Ｈｙ Ｒ Ａ Ｌ

(Hybrid Randomization Algorithm)

仕　様　書

㈱ローレルインテリジェントシステムズ

平田　耕蔵

目　　次

１．設計基準・設計方針

設計基準はＡＥＳと同等のスペックに対応できるものとしている。
すなわちデータブロック長１２８ｂｉｔ、キー長１２８ｂｉｔ、１９２ｂｉｔ、２５６ｂｉｔに対応している。

設計方針については、これまで発表されているブロック暗号の暗号モードとは異なるモードについての拡張性を有する事、その為の暗号化構成する諸機能について、特別な設計を行った。
ＨｙＲＡＬ（Hybrid Randomize Algorithm）と名づけた所以である。

２．諸元

アルゴリズム名：ＨｙＲＡＬ（Hybrid Randomize Algorithm）
カ　テ　ゴ　リ：共通鍵暗号技術ブロック暗号
概　　略：「主アルゴリズム」は、基本関数（***fi*** 関数）より組み立てた拡大関数を使用するアルゴリズムで、データ長１２８ビット、鍵長１２８ビット、１９２～２５６ビットをサポートしたもので、一般化フィステル構造を採ったものである。
秘密鍵を「鍵生成アルゴリズム」により中間鍵を生成、中間鍵よ「鍵割り当て」によりラウンド鍵と基本関数入力鍵が割り当てられる。
ラウンド鍵は主アルゴリズムに、基本関数入力鍵は拡大関数に適用される。
秘密鍵長より「主アルゴリズム」、「鍵生成アルゴリズム」、「鍵割り当て」はそれぞれ二種類存在する。
平 文 入 力 長：128bit
暗 号 出 力 長：128bit
基本関数入出力長：32bit／32bit
拡大関数入出力長：128bit／128bit
秘　密　鍵　長：128bit／192bit、256bit（２モード）
中　間　鍵　長：128bit
ラ ウ ン ド 鍵 長：128bit
基本関数入力鍵長：128bit

３．記号の説明（データ構造）

x（1bit）：1または0
x（8bit）：斜体小文字　*x* =（[MSb]x_7、x_6、x_5、x_4、x_3、x_2、x_1、x_0[LSb]）
X（32bit）：斜体太字 ***X*** =（[MSB]*x0*、*x1*、*x2*、*x3* [LSB]）
X（128bit）：大文字太字 **X** =（[MSW]***X0***、***X1***、***X2***、***X3*** [LSW]）
２．２．演算
⊕：Ｘｏｒ
＋：算術加算
≫i：iビット右シフト
8 bit データの乗算は、GF(2^8)の元を多項式表現し、法多項式x^8+x^4+x^3+x+1を使用。
多項式表現においては、0ビット目（最下位）をx^0の係数とする。

４．ＨｙＲＡＬの概要

４．１．ＨｙＲＡＬ全体構造

ＨｙＲＡＬの全体構造には鍵長 128bit の場合と、129～256bit が存在する。

４系列の一般化フィステル型で **G1**、**G2**、**F1**、**F2** と４つの異なるアルゴリズムの関数を接続している。これまでのフィステル型とは異なり拡大鍵を逆順に使用し同一のアルゴリズムで復号する方法ではなく、複合時には各々の逆関数を使用して復号する。

４．２．基本関数（ f_i 関数）

f_i（ i= 1、2、3、4、5、6、7、8）

入力 32bit 出力 32bit のものであり，入力のバイト転置により８種類ある。

入力バイトの転置後にＳ層およびＰ層がある。

［詳細は節５を参照］

４．３．拡大関数

G1、**G2**、**F1**、**F2**

基本関数（ fi 関数）を用いた入力 128bit 出力 128bit のものであり **G1**、**G2**、**F1**、**F2** の４種類の拡大関数がある。［詳細は節６を参照］

４．４．鍵生成部

OKi（128bit）：秘密鍵（i=0、1、2）

KMi（128bit）：中間鍵（i=1、2、3、4）

Rki（128bit）：ラウンド鍵（i=1、2、3、4、5、6、7、8、9）

IKi（128bit）：基本関数入力鍵（i=1、2、3、4、5、6）

鍵処理モードでは **OKi** と定数 **Li**（i= 1、2、3）を **G1**、**G2** 関数に入力しその出力を **KMi** とする。

拡大鍵の割り当てでは **KMi** から拡大キーである **RKi** と **IKi** を生成する。［詳細は節８を参照］

５．基本関数（ f_i 関数 ）

f_i 関数は、SP 構造の 32bit 入出力関数で、S 層では最大線形・差分確率が 2^{-6} の８ﾋﾞｯﾄｂｉｔ「S-box」を４並列で用いる。P 層は４バイト入出力のMDS行列とする。（**図１**）

また、関数に入力される直前に転置を行い、その転置の仕方が８通りあるので、転置種類毎に **T1**～**T1** と表記する。

従って、**T1** の転置に対応する f_i 関数は f_1、**T2** の転置対応する f_i 関数は f_2 となり、f_i 関数は f_1～ f_8 の８種類が存在する。

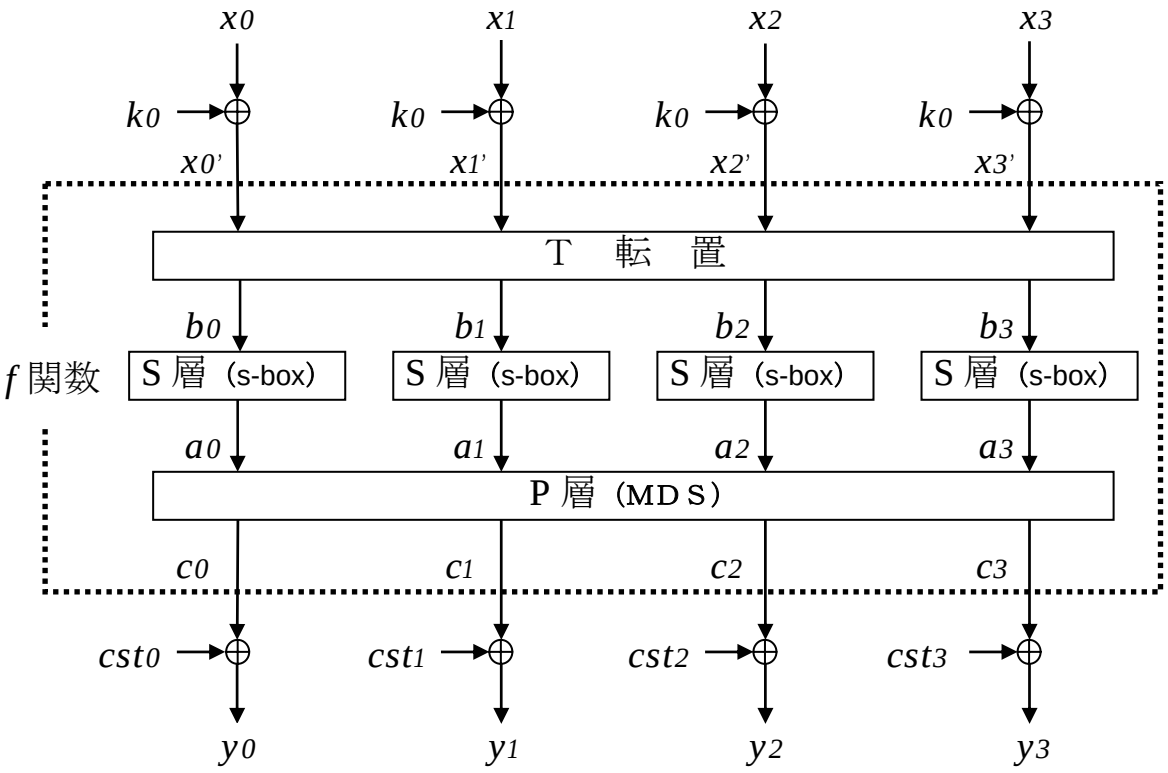


図１： f_i 関数

ここで、*k0*、*k1*、*k2*、*k3* は、f_i 関数入力鍵 ***IKi*** の 8bit 要素。***IKi*** =(*k0*、*k1*、*k2*、*k3*)。

cst0、*cst1*、*cst2*、*cst3* は、定数 0x11、0x22、0x44、0x88 であり、特異点（All Null）を解消するためのものである。

５．１．転置

f_1～ f_8 に対応する転置 **T1**～**T8** は、**図２**の通りである。

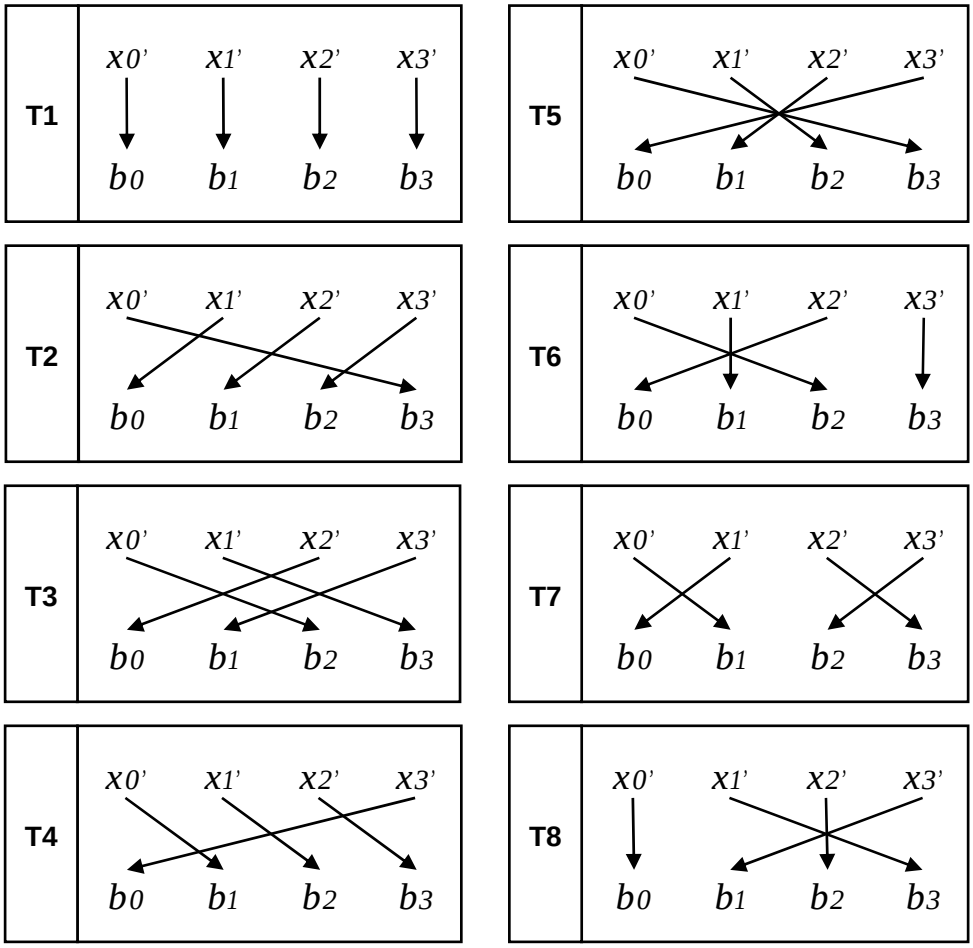


図２：T 転置

５．２．Ｓ層

転置された入力 *b0*、*b1*、*b2*、*b3* は、**テーブル１**の s-box により変換され *c0*、*c1*、*c2*、*c3* として出力される。

s-box		*b0* or *b1* or *b2* or *b3* の下位 4bit															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b0 or *b1* or *b2* or *b3* の上位 4bit	0	16	5e	d3	af	36	43	a6	49	33	93	3b	21	91	df	47	f4
	1	b6	70	06	d0	81	82	fa	a1	10	b5	3c	ba	97	85	b7	79
	2	ed	5c	ca	05	87	bf	24	4c	51	ec	17	61	22	f0	3e	18
	3	a7	64	13	ab	e9	09	25	54	2d	31	69	f5	37	67	fe	1d
	4	0b	28	a3	2f	e4	0f	d4	da	1b	fc	e6	ac	53	04	27	a9
	5	94	8b	d5	c4	90	6b	f8	9d	c5	db	ea	e2	ae	63	07	7a
	6	5b	23	34	38	03	8c	46	68	cd	1a	1c	41	7d	a0	9c	dd
	7	08	4e	e3	d7	1e	b3	50	5d	c6	0e	ad	cf	d6	eb	0d	b1
	8	fb	7c	c3	2e	65	48	b8	8f	ce	e7	62	d2	12	4a	c8	26
	9	a5	8e	3d	76	86	57	bc	bd	11	75	71	78	1f	ef	e0	0c
	A	de	6a	6d	32	84	72	8a	d8	f9	dc	9a	89	9f	88	14	2a
	B	9b	9e	d9	95	b9	a4	02	f7	96	73	56	be	7f	80	7e	83
	C	00	01	f6	8d	7b	d1	52	cb	b0	e1	c7	e5	29	c0	4f	e8
	D	58	3f	cc	fd	ee	b2	40	ff	99	2b	5f	60	aa	4b	b4	74
	E	2c	45	6c	92	66	42	39	f3	77	bb	19	59	20	6f	35	f2
	F	c1	0a	15	98	a2	c2	44	30	55	4d	c9	a8	5a	f1	6e	3a

テーブル１：s-box

５．３．s-box の生成

s-box は非線形層である GF(2⁸)上の逆関数 $s = z^{-1}$(ただし $z^{-1}=0$)と線形関数であるアフィン変換**式１**とグレイコード変換**式２**の組み合わせで構成されたものが前述の s-box となる。

生成の過程を**図３**で表す。

s-box の最大線形及び最大差分確率は 2^{-6} である。

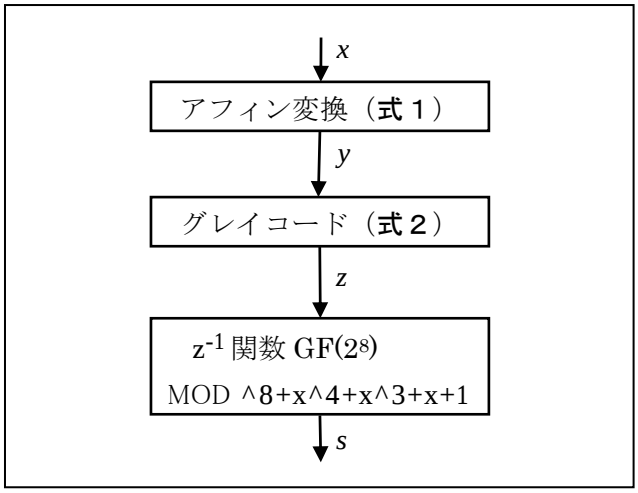


図３：s-box の生成

$$\begin{pmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

式１： アフィン変換 $y=(x+64) \mathrm{MOD} 256$

$$\begin{pmatrix} z_7 \\ z_6 \\ z_5 \\ z_4 \\ z_3 \\ z_2 \\ z_1 \\ z_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix}$$

式２： グレイコード $z = y \oplus (y \gg 1)$

５．４．Ｐ層

Ｓ層で出力された *a0*、*a1*、*a2*、*a3* は、ＭＤＳ行列式により *c0*、*c1*、*c2*、*c3* が出力される。（**式３**）

$$\begin{pmatrix} c0 \\ c1 \\ c2 \\ c3 \end{pmatrix} = \begin{pmatrix} 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 \\ 7 & 3 & 1 & 2 \\ 7 & 4 & 5 & 3 \end{pmatrix} \begin{pmatrix} a0 \\ a1 \\ a2 \\ a3 \end{pmatrix}$$

式３：ＭＤＳ行列式

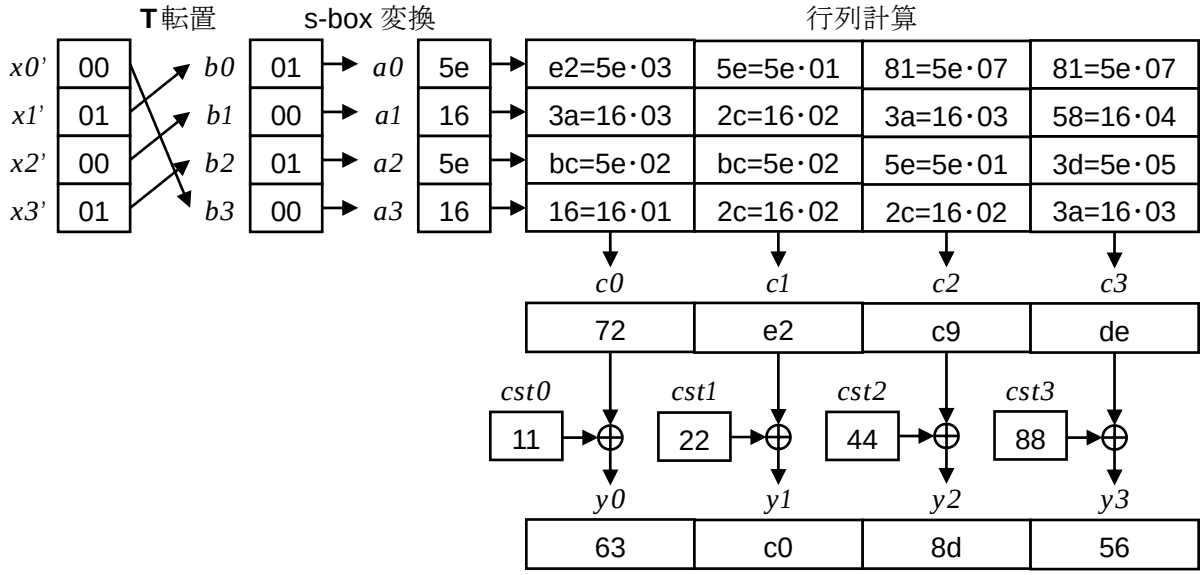
５．５．基本関数（ f_i 関数 ）の実装

32bit を一度に計算できる計算機において実際のプログラムは、Ｓ層の入力値 *bi*（8bit）に対して、MDS 行列式の列に対応する結果（32bit）はあらかじめ計算しておく事が出来るので、16×16 の 32bit データテーブルを行列式の各行に応じて４種類を作成しておく。

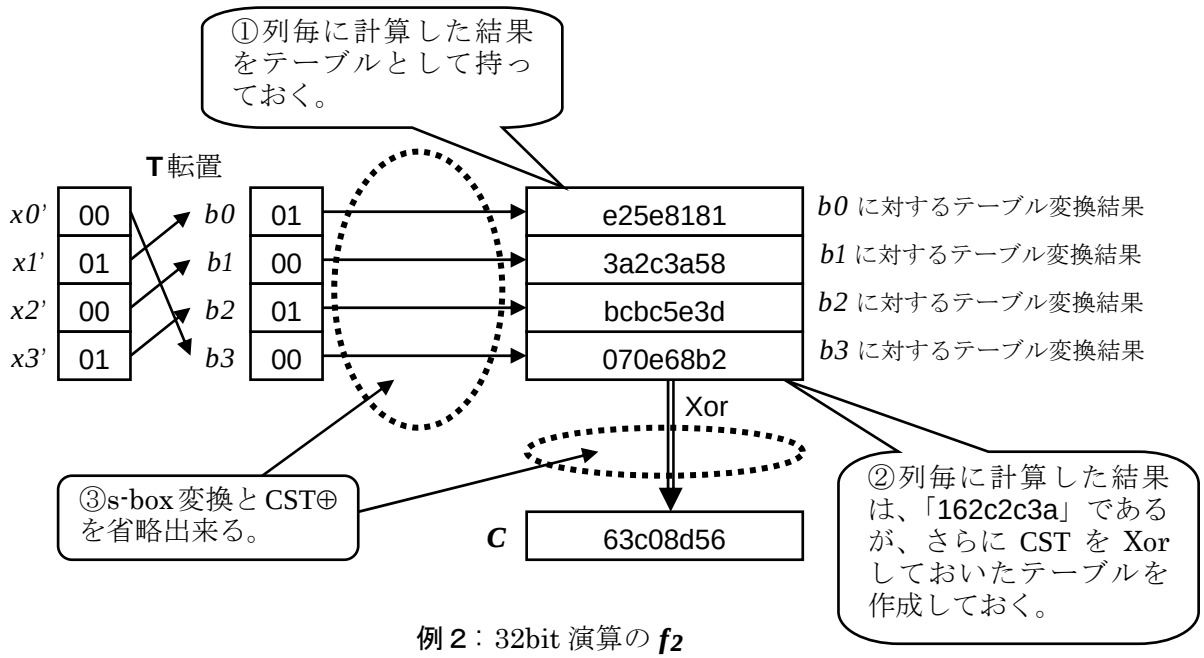
*xi'*を転置させ、あらかじめ作成しておいた四つのテーブルより行毎の計算結果を得る事が出来る。

この方法により、*bi* から s-box 変換した *ai* を作成するこが省略出来る。

8bit 単位で演算できる計算機の場合と、32bit 単位で演算できる計算機の場合の f_2 関数の計算例を示す。



例１：8bit 演算の f_2



例２：32bit 演算の f_2

６．拡大関数 **G1**、**G2**、**F1**、**F2**

６．１．**G** 関数

G 関数は各ラウンドで f_i 関数を１回使用する。

G1 関数および **G1**$^{-}$関数をＰ７**図４・５**に、**G2** 関数および **G2**$^{-}$関数をＰ７**図６・７**に示す。

逆関数は、暗号アルゴリズムの下端を入力として計算し、使用する f_i 関数は暗号と同じものを使用する。

６．２．**F** 関数

F 関数は各ラウンドで f_i 関数を２回使用する。

F1 関数および **F1**$^{-}$関数をＰ８**図８・９**に、**F2** 関数および **F2**$^{-}$関数をＰ８**図１０・１１**に示す。

逆関数は、**G** 関数と同様に暗号アルゴリズムの下端を入力として計算し、使用する f_i 関数は暗号と同じものを使用する。

F 関数においては中間鍵が入力され、$IKi,0$、$IKi,1$、$IKi,2$、$IKi,3$ は **IKi の** 32bit **要素である。**

※IKi=（[MSW] $IKi,0$、$IKi,1$、$IKi,2$、$IKi,3$ [LSW]）

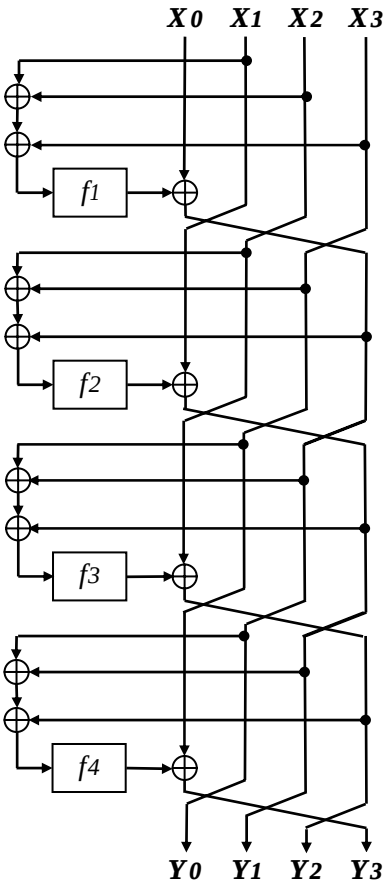


図４：**G1** 関数（暗号）

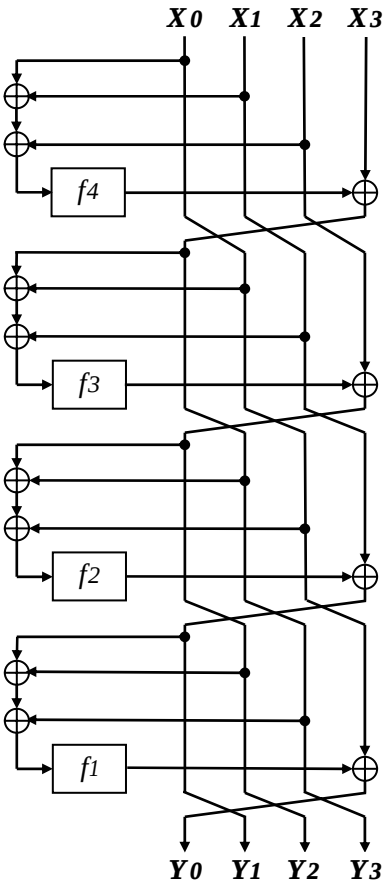


図５：**G1^{-1}** 関数（復号）

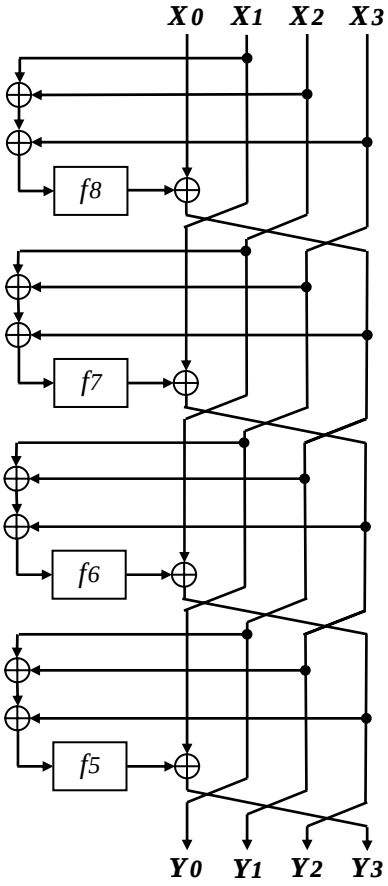


図６：**G2** 関数（暗号）

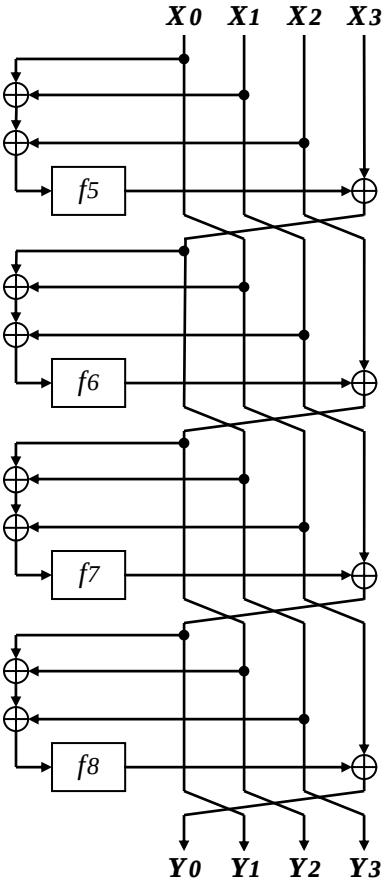


図７：**G2^{-1}** 関数（復号）

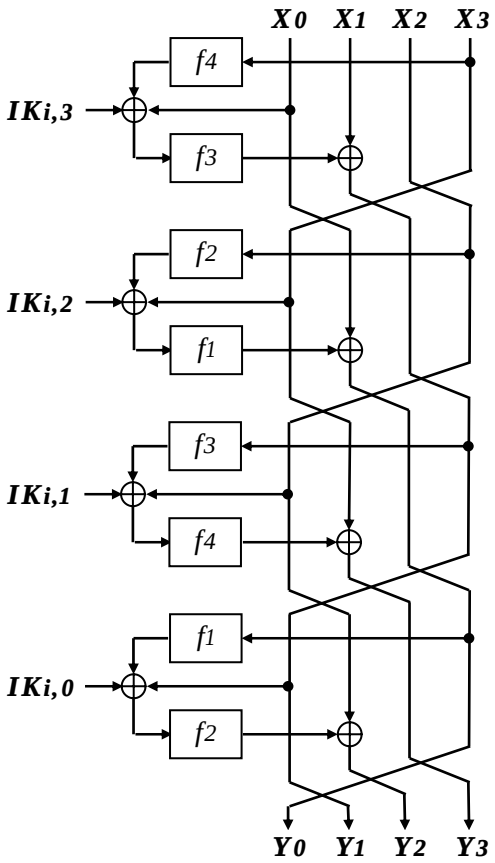


図８：**F1** 関数（暗号）

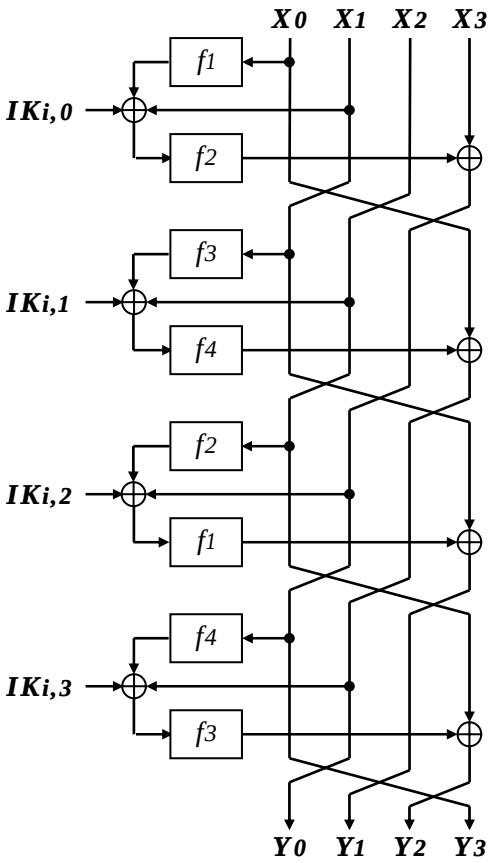


図９：**F1**$^{-1}$ 関数（復号）

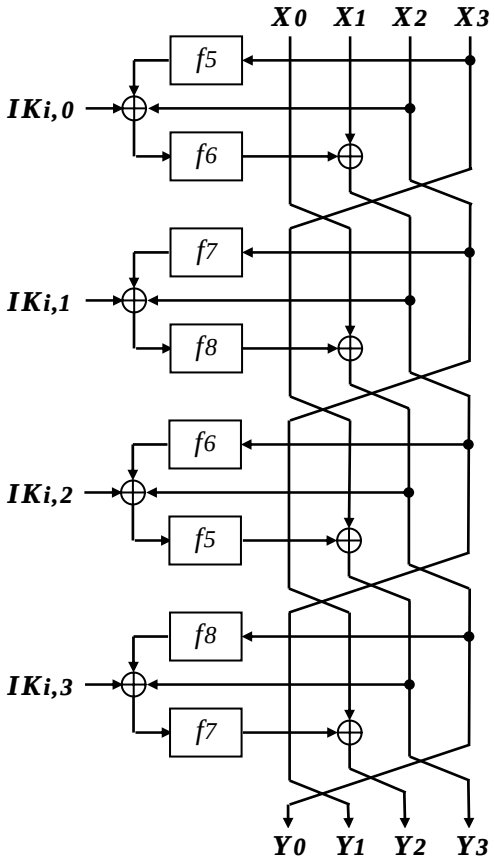


図１０：**F2** 関数（暗号）

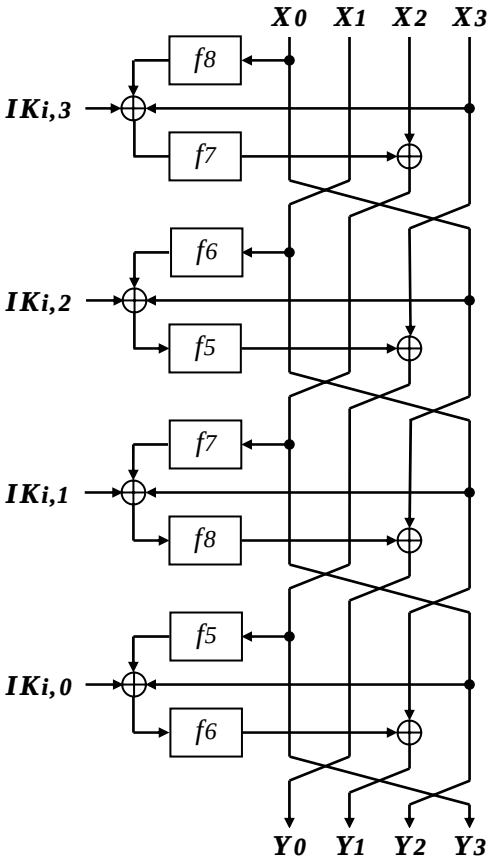


図１１：**F2**$^{-1}$ 関数（復号）

7．ＨｙＲＡＬの全体構造

HyRAL の鍵長 128bit の場合の暗号を**図１２**、復号**を図１３**にて示し、192、256bit の場合の暗号を**図１４**、復号**を図１５**にて示す。

鍵長 128bit の場合は、ラウンド鍵 **RK1**～**RK6** の６種類と f_i 関数入力鍵 **IK1**～**IK4** の４種類を使用する。

鍵長 192、256bit の場合は、ラウンド鍵 **RK1**～**RK9** の９種類と f_i 関数入力鍵 **IK1**～**IK6** の６種類を使用する。

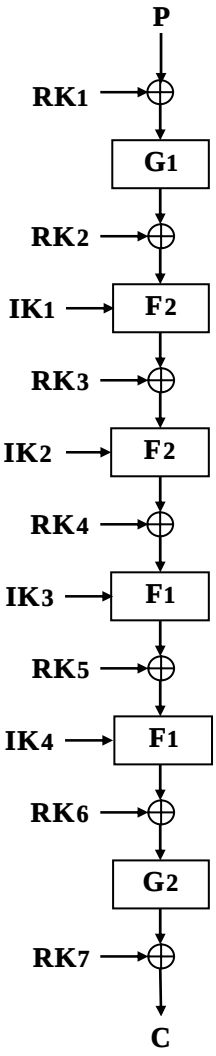


図１２

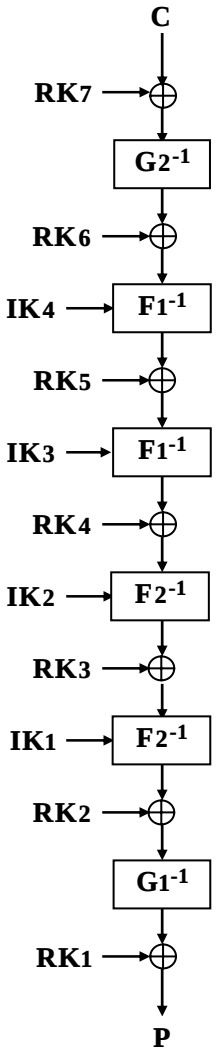


図１３

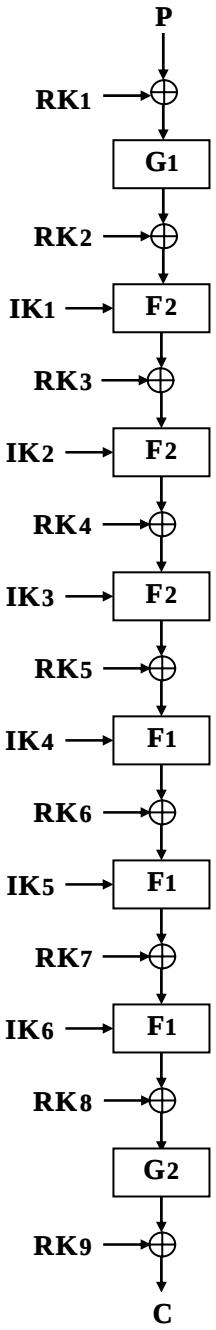


図１４

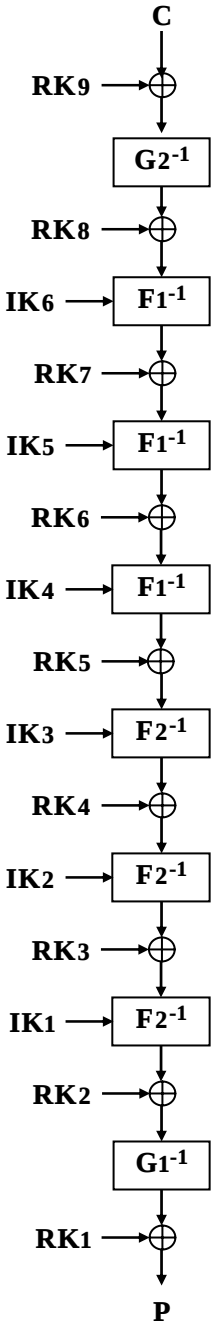


図１５

８．鍵処理モード

G1 、**G2** 関数を使用して、秘密鍵 **OK** から中間鍵 **KMi**(Key material)を作成し、それを元に、暗号化変換の各ラウンドにおいて使用する拡大鍵(**RKi**、**IKi**)を生成する。秘密鍵長が 128 bit 時と、192bit および 256 bit 時では、異なる鍵処理モードを使う。前者を Single Key Mode,後者を Double Key Mode と呼ぶ。

Double Key Mode の場合、秘密鍵 **OK** は **OK1** と **OK2** に分けて処理する。

８．１．Single Key Mode（Original Key ”OK”128bit）

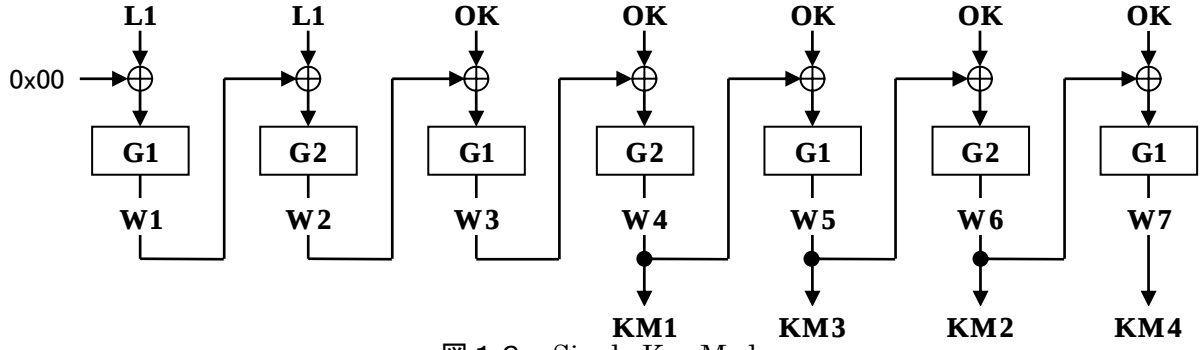


図１６：Single Key Mode

ここで、**W1**～**W7** は **KMi** 生成途中で作成される中間データ。

L1 は LABEL と呼ぶ定数であり、以下の値をとる。

L1 = (48 59 52 41 4C 40 40 40 00 00 00 00 00 00 00 12)

８．２．Double Key Mode（Original Key ”OK” 192bit および 256 bit）

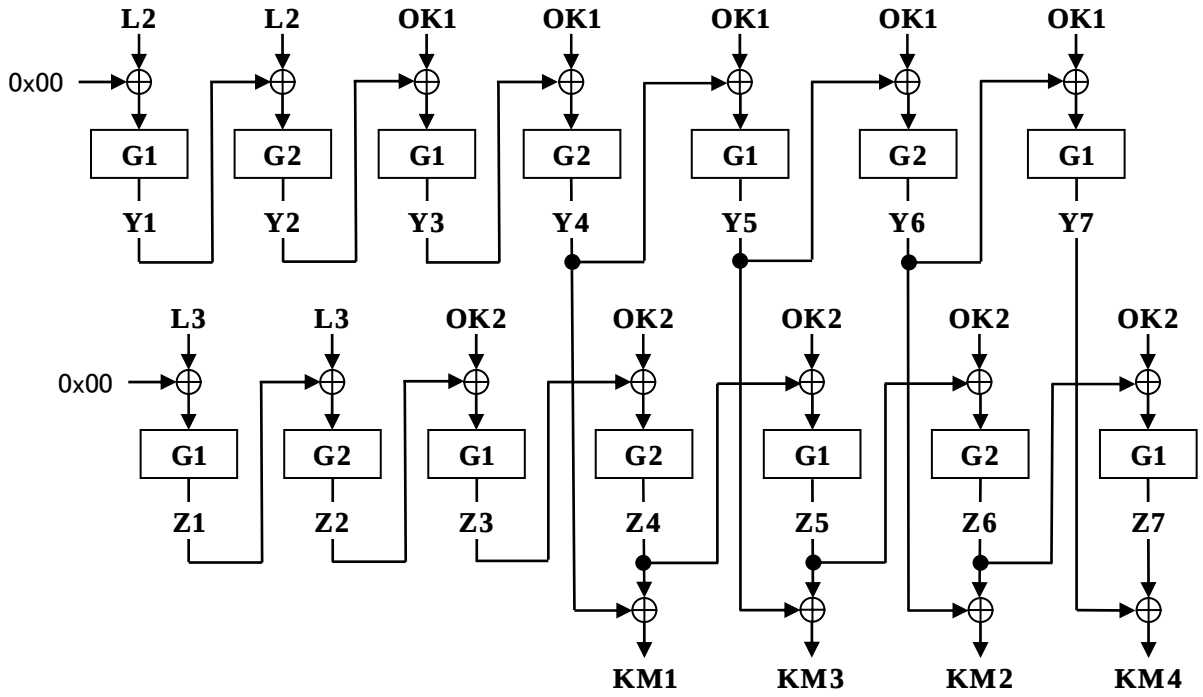


図１７：Double Key Mode

ここで、**Y1**～**Y7**、**Z1**～**Z7** は **KMi** 生成途中で作成される中間データ。

L2、**L3** は LABEL と呼ぶ定数であり以下の値をとる。

L2 = (48 59 52 41 4C 40 40 40 00 00 00 00 00 00 00 22)

L3 = (48 59 52 41 4C 40 40 40 00 00 00 00 00 00 00 32)

８．４．中間鍵生成アルゴリズムの実装

LABEL が定数であり **W2、Y2、Z2** は事前計算可能である。

実装時には **L1**、**L2**、**L3** を格納するのではなく、**W2、Y2、Z2** を格納する。

W2、Y2、Z2 の事前計算結果は下記の通りである。

W2 = (62 8c cd a0 3b 15 65 c1 3b ad 2d 4f b8 80 6a c5)

Y2 = (f9 25 1a 23 65 cd 3c 2e 80 66 cb bb fe 31 6b 7b)

Z2 = (5d e2 86 25 65 6b 71 ff 9f fb 1e 12 ee f1 27 f5)

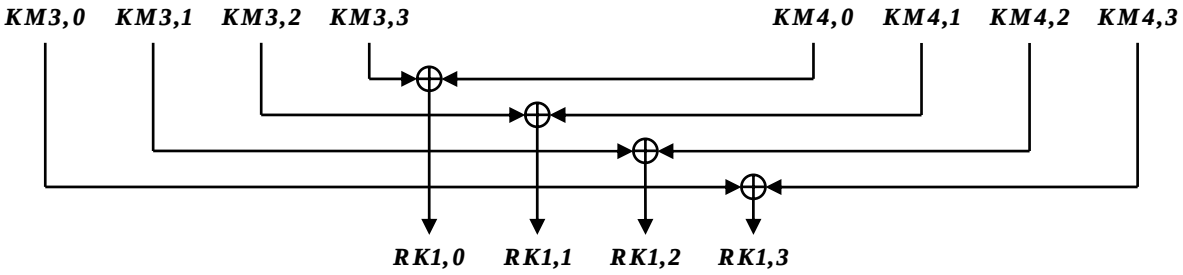
９．拡大鍵の割り当て

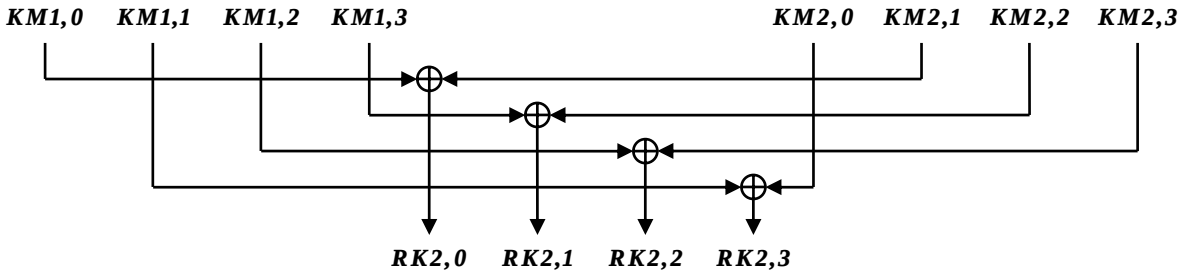
Original Key を入力として生成した Key Material(**KMi**)を使用して実際の Sub Key を合成、割り当てる。

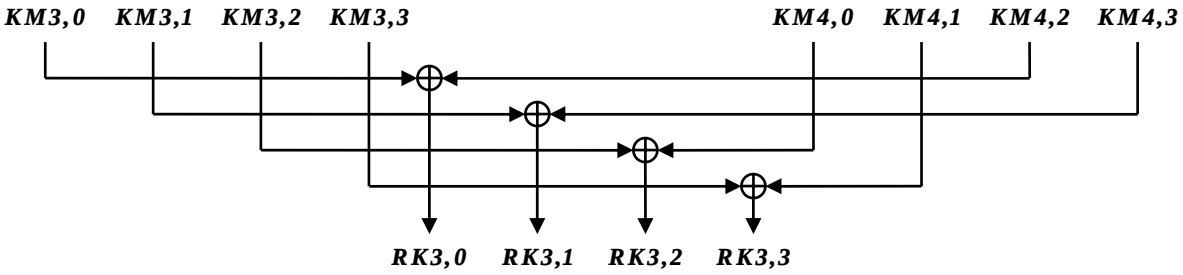
各ラウンドの拡大鍵 **RKi**、**IKi** は、中間鍵 **KMi** を利用して以下のように求めるが、各々の拡大鍵は **KMi** の二つを合成(Xor)して割り当てるが、**KMi** を 32bit 単位で区切った４つのブロックを入れ替えて Xor をとる。

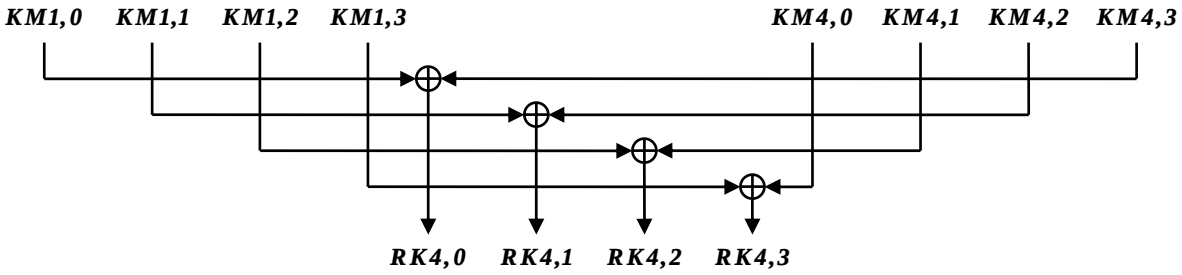
９．１．Single Key Mode

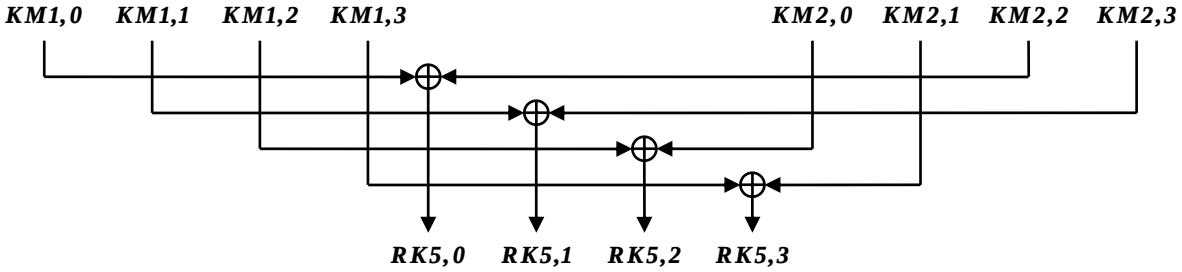
鍵長 128bit の場合、**RK1**～**RK7**、**IK1**～**IK4** は、以下の図の様に求める。

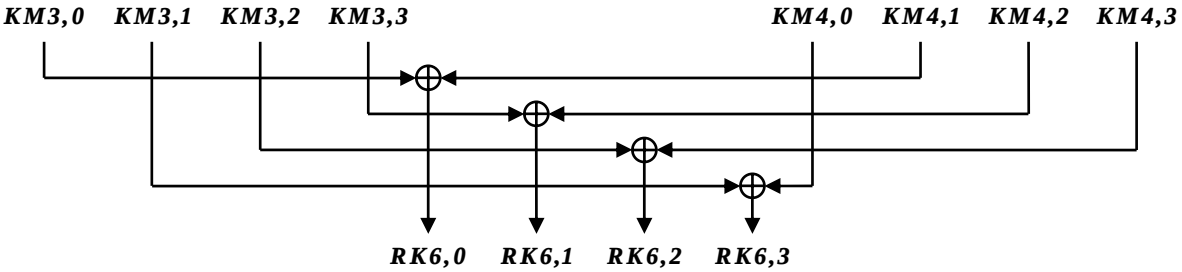


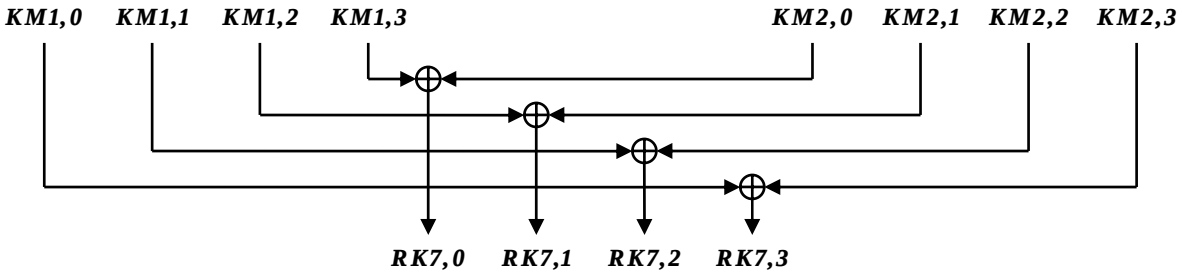


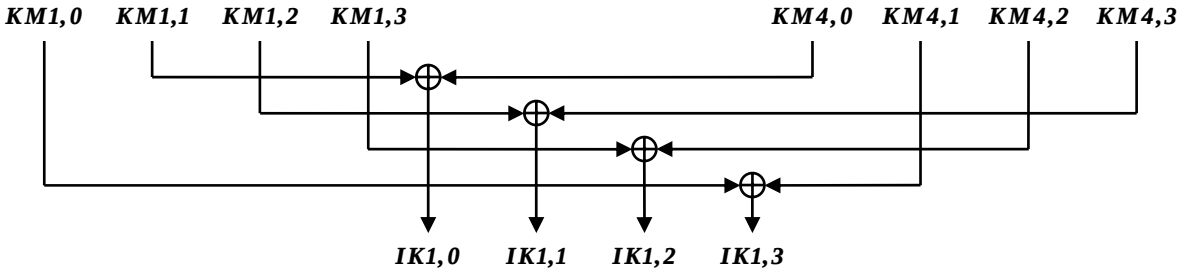


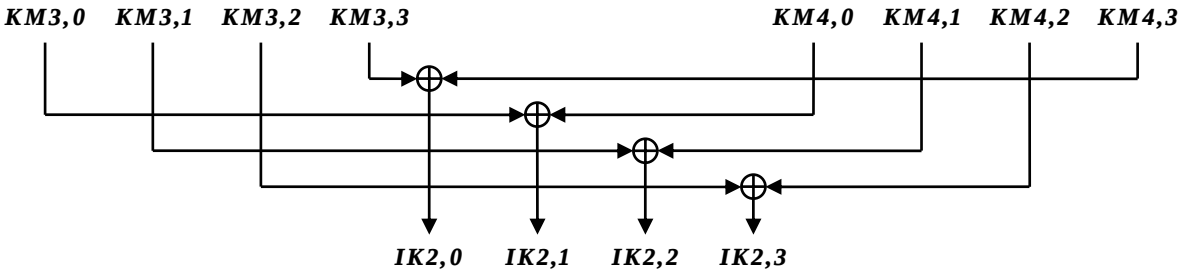

KM1,0 KM1,1 KM1,2 KM1,3 KM4,0 KM4,1 KM4,2 KM4,3
RK4,0 RK4,1 RK4,2 RK4,3

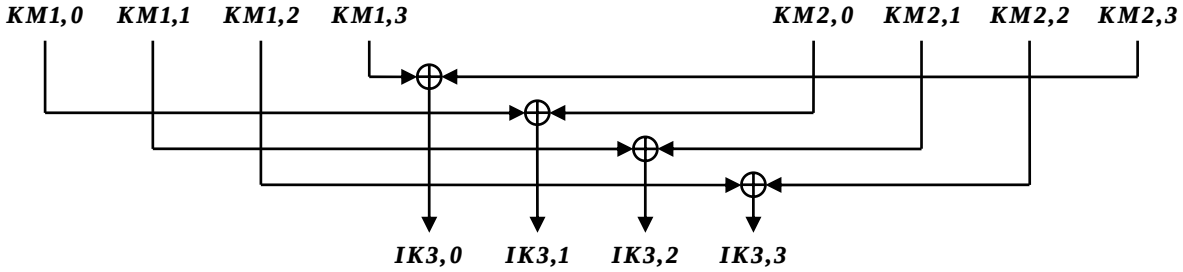

KM1,0 KM1,1 KM1,2 KM1,3 KM2,0 KM2,1 KM2,2 KM2,3
RK5,0 RK5,1 RK5,2 RK5,3

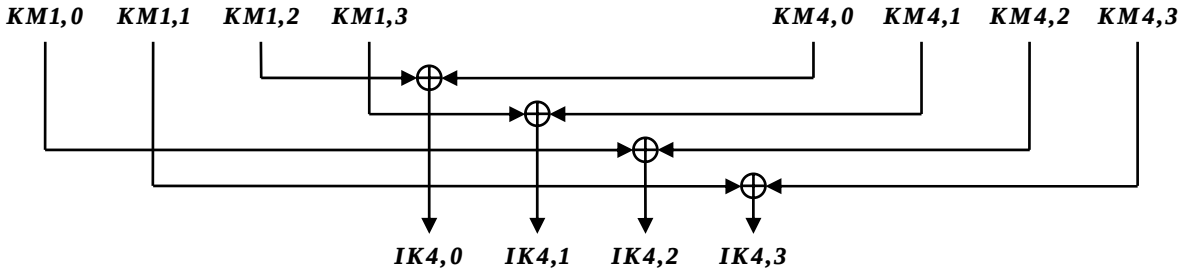

KM3,0 KM3,1 KM3,2 KM3,3 KM4,0 KM4,1 KM4,2 KM4,3
RK6,0 RK6,1 RK6,2 RK6,3


KM1,0 KM1,1 KM1,2 KM1,3 KM2,0 KM2,1 KM2,2 KM2,3
RK7,0 RK7,1 RK7,2 RK7,3


KM1,0 KM1,1 KM1,2 KM1,3 KM4,0 KM4,1 KM4,2 KM4,3
IK1,0 IK1,1 IK1,2 IK1,3

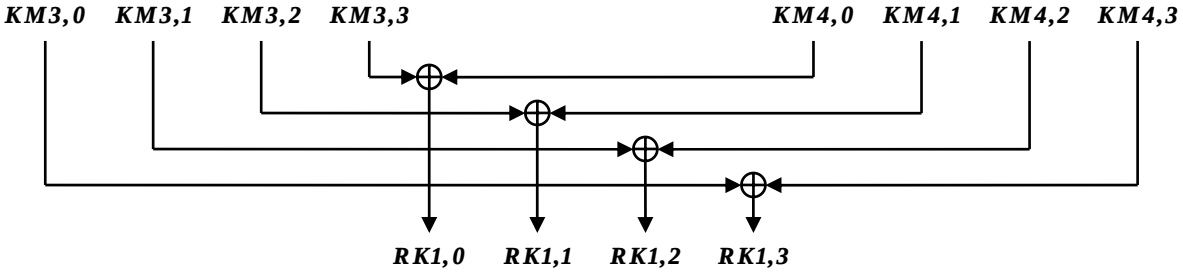


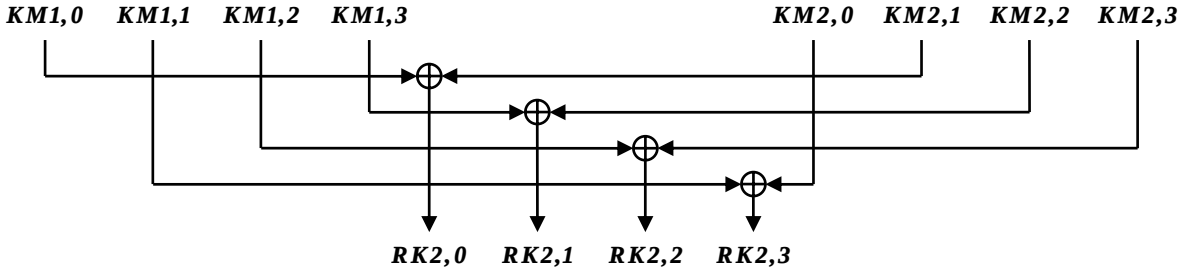


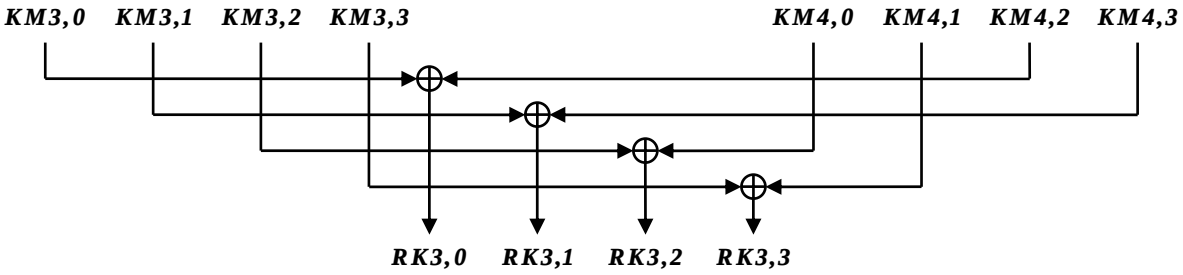


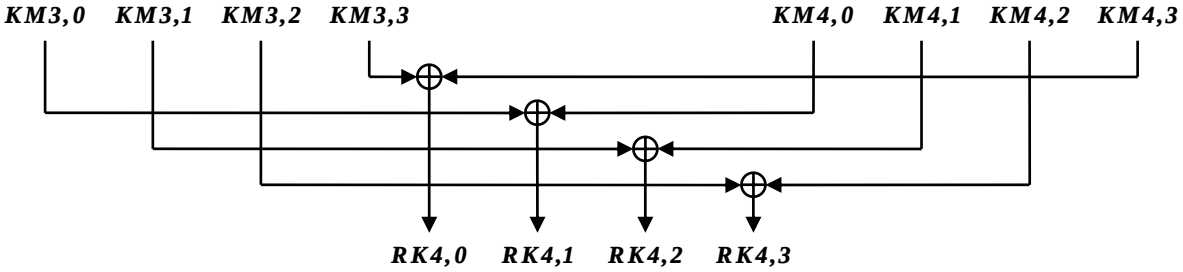
9．２．Double Key Mode

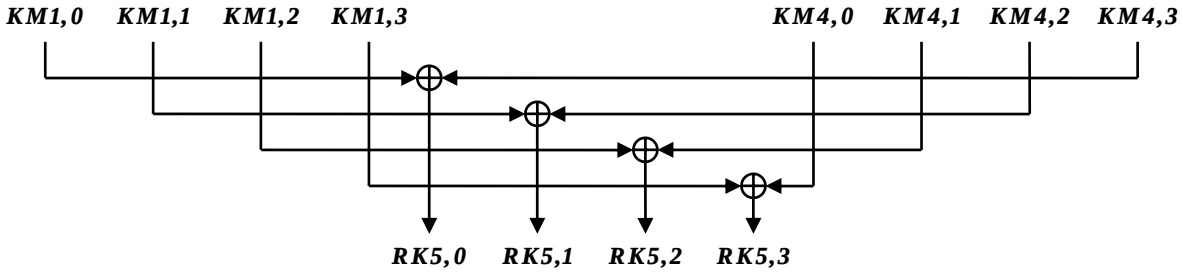
鍵長 192bit および 256 bit の場合、**RK1**～**RK9**、**IK1**～**IK6** は、下記の図の様に求める。

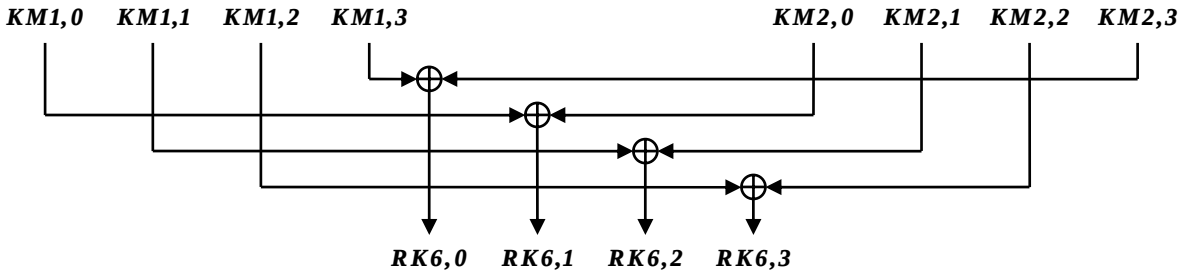


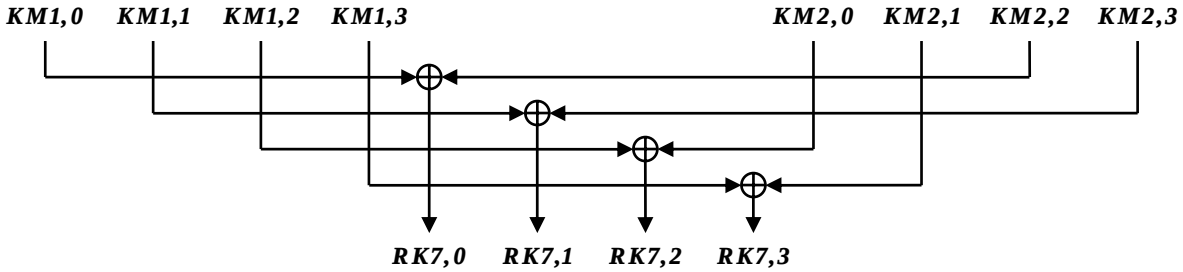


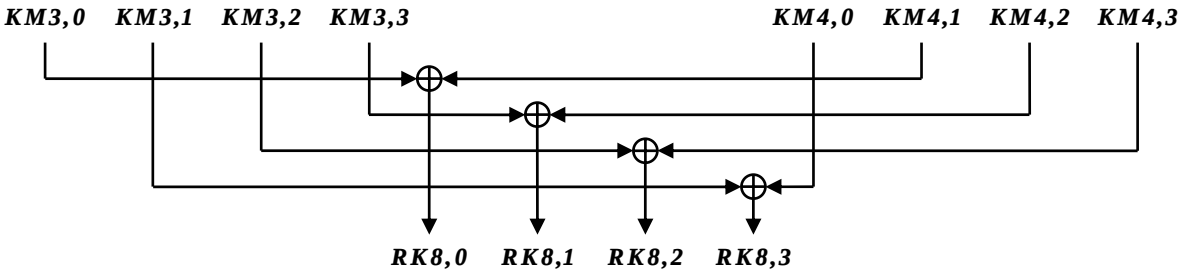

KM3,0 KM3,1 KM3,2 KM3,3 KM4,0 KM4,1 KM4,2 KM4,3
RK3,0 RK3,1 RK3,2 RK3,3

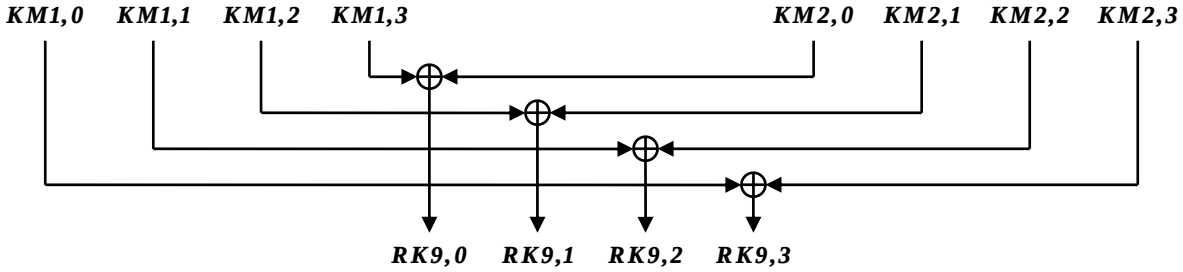

KM3,0 KM3,1 KM3,2 KM3,3 KM4,0 KM4,1 KM4,2 KM4,3
RK4,0 RK4,1 RK4,2 RK4,3

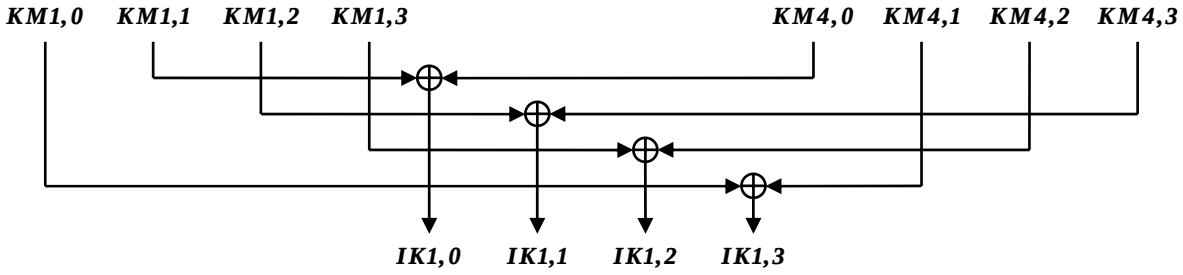

KM1,0 KM1,1 KM1,2 KM1,3 KM4,0 KM4,1 KM4,2 KM4,3
RK5,0 RK5,1 RK5,2 RK5,3

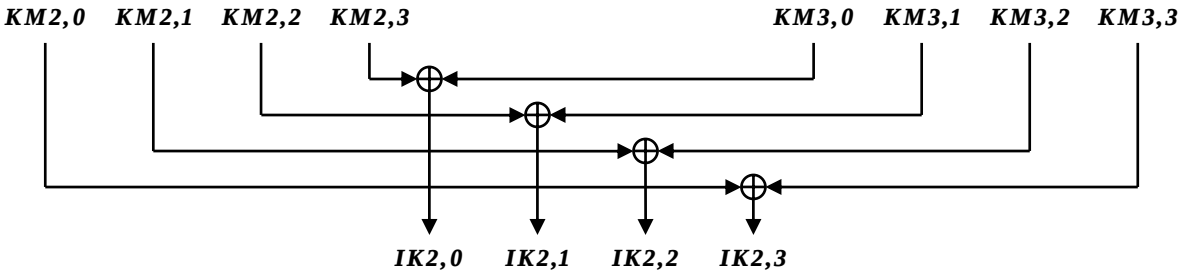

KM1,0 KM1,1 KM1,2 KM1,3 KM2,0 KM2,1 KM2,2 KM2,3
RK6,0 RK6,1 RK6,2 RK6,3

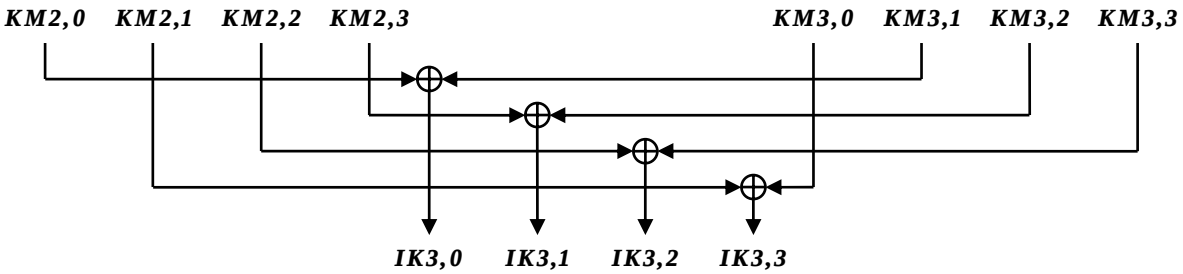

KM1,0 KM1,1 KM1,2 KM1,3 KM2,0 KM2,1 KM2,2 KM2,3
RK7,0 RK7,1 RK7,2 RK7,3

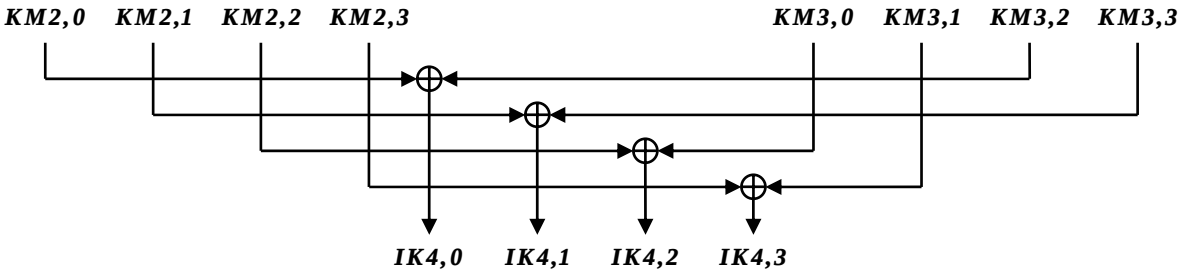

KM3,0 KM3,1 KM3,2 KM3,3 KM4,0 KM4,1 KM4,2 KM4,3
RK8,0 RK8,1 RK8,2 RK8,3

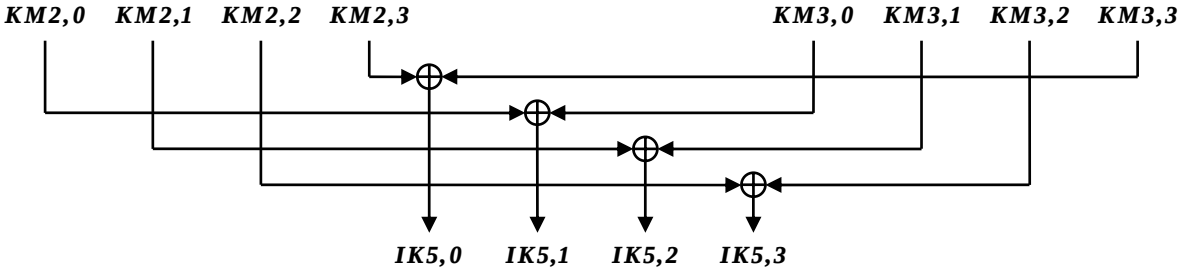

KM1,0 KM1,1 KM1,2 KM1,3 KM2,0 KM2,1 KM2,2 KM2,3
RK9,0 RK9,1 RK9,2 RK9,3

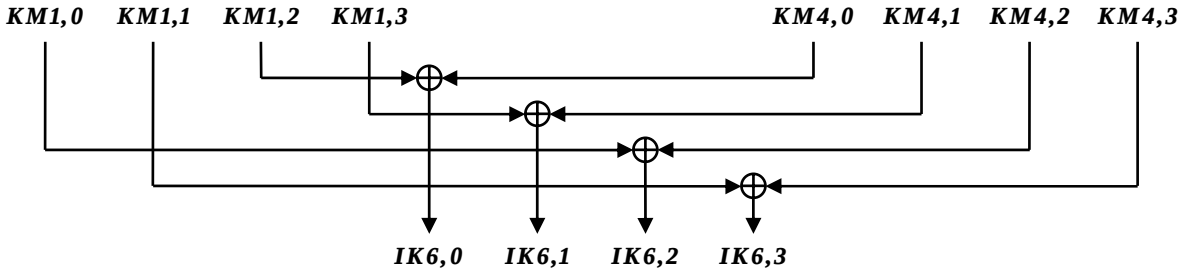

KM1,0 KM1,1 KM1,2 KM1,3 KM4,0 KM4,1 KM4,2 KM4,3
IK1,0 IK1,1 IK1,2 IK1,3


KM2,0 KM2,1 KM2,2 KM2,3 KM3,0 KM3,1 KM3,2 KM3,3
IK2,0 IK2,1 IK2,2 IK2,3


KM2,0 KM2,1 KM2,2 KM2,3 KM3,0 KM3,1 KM3,2 KM3,3
IK3,0 IK3,1 IK3,2 IK3,3







１０．実装方法

・Ｃ言語によるアプリケーション

・ＦＰＧＡに実装

１１．推奨用途

・ファイル暗号および通信文暗号、他

・ＦＰＧＡを実装したハードウェアによる耐改ざん装置