とする。T_0関数の出力を入力データの下位32ビットと排他的論理和し、出力とする。

$$Y_0 = X_0 \otimes \text{Const}$$
$$Y_1 = X_1 \oplus T_0(Y_0)$$

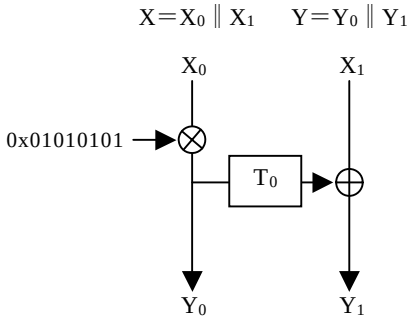

図 3.13 MT 関数

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.