暗号技術仕様書:Hierocrypt–L1

株式会社 東芝

May 2002

目 次

1 設計方針

Hierocrypt–L1 はブロック長 64 ビット、鍵長 128 ビットの暗号として設計された。
Hierocrypt–L1 の設計に当たり、我々は次の点を重視した。

- 主要な攻撃法に対する十分な安全性
- スマートカードやミドルウェアでの暗号化の高速性
- 実装の効率性
- 設計の透明性

以上の条件を満足するために、データ攪拌部の設計には SPN 構造の一種である入れ子型 SPN 構造 (後述) を採用した。また、鍵スケジュール部の設計には Feistel 構造を利用した。
本節では、これらの構造と選択の理由についての概略を述べ、次節で設計基準について述べる。

1.1 データ攪拌部

データ攪拌部の設計においては、Feistel 構造と並んで有力なブロック暗号の代表的な基本構造である SPN 構造を採用した。
SPN 構造の長所には以下のものがある。

- 符号理論に基づいて、差分／線形解読法に対して安全な暗号を設計する手法が確立している
- Feistel 型と異なり、素通しのデータ経路がない
- Feistel 型と比較して、自明な弱鍵が存在しにくい
- ハード実装で高速

一方、短所としては以下の通りである。

- 一般に暗号化と復号で設計が異なるので、実装サイズが大きくなる
- 拡散層の幅が Feistel 型の 2 倍であり、計算コストが増加しやすい

上記の比較から、データ攪拌部での SPN 構造の選択は、安全性をより重視したものであることが分かる。また、暗号化と復号でほとんど設計が変わらないという長所も考慮し、鍵スケジュール部では Feistel 型を採用している。

1.1.1 入れ子型 SPN 構造

SPN 構造の長所は、符号理論を利用した高い安全性を保証する設計手法である。そこでは、活性 S-box 数を効率的に保証するため、拡散層を最大距離分離符号 (MDS 符号) で設計した MDS 行列を利用する。S-box サイズが一定のとき、拡散層を MDS 行列とすると計算コストはおおよそブロック・サイズの自乗に比例する。ブロック長が 64 ビットで S-box サイズが 8 ビットとすると、1 段に含まれる S-box は 8 個となり、64 ビット暗号としては計算コストは高めとなる。
計算コストを抑えるため 128 ビット暗号の SQUARE 暗号や Rijndael 暗号では、16 個の S-box を 4 行 4 列の行列状に並べ、同一行内での MDS 行列演算 (拡散層) とバイトの並べ替えを組み合わせる方法を取っている [3, 4]。この手法に従うと、拡散層の計算コストは下がるが、SQUARE 攻撃と呼ばれる両暗号に特化した有効な攻撃法が提案されている。
我々は、計算コストを抑えながら、SQUARE 攻撃により強い暗号の設計を目指し、入れ子型 SPN 構造を提案した [5, 6]。入れ子型 SPN 構造とは、上位の SPN 構造の S-box の位置に下位の SPN

構造を埋め込んだ、再帰的階層構造である。SPN 構造自体が安全性の高い計算モジュールなので、入れ子型 SPN 構造では安全性が再帰的に保証されるのが特徴である。計算コストに関しては、下位の拡散層では拡散範囲が局所化され、上位拡散層では MDS 符号での語長が長く、語数が減るので、オリジナルの SPN 構造より有利になっている。

我々は、一般の入れ子型 SPN 構造暗号に対し、以下の条件を満たすよう設計することを提案した。

- 各レベルで、SPN 構造の最終段は S-box 層で終了し、拡散層を伴わない (例: SPSP...PS)
- 各レベルで、拡散層は MDS 行列である (語長は各レベルの S-box サイズ)
- 最上位レベルを除き、SPN の段数は偶数段とする (段数は S-box 層の個数)
- 鍵加算は、最下位レベルの全ての S-box と出力 (暗号文) の直前に置く

これらの条件を満たすことによって、活性 S-box 数が階層的に保証され、差分／線形解読法等に対する高い安全性が効率的に保証できる。

S-box 層で SPN 構造を終えるのは、通常の SPN 型暗号で採用される構造で、線形変換である拡散層を追加したとしても安全性がほとんど向上しないためと考えられるためである。

拡散層を MDS 行列とするのは、活性 S-box 数の下限値を最大にし、差分／線形解読に対する高い安全性を実現するためである。

鍵加算を最下位の全 S-box の直前に置くのは、最下位 S-box が唯一の非線形性を持つ要素であり、その効果を最大限に生かすためである。

次に、64 ビット・ブロック暗号 Hierocrypt–L1 でのスペックを示す。

- S-box のサイズは 8 ビット
- 2 階層の入れ子構造 (上位レベルと下位レベル)
- 下位レベルは 2 段 SPN 構造
- 上位レベルの拡散層の結合範囲は S-box 2 個分 (64 ビット=32 ビット ×2)
- 下位レベルの拡散層の結合範囲は S-box 4 個分 (32 ビット=8 ビット ×4)

S-box サイズが 8 ビットとするのは、表引きでの実装が現実的な上限だからである。一般に、サイズが大きいほど、S-box の達成可能な安全性は高まる。

2 階層としたのは、3 階層以上では計算コストが増加するからである。

下位レベルを 2 段 SPN 構造 (SPS) としたのは、活性 S-box 数が効率的に保証できる最小の段数だからである。

上位拡散層の結合範囲を S-box 2 個分、下位拡散層の結合範囲を S-box 4 個分としたのは、本提案と同時提案の 128 ビット・ブロック暗号 Hierocrypt–3 と下位拡散層が共有でき、比較的効率の良い実装が可能だからである。

1.1.2 入れ子型 SPN 構造のまとめ

入れ子型 SPN 構造は非常に簡潔であり、構造自体の透明性は高い。また、暗号のブロック・サイズの変更にも、下位の拡散層のサイズを変えることによって柔軟に対応できる。

また、構成要素である (最下位)S-box と各レベルの拡散層に対して満たすべき条件はあるが、各々をある程度独立に設計することができる。そのため、一部の要素に問題が発見されても、その部分だけ設計しなおすことが可能である。

1.2 鍵スケジュール部

Hierocrypt–L1 の鍵スケジュール部は、繰り返し段構造からなる中間鍵生成部と、各段の中間鍵から拡大鍵を生成する拡大鍵生成部からなる。鍵スケジュール部の設計で心がけたのは、安全面に関しては、拡大鍵間の単純な依存関係によって、鍵の全数探索による探索範囲が実質的に狭くなることがないようにすることである。また、実装面では、1 段当たりの計算時間がデータ攪拌部より短く、on-the-fly で復号のときも鍵生成の初期遅延が十分小さくなるように留意した。

Hierocrypt–L1 では鍵サイズは 128 ビットなので、その自由度を落とさないため、中間鍵は 128 ビットで段関数は全単射とした。また、拡大鍵生成部はデータ攪拌部 1 段当たり 128 ビットの拡大鍵を供給する。

中間鍵の繰り返し段構造のブロック幅 128 ビットは、SPN 構造で構成すると暗号化より大幅に実行速度が落ちる。ローエンド環境で必要な on-the-fly では、鍵スケジュール部はデータ攪拌部より高速であることが望ましいので、以下の構成を採用した。

128 ビットを 64 ビット 2 組に分け、一方を繰り返し線形変換型、他方を Feistel 型とし、前者が後者の鍵に相当するデータを供給する。この構造で線形変換を全単射にすれば、128 ビット幅の全単射の繰り返し段構造が構成できる。また、Feistel 構造の F 関数の拡散層は 32 ビット幅になり、十分高速な動作が実現できる。

Hierocrypt–L1 の中間鍵生成部の最大の特徴は、中心付近の段で変換を逆転し、中間状態値を逆順に再現する折り返し型としたことである。ここで便宜上、対応するデータ攪拌部の位置を元に、折り返し前を平文側、折り返し後を暗号文側と呼ぶことにする。この折り返しがあるので、最終段の中間鍵状態に、初期設定の状態から 1,2 回の変換で到達する。このため、on-the-fly の復号時でも、すぐに拡大鍵をデータ攪拌部に供給でき、初期遅延は小さくなる。

中間鍵での折り返し構造は、平文側と暗号文側の対応する段同士で中間鍵が一致することを意味する。そのため、異なる段で同じ拡大鍵ができないように、中間鍵から拡大鍵を生成する規則は平文側と暗号文側で変える必要がある。

中間鍵から拡大鍵の生成は、中間鍵およびにそれが作られる途中の中間状態のデータを 32 ビット単位に分け、それらの排他的論理和で構成した。この際、平文側と暗号文側で同じ拡大鍵が生成されるのを避けるのはもちろん、単純な依存関係によって弱鍵となることがないよう注意した。

また、中間鍵に周期パターンが生じないよう、繰り返し線形変換部分に段数に依存した定数による排他的論理和を置いた。この定数は、小さな整数の平方根を用いて作成し、設計の透明性を持たせた。

2　設計基準

Hierocrypt の設計に当たり、次の点を重視した。

- 主要な攻撃法に対する十分な安全性
- スマートカードやミドルウェアでの暗号化の高速性
- 実装の効率性
- 設計の透明性

これらの条件の具体的内容について、以下で検討する。

2.1　設計基準の細目

2.1.1　安全性について

最も基本的な安全性は暗号化鍵に対する全数探索であるが、これは暗号化鍵のビット長だけで決まる。暗号化鍵ビット長以外の主要な安全性の評価基準には以下のものがある。

1a) 差分解読法に対する強度

1b) 線形解読法に対する強度

1c) 高階差分解読法に対する強度

1d) 補間攻撃法に対する強度

1e) SQUARE 攻撃に対する強度

1f) truncated 差分解読法に対する強度

1g) impossible 差分攻撃に対する強度

2.1.2　高速性について

暗号化および復号における高速性に関しては、以下の要素が重要である。

2a) データ攪拌部の速度

2b) 鍵設定時間

2c) on-the-fly 鍵スケジュールの速度

2.1.3　実装効率

暗号化および復号における実装効率に関しては、以下の要素が重要である。

3a) ソースコードが短い

3b) RAM が小さい

3c) ROM が小さい

2.2 構成要素の検討

2.2.1 全体構造

Hierocrypt–L1 では入れ子型 SPN 構造を採用している。入れ子型 SPN 構造は、通常の SPN 型アルゴリズムの S-box の部分を下位の SPN 構造で構成したものである。入れ子型 SPN 構造の特徴は次の通りである。

- 単純 SPN 構造では、拡散層を MDS 行列とすると、計算コストは S-box の並列度の自乗に比例して増加する。入れ子型 SPN 構造では、大小 2 種類の拡散層の組合せにより、計算コストの上昇を緩和できる。
- 階層的にブランチ数が保証できるので、活性 S-box 数を効率的に確保できる。
- 単純な階層構造であり、各構成要素がある程度独立に設計できる。

ブロック長 64 ビットの Hierocrypt–L1 の概略は以下の通りである。

- データ攪拌部の基本構造は、64 ビット非線形変換 XS と 64 ビットの上位拡散層 (線形変換)MDS_{H} を交互に並べたもの。最後は XS で終了
- 段数は XS の個数
- XS は、32 ビット非線形変換 xs の 2 並列で構成
- xs は、4 並列の 8 ビット (下位)S-box の 2 層を 32 ビットの下位拡散層で挟んだ構造。
- 拡大鍵の挿入個所は、S-box の直前と出力の直前。

この構造の構成要素は、以下の通り。

- S-box
- 下位拡散層 (MDS_{L})
- 上位拡散層 (MDS_{H})
- 鍵スケジュール

これらの要素を設計する際、安全性／速度／実装効率に関するどのような基準を考慮したかを以下に述べる。

2.2.2 S-box

S-box の評価指標として以下が重要である。

i) 最大差分確率と最大線形確率

ii) 代数次数

iii) 多項式表現での項数

iv) 単純な代数構造の非存在

これらの指標と設計方針の細目との関係を表 1 に示す。

表 1: S-box の評価指標と設計方針の依存関係

指標	設計方針
i	1a,1b
ii	1c,1d
iii	1c,1d
iv	1c,1d

2.2.3 下位拡散層 MDS_{L}

MDS_{L} の評価指標として以下が重要である。

i) 最大距離分離 (MDS) 行列である

ii) S-box と組み合わせたときの、多項式表現 (ビット単位) での項数

iii) 巡回型である

これらの指標と設計方針の細目との関係を表 2 に示す。

表 2: MDS_{L} の評価指標と設計方針の依存関係

指標	設計方針
i	1c,1d
ii	1c,1d
iii	3c

2.2.4 MDS_{H}

MDS_{H} の評価指標として以下が重要である。

i) 最大距離分離 (MDS) 行列であること

ii) バイト単位で見た複数経路性

iii) バイト間結合数ができるだけ少ない

iv) 巡回型である

これらの指標と設計方針の細目との関係を表 3 に示す。

2.3 構成要素の設計

2.3.1 S-box

S-box は、Hierocrypt–L1 の非線形性を持つ唯一の構成要素である。S-box の設計に当たって、我々は以下の点を最も重要視した。

表 3: MDS_{H} の評価指標と設計方針の依存関係

指標	設計方針
i	1c,1d,1e,1f
ii	1c,1d,1e,1f

i) 最大差分確率と最大線形確率

ii) 代数次数

iii) 多項式表現での項数

iv) 単純な代数構造の非存在

最大差分確率 dp^s および最大線形確率 lp^s は次式で定義する。

$$dp^s \equiv \max_{\Delta x \neq 0, \Delta y} \frac{\#\left\{x | f(x) \oplus f(x \oplus \Delta x) = \Delta y\right\}}{2^n}.$$

$$lp^f \equiv \max_{\Gamma x, \Gamma y \neq 0} \left| 2 \cdot \frac{\#\left\{x | x \cdot \Gamma x = f(x) \cdot \Gamma y\right\}}{2^n} - 1 \right|^2.$$

線形確率を上記の正規化された定義にすると、差分確率と線形確率の扱いがほとんど同様に行なえるという利点がある。

入出力ビット長が等しく偶数 N の場合、最大差分／線形確率の最小値は 2^{-N+2} であり、ガロア体 $\mathrm{GF}(2^N)$ 上のべき乗演算で作れることが知られている。$N = 8$ の場合は、両確率の最小値はともに 2^{-6} になる。

ただし、$\mathrm{GF}(2^8)$ 上のべき乗関数そのままでは、高階差分攻撃法や補間攻撃法などの代数的攻撃が適用しやすいので、何らかの変換を組み合わせる必要がある。ただし、組合せの変換自体は最大差分／線形確率を不変に保つ性質のものでなくてはならない。そのような変換として、入力側でのビット置換と、出力側での定数加算 (排他的論理和) を選んだ。出力側ではより一般的なアフィン変換も考えられたが、拡散層 mds_{L} と接続するとき定数倍の演算が行なわれるので、S-box 単独での乗数の決定は行なわないことにした。ここでの定数倍は、mds_{L} の設計の項で後述する。

$$s(x_{(8)}) = Add(Power(Perm(x_{(8)}))) \ .$$

ここで、

$$\begin{aligned}
&y_{(8)} = Perm(x_{(8)}) \ , \\
&y_{i(1)} = x_{\pi[i](1)} \ , \\
&Power \ : \mathrm{GF}(2^8) \ \rightarrow \ \mathrm{GF}(2^8) \ , \\
&Power\left(x_{(8)}\right) = {x_{(8)}}^{247} \ , \\
&Add\left(x_{(8)}\right) = x_{(8)} \oplus \texttt{0x07} \ .
\end{aligned}$$

べき乗の指数を 247 に選んだのは、これが最大差分／線形確率が 2^{-6} になるものの中で、位数が最大であるからである。なお、ここで原始多項式には、$z^8 + z^6 + z^5 + z + 1$ を用いた。

i	1	2	3	4	5	6	7	8
$\pi[i]$	3	7	5	8	6	2	4	1

定数加算値 0x07 は、入力と出力のハミング重みの組合せの分布が、ランダム関数の分布に最も近くなるように選んだ (相関計数は 0.09375)。ここで、8-bit 変数 $x_{(8)}$ のハミング重みは、次式で定義する。

$$\sum_{\substack{i=1,\\ x_{i(8)}=1}}^{8} 1\ .$$

入力ビットの並べ替えは最大差分／線形確率を変えず、かつハミング重みも変えない。上述の並べ替えは、主として代数構造を壊すために導入した。選択基準としては、x^{247} とそのビットを並べ替えたものを多項式表現で表わし、異なる項の個数が最多になるものとした。

2.3.2 mds_{L}

mds_{L} の設計基準として、以下を重視した。

i) 最大距離分離 (MDS) 行列である

ii) S-box と組み合わせたときの、多項式表現 (ビット単位) での項数が最大

iii) 巡回型である。

i) は xs が活性の場合、接続する 8 個の S-box のうち 5 個以上が活性になることを保証するための条件で、GF(2^8) 上の四則演算を利用して構成する。

MDS 行列であるか否かは次の定理で判定できる。

定理 GF(2^N) 上の正方行列が MDS 行列であるための必要十分条件は、任意の小行列が正則であることである。¶

ii) の条件は、補間攻撃や高階差分攻撃に対する耐性を高めるものである。iii) の条件は、効率的な実装を行うための工夫である。

上の条件を満足するような行列として、次式のような mds_{L} を選択した。

$$\begin{pmatrix} \texttt{C4} & \texttt{65} & \texttt{C8} & \texttt{8B} \\ \texttt{8B} & \texttt{C4} & \texttt{65} & \texttt{C8} \\ \texttt{C8} & \texttt{8B} & \texttt{C4} & \texttt{65} \\ \texttt{65} & \texttt{C8} & \texttt{8B} & \texttt{C4} \end{pmatrix}$$

ここで、行列要素は GF(2^8) の元を 16 進表現したものである (後述)。

また、逆行列は以下のとおりである。

$$\begin{pmatrix} \texttt{82} & \texttt{C4} & \texttt{34} & \texttt{F6} \\ \texttt{F6} & \texttt{82} & \texttt{C4} & \texttt{34} \\ \texttt{34} & \texttt{F6} & \texttt{82} & \texttt{C4} \\ \texttt{C4} & \texttt{34} & \texttt{F6} & \texttt{82} \end{pmatrix}$$

2.3.3 MDS_{H}

MDS_{H} の設計基準として、以下を重視した。

i) 最大距離分離 (MDS) 行列である

ii) バイト単位で見た複数経路性

iii) バイト間結合数ができるだけ少ない

iv) 巡回型である

MDS_{H} に対する条件 i) は、mds_{L} が MDS であるというのと若干異なる。mds_{L} では、入出力が 8 ビットの S-box に対するブランチ数が 5 という条件だったが、MDS_{H} では、入出力が 32 ビットの xs に対するブランチ数が 3 という条件に変わる。MDS_{H} に対する最も自然な実現方法は、ガロア体 GF(2^{32}) の元を要素に持つ 2 行 2 列の行列で構成する方法である。

しかしながら、GF(2^{32}) 上の演算は計算コストが掛かるので、別の実装法を検討した。そこで、MDS 行列の構成法に関する次の補題を利用することにした。

補題 $MDS(m,n)$ を m 並列の n ビット語に対する MDS 写像 (行列) とする。これは、語長 n ビットの $(2m,m,m+1)$ 符号に対応する。このとき、任意の正の整数を n' とすると、$MDS(m,n)$ を n' 個連接することによって、語長 $n'n$ ビットの $(2m,m,m+1)$ 符号に対する変換 $MDS(m,n'n)$ が構成できる。

以下にこの補題が正しいことを、実際に $MDS(m,n'n)$ を構成することで示す。
n' 個の写像 $MDS(m,n)$ を考える。

$$\begin{aligned} MDS(m,n)_j \ :\ & x_{n'\cdot 1+j(n)} \| x_{n'\cdot 2+j(n)} \| \cdots \| x_{n'\cdot m+j(n)} \\ & \mapsto y_{n'\cdot 1+j(n)} \| y_{n'\cdot 2+j(n)} \| \cdots \| y_{n'\cdot m+j(n)} \quad , \quad 1 \le j \le n' \ , \end{aligned}$$

ここで、変数 $x_{k(n)}$,$y_{k(n)}$ に対する以下の連接を考える。

$$\begin{aligned} X_{i(n'n)} &= x_{n'\cdot i+1(n)} \| x_{n'\cdot i+2(n)} \| \cdots \| x_{n'\cdot i+n'(n)} \quad , \\ Y_{i(n'n)} &= y_{n'\cdot i+1(n)} \| y_{n'\cdot i+2(n)} \| \cdots \| y_{n'\cdot i+n'(n)} \quad , \quad 1 \le i \le m \ . \end{aligned}$$

ここで、次の写像 $MDS(m,n'n)$ を考える。

$$\begin{aligned} MDS(m,n'n) \ :\ & X_{1(n'n)} \| X_{2(n'n)} \| \cdots \| X_{m(n'n)} \\ & \mapsto Y_{1(n'n)} \| Y_{2(n'n)} \| \cdots \| Y_{m(n'n)} \quad . \end{aligned}$$

写像 $MDS(m,n'n)$ に対する入力差分が非零であるとき、少なくともその部分写像の 1 個 $MDS(m,n)_j$ は活性である。よって、入出力の n ビット語 $x_{n'i+j(n)}$, $y_{n'i'+j(n)}$ のうち $m+1$ 個は活性である。各 n ビット語は各々別の $n'n$ ビット語 $X_{i(n'n)}$,$Y_{i'(n'n)}$ に属するから、最終的に、$MDS(m,n'n)$ も MDS 写像であることが分かる。¶

GF(2^4) 上の 2 行 2 列の MDS 行列が構成できるので、上記の補題を用いると GF(2^4) の演算だけで 32 ビット語に対する MDS が構成できる。つまり、上記の補題で $n=4$, $n'=8$, $m=2$ の例が構成できる。具体的には、S-box の同じ位置のビットを集めることで、各 xs から 4 ビットの語が得られ GF(2^4) の元が構成でき、それを結合する 2 行 2 列の $MDS(2,4)$ ができる。これを 8 個

並列にして、連接することで $MDS(2,32)$ が実現できる。ここで、$MDS(2,4)$ として同じものを使うと、同じ S-box に属するビットは並列的に同じ処理を受けることになる。そして、その処理とは他のビットとの排他的論理和だけであるから、$MDS(2,32)$ 写像はバイト単位の線形結合に他ならない。よって、バイト間の結線は $GF(2)$ を要素とする 8 行 8 列の行列で表わせる。

SPN 構造と MDS 行列を組み合わせた暗号の代表的なものに、SQUARE と Rijndael があり、既に SQUARE 攻撃と呼ばれる SPN 構造を利用した専用攻撃法がある。その基本形は、1 バイトは全パターンを巡り他のバイトは固定するという 256 個の平文を用いるものである。そこでは、256 個の平文に対し、中間出力ビットで成り立つバランス式から鍵を推定する。SQUARE 暗号でこの攻撃が可能なのは、ある段の S-box と結線で繋がっている 2 段前の S-box が、各 4 バイト・セットの中に各々1 個ずつしかないという特殊な条件のためである。そこで我々は、ある段の S-box と結線で繋がっている 2 段前の S-box が、各 4 バイト・セットの中に 2 個以上あり、逆行列でも同様の条件が成り立つという条件で、MDS_{H} を選ぶことにした。この条件を複数経路性と呼ぶことにする。4 バイト・セット間の結合は MDS_{H} 行列に対応する GF(2^4) の元で決まる。よって、GF(2^4) の元ごとに複数経路性の可否が決まっている。実際に複数経路性を満たす GF(2^4) の元は、16 進表現で、3, 5, 6, 7, A, B, C, E の 8 個である。

具体的には、行列は以下の手順で選択する。なお、行列に対するその他の条件として、mds_{L} と異なり、巡回性は課さない。その理由は、MDS_{H} は行列要素が 4 個しかないので、全数探索が容易であり、巡回性の条件をはずすことでより実装効率の良いものが選択できるからである。手順は以下の通り。

[SQUARE 攻撃に対して耐性の高い MDS 行列を作る手順]

1. 定数倍の候補である GF(2^4) の元で行列を作る
2. MDS であれば次項へ。否なら前項に戻る
3. 逆行列を求め、行列要素が全部 GF(2^4) の候補に含まれているなら次項へ。否なら先頭に戻る。
4. 最終的に残った行列を候補とする

上記の手順により、行内要素の回転と反転の自由度を除き、4 種類の候補が求まった。この中から、ソフト実装が最も効率良くできるものを選んだ結果、下記の行列が得られた。

$$\begin{pmatrix} 5 & 7 \\ A & B \end{pmatrix}$$

ここで、行列要素は GF(2^4) の元を 16 進表現したものであり、原始多項式は z^4+z+1 である。

復号に用いる行列は、上記行列の逆行列である。

$$\begin{pmatrix} C & A \\ 5 & B \end{pmatrix}$$

2.3.4 $P^{(n)}$

$P^{(n)}$ は鍵スケジュール部の拡散層として利用される。$P^{(n)}$ に対し、以下の設計基準を設けた。

i) 計算コストが低い

ii) スケーラビリティがある

iii) 拡散性が高い

iv) 全単射である

v) 最大距離分離 (MDS) 行列でなくても良い。

条件 i) は、鍵スケジュールの計算時間がデータ攪拌部より短くするために必要である。
条件 ii) は、ブロック・サイズの変更に柔軟に対応するのに有効である。
条件 iii) は、拡大鍵から暗号化鍵の推定を困難にするために必要である。
条件 iv) は、暗号化鍵の情報量が鍵スケジュールの過程で減衰することを防ぐために必要である。
条件 v) は、高速性と安全性に対するバランスを考慮して提案した。安全性に関しては望ましくない条件だが、条件 iv) があるので問題はないと考えられる。
上記の条件を満たすものとして、$(4n)$ ビット・データに対する次式の線形変換を選択した。

$$Y_{(4n)} = P^{(n)}\left(X_{(4n)}\right) \ .$$

$$\begin{pmatrix} y_{1(n)} \\ y_{2(n)} \\ y_{3(n)} \\ y_{4(n)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_{1(n)} \\ x_{2(n)} \\ x_{3(n)} \\ x_{4(n)} \end{pmatrix} \ .$$

この線形変換は、次式に示すように 2 種類の involution 型線形変換の合成関数になっている。

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \ .$$

3　暗号アルゴリズム

以下では、64 ビット・ブロック暗号 Hierocrypt–L1 の仕様を記述する。Hierocrypt–L1 の暗号化鍵の鍵長は 128 ビットである。

3.1　表記法

n ビットデータは、下付き添え字 (n) を付けて表わし、n が 16 以上のときは変数名を大文字、16 未満のときは小文字で表わす。その構成ビットには 1 から n の番号を付け、big endian を用いる。例えば、$X_{(n)}$ は n ビットデータであり、次式のようにビットの連接として書いたとき、MSB は $x_{n(1)}$ である。

$$X_{(n)} = x_{1(1)} \| x_{2(1)} \| \cdots \| x_{n(1)} \ .$$

1 ビットより大きいサブデータに分割する場合は、MSB を含むサブデータの番号を 1 とする。

$$X_{(mn)} = x_{1(n)} \| x_{2(n)} \| \cdots \| x_{m(n)} \ .$$

すなわち、サブデータ $x_{i(n)}$ の LSB は元のデータの $i \cdot n$ 番目ビット $x_{in(1)}$ と一致する。

64 ビット・データ $X_{(64)}$ を 2 つの 32 ビット・データ $X_{(64)} = X_{1(32)} \| X_{2(32)}$ に分割して表現する例を示す。

$$X_{(64)} = X_{1(32)} \| X_{2(32)} \ ,$$

さらに各 32 ビット・データを 4 つの 8 ビット・データに分割して表現する例を示す。

$$X_{i(32)} = x_{4i-4+1(8)} \| x_{4i-4+2(8)} \| \cdots \| x_{4i(8)} \ , \quad i = 1, 2, 3, 4 \ ,$$

さらに各 8 ビット・データを 8 個の 1 ビット・データに分割して表現する例を示す。

$$x_{j(8)} = x_{8j-8+1(1)} \| x_{8j-8+2(1)} \| \cdots \| x_{8j(1)} \ , \quad j = 1, 2, \ldots, 8.$$

3.2　構造

本節では、データ攪拌部と鍵スケジュール部の動作を述べる。そこで利用される基本演算については、次節にまとめて記述する。

3.2.1　暗号化

Hierocrypt–L1 の暗号化は、5 回の段関数 (ρ 関数)、XS 関数 (XS)、最終鍵加算 (AK) で構成される (図 2)。XS 関数も 1 段と数え、段数は 6 段である。

また、$X^{(0)}_{(64)} \equiv P_{(64)}$ は平文とする。

データ $X^{(t)}_{(64)}$ は、データ $X^{(t-1)}_{(64)}$ とラウンド鍵 $K^{(t)}_{(128)}$ を入力とする t 回目の ρ の出力である。

$$X^{(t)}_{(64)} = \rho\left(X^{(t-1)}_{(64)}, K^{(t)}_{(128)}\right), \quad t = 1, 2, \cdots, 5 \ .$$

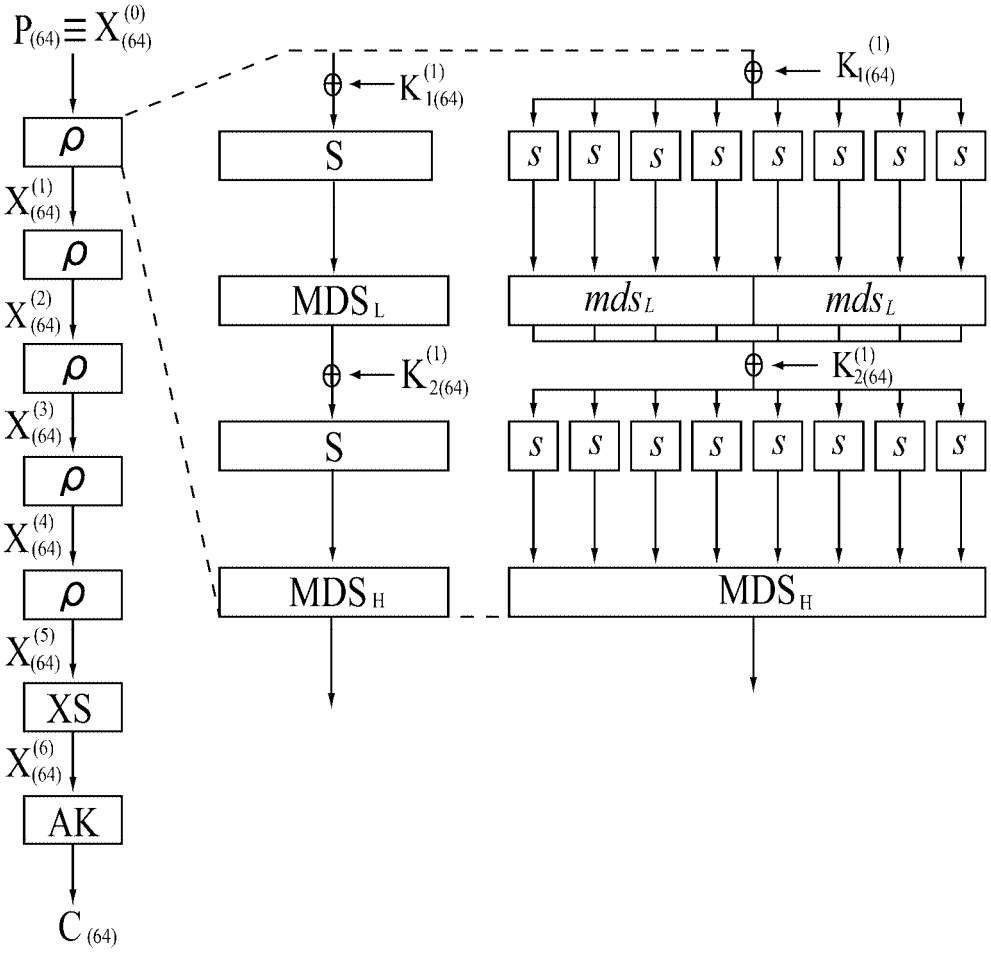


図 1: Hierocrypt–L1 の構造

データ $X_{(64)}^{(6)}$ は、$X_{(64)}^{(5)}$ とラウンド鍵 $K_{(128)}^{(6)}$ を入力とする XS 関数の出力である。

$$X_{(64)}^{(6)} = XS\left(X_{(64)}^{(5)}, K_{(128)}^{(6)}\right) .$$

暗号文 $C_{(64)}$ は、$X_{(64)}^{(6)}$ と最終鍵 $\left(K_{1(32)}^{(7)} \| K_{2(32)}^{(7)}\right)$ との加算 (AK) として得られる。

$$C_{(64)} = X_{(64)}^{(6)} \oplus \left(K_{1(32)}^{(7)} \| K_{2(32)}^{(7)}\right) .$$

$$P_{(64)} \equiv X_{(64)}^{(0)} \overset{\rho}{\longmapsto} X_{(64)}^{(1)} \overset{\rho}{\longmapsto} \cdots \overset{\rho}{\longmapsto} X_{(64)}^{(5)} \overset{XS}{\longmapsto} X_{(64)}^{(6)} \overset{AK}{\longmapsto} C_{(64)}$$

図 2: Hierocrypt–L1 の暗号化

3.2.2 復号

本暗号の復号は、暗号化の逆で、最終鍵加算、XS 関数の逆関数 (XS^{-1})、5 回の段関数の逆関数 (ρ^{-1}) で構成される。

$$
\begin{aligned}
&X^{(6)}_{(64)} = C_{(64)} \oplus \left(K^{(7)}_{1(32)} \| K^{(7)}_{2(32)} \right) \ , \\
&X^{(5)}_{(64)} = XS^{-1} \left(X^{(6)}_{(64)}, K^{(6)}_{(128)} \right) \ , \\
&X^{(t-1)}_{(64)} = \rho^{-1} \left(X^{(t)}_{(64)}, K^{(t)}_{(128)} \right) \ , \ t = 5, \ldots, 2, 1.
\end{aligned}
$$

平文 $P_{(64)}$ は最終出力 $X^{(0)}_{(64)}$ として得られる。

$$
P_{(64)} = X^{(0)}_{(64)} \ .
$$

3.2.3 鍵スケジューリング

鍵スケジューリングの主要部は、t 段目の中間鍵出力 $Z^{(t)}_{(128)}$ を順次求める処理と、中間鍵からデータ攪拌部で使用する拡大鍵 $K^{(t)}_{(128)}$ を合成する処理の 2 種類で構成される。また、これらに先立ち、暗号化鍵から中間鍵の初期値を作る前処理を行なう。

$$
\begin{aligned}
K_{(128)} \equiv Z^{(-1)}_{(128)} &\overset{\sigma_0}{\longmapsto} Z^{(0)}_{(128)} \overset{\sigma}{\longmapsto} Z^{(1)}_{(128)} \overset{\sigma}{\longmapsto} Z^{(2)}_{(128)} \overset{\sigma}{\longmapsto} \cdots \overset{\sigma}{\longmapsto} Z^{(4)}_{(128)} \\
&\overset{\sigma^{-1}}{\longmapsto} Z^{(5)}_{(128)} \overset{\sigma^{-1}}{\longmapsto} \cdots \overset{\sigma^{-1}}{\longmapsto} Z^{(T+1)}_{(128)}
\end{aligned}
$$

図 3: Hierocrypt–L1 の鍵スケジュール (中間鍵)

中間鍵の更新には、$1 \le t \le 4$ では関数 σ、$5 \le t \le 7$ ではその逆関数 σ^{-1} を利用する。このため、$t = 4$ に対し、中間鍵出力値は対称に並ぶ。すなわち、次式が成立する。

$$
Z^{(t)}_{(128)} = Z^{(8-t)}_{(128)} \ , \ \ 5 \le t \le 7 \ .
$$

t 段目の中間鍵出力 $Z^{(t)}_{(128)}$ と t 段目拡大鍵 $K^{(t)}_{(128)}$ $(1 \le t \le 7)$ は、次式のように、4 個の 32 ビット・サブブロックに分割される。

$$
\begin{aligned}
&Z^{(t)}_{(128)} = Z^{(t)}_{1(32)} \| Z^{(t)}_{2(32)} \| Z^{(t)}_{3(32)} \| Z^{(t)}_{4(32)} \ , \\
&K^{(t)}_{(128)} = K^{(t)}_{1(32)} \| K^{(t)}_{2(32)} \| K^{(t)}_{3(32)} \| K^{(t)}_{4(32)} \ .
\end{aligned}
$$

3.2.4 段依存定数

鍵スケジュール部において、周期パターンの出現を抑制し、関連鍵攻撃などに対する耐性を向上するために、各段で段依存性のある定数加算を挿入する。この段依存定数は、平方根を整数で割った無理数を 2 進展開して得られる以下の 5 個の 32 ビット・データを利用する。

$$
\begin{aligned}
H_0 &= \texttt{0x5A827999} = trunc\left(\sqrt{2}/4\right), \\
H_1 &= \texttt{0x6ED9EBA1} = trunc\left(\sqrt{3}/4\right), \\
H_2 &= \texttt{0x8F1BBCDC} = trunc\left(\sqrt{5}/4\right), \\
H_3 &= \texttt{0xCA62C1D6} = trunc\left(\sqrt{10}/4\right), \\
H_4 &= \texttt{0xF7DEF58A} = trunc\left(\sqrt{15}/4\right),
\end{aligned}
$$

ただし、$trunc(x) = \lfloor 2^{32} x \rfloor$ である。

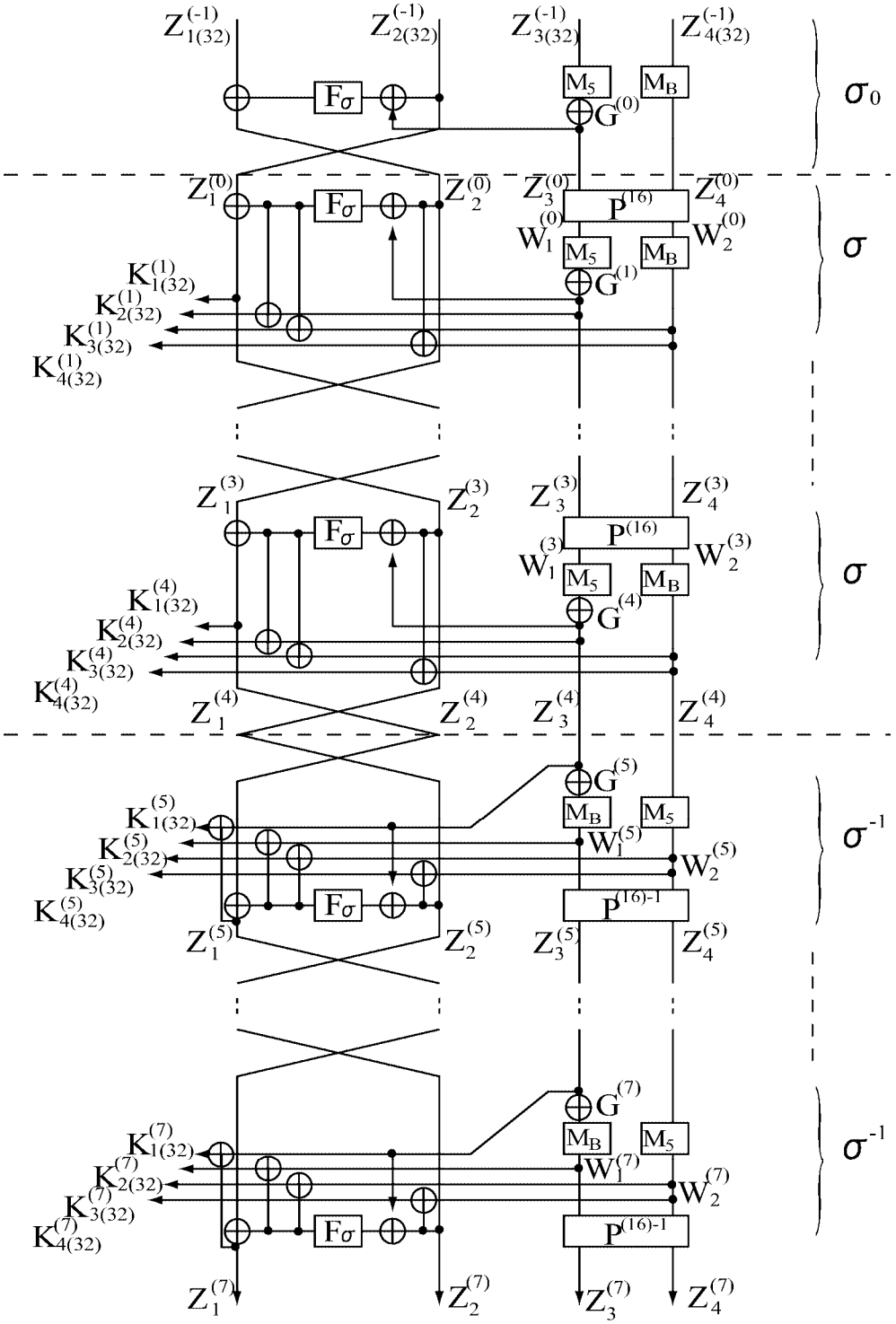

$Z^{(-1)}_{1(32)}$
$Z^{(-1)}_{2(32)}$
$Z^{(-1)}_{3(32)}$
$Z^{(-1)}_{4(32)}$
F_σ
M_5
M_B
$G^{(0)}$
σ_0
$Z^{(0)}_1$
$Z^{(0)}_2$
$Z^{(0)}_3$
$Z^{(0)}_4$
$P^{(16)}$
$W^{(0)}_1$
$W^{(0)}_2$
$G^{(1)}$
σ
$K^{(1)}_{1(32)}$
$K^{(1)}_{2(32)}$
$K^{(1)}_{3(32)}$
$K^{(1)}_{4(32)}$
$Z^{(3)}_1$
$Z^{(3)}_2$
$Z^{(3)}_3$
$Z^{(3)}_4$
$W^{(3)}_1$
$W^{(3)}_2$
$G^{(4)}$
$K^{(4)}_{1(32)}$
$K^{(4)}_{2(32)}$
$K^{(4)}_{3(32)}$
$K^{(4)}_{4(32)}$
$Z^{(4)}_1$
$Z^{(4)}_2$
$Z^{(4)}_3$
$Z^{(4)}_4$
$G^{(5)}$
$W^{(5)}_1$
$W^{(5)}_2$
σ^{-1}
$K^{(5)}_{1(32)}$
$K^{(5)}_{2(32)}$
$K^{(5)}_{3(32)}$
$K^{(5)}_{4(32)}$
$P^{(16)-1}$
$Z^{(5)}_1$
$Z^{(5)}_2$
$Z^{(5)}_3$
$Z^{(5)}_4$
$G^{(7)}$
$W^{(7)}_1$
$W^{(7)}_2$
$K^{(7)}_{1(32)}$
$K^{(7)}_{2(32)}$
$K^{(7)}_{3(32)}$
$K^{(7)}_{4(32)}$
$Z^{(7)}_1$
$Z^{(7)}_2$
$Z^{(7)}_3$
$Z^{(7)}_4$

図 4: 鍵スケジュール

3.2.5 前処理: 初期設定とダミー段

暗号化鍵 $K_{(128)}$ をそのまま中間鍵 $Z^{(-1)}_{(128)}$ とする。これにダミー処理 σ_0 を行なって中間鍵 $Z^{(0)}_{(128)}$ を作る。

[中間鍵の初期化]

$$Z^{(-1)}_{(128)} = K_{(128)} \ .$$

[ダミー段処理] (σ_0 関数) 繰り返し段構造の前に、以下の 1 段分のダミー処理を行なう。これは後で示す鍵段関数 σ から拡散層 $P^{(16)}$ を取り除いたものである。

入出力仕様 $Z^{(0)}_{(128)} = \sigma_0\left(Z^{(-1)}_{(128)}, G^{(0)}_{(32)}\right)$

関数定義

$$\begin{aligned}
&Z^{(0)}_{3(32)} = M_5\left(Z^{(-1)}_{3(32)}\right) \oplus G^{(0)}_{(32)} \ , \\
&Z^{(0)}_{4(32)} = M_B\left(Z^{(-1)}_{4(32)}\right) \ , \\
&Z^{(0)}_{1(32)} = Z^{(-1)}_{2(32)} \ , \\
&Z^{(0)}_{2(32)} = Z^{(-1)}_{1(32)} \oplus F_\sigma\left(Z^{(-1)}_{2(32)} \oplus Z^{(0)}_{3(32)}\right) \ .
\end{aligned}$$

なお、段依存定数 $G^{(0)}_{(32)}$ には次の値を用いる。

$$G^{(0)}_{(32)} = H_0 \ .$$

3.2.6 中間鍵の段関数 (σ 関数)

中間鍵出力 $Z^{(t)}_{(128)}$ は、$t = 4$ に至るまでは順方向の変換 σ を行なうが、それ以降は逆の変換 σ^{-1} を行なう。この折り返し型の変換では、$t = 4$ を境に値が対称的になる。

$$Z^{(t)}_{(128)} = Z^{(8-t)}_{(128)} \ , \quad 5 \le t \le 7 \ .$$

対応するデータ攪拌部の位置から、順方向の区間 ($1 \le t \le 4$) を平文側、逆方向の区間 ($5 \le t \le 7$) を暗号文側と呼ぶことにする。

[中間鍵の更新 (平文側)] $(1 \le t \le 4)$

入出力仕様 $Z^{(t)}_{(128)} = \sigma\left(Z^{(t-1)}_{(128)} \ , \ G^{(t)}_{(32)}\right)$

関数定義

$$\begin{aligned}
&W^{(t-1)}_{1(32)} \| W^{(t-1)}_{2(32)} = P^{(16)}\left(Z^{(t-1)}_{3(32)} \| Z^{(t-1)}_{4(32)}\right) \ , \\
&Z^{(t)}_{3(32)} = M_5\left(W^{(t-1)}_{1(32)}\right) \oplus G^{(t)}_{(32)} \ , \\
&Z^{(t)}_{4(32)} = M_B\left(W^{(t-1)}_{2(32)}\right) \ , \\
&Z^{(t)}_{1(32)} = Z^{(t-1)}_{2(32)} \ , \\
&Z^{(t)}_{2(32)} = Z^{(t-1)}_{1(32)} \oplus F_\sigma\left(Z^{(t-1)}_{2(32)} \oplus Z^{(t)}_{3(32)}\right) \ .
\end{aligned}$$

表 4: Hierocrypt–L1 の鍵スケジュール

中間鍵段数 t	拡大鍵	関数	定数 $G^{(t)}_{(32)}$
0 (dummy)	—	σ_0	$G^{(0)}_{(32)} = H_0$
1	$K^{(1)}_{(128)}$	σ	$G^{(1)}_{(32)} = H_1$
2	$K^{(2)}_{(128)}$	σ	$G^{(2)}_{(32)} = H_2$
3	$K^{(3)}_{(128)}$	σ	$G^{(3)}_{(32)} = H_3$
4	$K^{(4)}_{(128)}$	σ	$G^{(4)}_{(32)} = H_4$
5	$K^{(5)}_{(128)}$	σ^{-1}	$G^{(5)}_{(32)} = H_4$
6	$K^{(6)}_{(128)}$	σ^{-1}	$G^{(6)}_{(32)} = H_3$
7	$K^{(7)}_{(128)}$	σ^{-1}	$G^{(7)}_{(32)} = H_2$

[中間鍵の更新 (暗号文側)] $(5 \le t \le 7)$

入出力仕様 $Z^{(t)}_{(128)} = \sigma^{-1}\left(Z^{(t-1)}_{(128)}, G^{(t)}_{(32)}\right)$

関数定義

$$
\begin{aligned}
Z^{(t)}_{1(32)} &= Z^{(t-1)}_{2(32)} \oplus F_\sigma\left(Z^{(t-1)}_{1(32)} \oplus Z^{(t-1)}_{3(32)}\right) , \\
Z^{(t)}_{2(32)} &= Z^{(t-1)}_{1(32)} , \\
W^{(t)}_{1(32)} &= M_B\left(Z^{(t-1)}_{3(32)} \oplus G^{(t)}_{(32)}\right) , \\
W^{(t)}_{2(32)} &= M_5\left(Z^{(t-1)}_{4(32)}\right) , \\
Z^{(t)}_{3(32)} \| Z^{(t)}_{4(32)} &= {P^{(32)}}^{-1}\left(W^{(t)}_{1(32)} \| W^{(t)}_{2(32)}\right) .
\end{aligned}
$$

3.2.7 拡大鍵生成

中間鍵から拡大鍵の生成規則は、中間鍵系列の折り返しの前後で異なるものを用いる。以下、平文側と暗号文側で分けて記述する。

[拡大鍵生成 (平文側)] $(1 \le t \le 4)$

$$
\begin{aligned}
V^{(t)}_{(32)} &= F_\sigma\left(Z^{(t-1)}_{2(32)} \oplus Z^{(t)}_{3(32)}\right) , \\
K^{(t)}_{1(32)} &= Z^{(t-1)}_{1(32)} \oplus V^{(t)}_{(32)} , \\
K^{(t)}_{2(32)} &= Z^{(t)}_{3(32)} \oplus V^{(t)}_{(32)} , \\
K^{(t)}_{3(32)} &= Z^{(t)}_{4(32)} \oplus V^{(t)}_{(32)} , \\
K^{(t)}_{4(32)} &= Z^{(t-1)}_{2(32)} \oplus Z^{(t)}_{4(32)} .
\end{aligned}
$$

[拡大鍵生成 (暗号文側)] $(5 \le t \le 7)$

$$
\begin{aligned}
V^{(t)}_{(32)} &= F_\sigma\left(Z^{(t-1)}_{1(32)} \oplus Z^{(t-1)}_{3(32)}\right) , \\
K^{(t)}_{1(32)} &= Z^{(t)}_{1(32)} \oplus Z^{(t-1)}_{3(32)} , \\
K^{(t)}_{2(32)} &= W^{(t)}_{1(32)} \oplus V^{(t)}_{(32)} , \\
K^{(t)}_{3(32)} &= W^{(t-1)}_{2(32)} \oplus V^{(t)}_{(32)} , \\
K^{(t)}_{4(32)} &= Z^{(t-1)}_{1(32)} \oplus W^{(t)}_{2(32)} .
\end{aligned}
$$

3.3 基本演算

ここでは前節で述べた暗号化／復号アルゴリズム／鍵スケジューリングの部品となる関数の詳細を示す。以下の節で述べる各関数の関係を図 5 にまとめた。

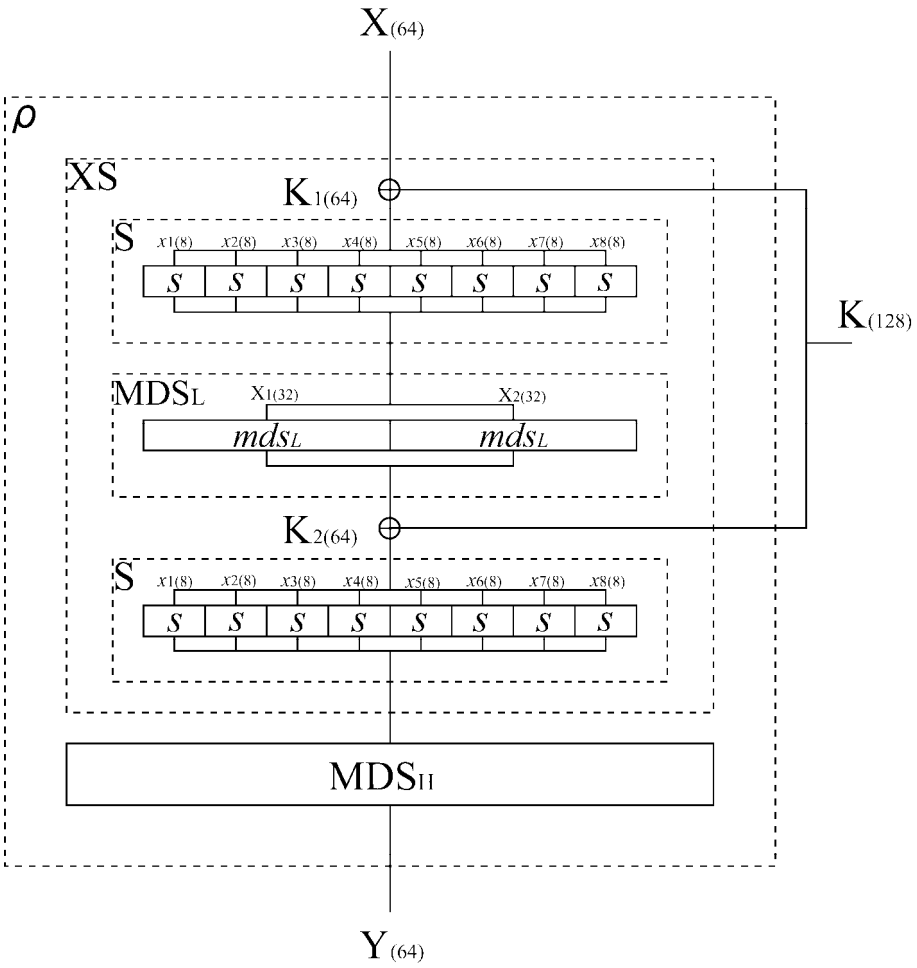


図 5: 段関数の詳細

3.3.1 段関数 ρ

データ攪拌部の段関数 ρ 関数は、XS 関数と MDS_{H} 関数の合成関数である。

入出力仕様 $Y_{(64)} = \rho\left(X_{(64)}, K_{(128)}\right)$

関数定義

$\rho\left(X_{(64)}, K_{(128)}\right) = MDS_{\mathrm{H}}\left(XS\left(X_{(64)}, K_{(128)}\right)\right)$.

復号に用いる ρ^{-1} 関数は、MDS_{H}^{-1} 関数と XS^{-1} 関数の合成関数である。

入出力仕様 $X_{(64)} = \rho^{-1}\left(Y_{(64)}, K_{(128)}\right)$
関数定義
$\rho^{-1}\left(Y_{(64)}, K_{(128)}\right) = XS^{-1}\left(MDS_{\mathrm{H}}^{-1}\left(Y_{(64)}\right), K_{(128)}\right)$.

3.3.2 XS 関数

XS 関数は、S 関数と 64 ビット鍵加算と MDS_{L} 関数の合成関数である。

入出力仕様 $Y_{(64)} = XS\left(X_{(64)}, K_{(128)}\right)$
関数定義
$K_{1(64)} \| K_{2(64)} = K_{(128)}$
$XS\left(X_{(64)}, K_{(128)}\right) = S\left(MDS_{\mathrm{L}}\left(S\left(X_{(64)} \oplus K_{1(64)}\right)\right) \oplus K_{2(64)}\right)$.

復号に用いる XS^{-1} 関数は、S^{-1} 関数と 64 ビット鍵加算と MDS_{L}^{-1} 関数の合成関数である。

入出力仕様 $X_{(64)} = XS^{-1}\left(Y_{(64)}, K_{(128)}\right)$
関数定義
$K_{1(64)} \| K_{2(64)} = K_{(128)}$
$XS^{-1}\left(Y_{(64)}, K_{(128)}\right) = S^{-1}\left(MDS_{\mathrm{L}}^{-1}\left(S^{-1}\left(Y_{(64)}\right) \oplus K_{2(64)}\right)\right) \oplus K_{1(64)}$.

3.3.3 S 関数

S 関数では、64 ビット・データを 8 ビットごとに 8 分割し、各々に同一の s 関数を適用する。

入出力仕様 $Y_{(64)} = S\left(X_{(64)}\right)$
関数定義
$x_{1(8)} \| x_{2(8)} \| \cdots \| x_{8(8)} = X_{(64)}$,
$Y_{(64)} = s(x_{1(8)}) \| s(x_{2(8)}) \| \cdots \| s(x_{8(8)})$.

復号で用いる S^{-1} 関数は以下のとおりである。

入出力仕様 $X_{(64)} = S^{-1}\left(Y_{(64)}\right)$
関数定義
$y_{1(8)} \| y_{2(8)} \| \cdots \| y_{8(8)} = Y_{(64)}$,
$X_{(64)} = s^{-1}(y_{1(8)}) \| s^{-1}(y_{2(8)}) \| \cdots \| s^{-1}(y_{8(8)})$.

3.3.4 s 関数

s 関数は入出力 8 ビットの S-box であり、以下の表で与えられる。数値の表記は 16 進表現である。

入出力仕様 $y_{(8)} = s(x_{(8)})$

関数定義

$(s(\mathtt{00})\ s(\mathtt{01})\ s(\mathtt{02})\ \ldots\ s(\mathtt{0F})\ s(\mathtt{10})\ s(\mathtt{11})\ \ldots\ s(\mathtt{FF})) =$

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

復号で用いる s^{-1} 関数は以下のとおりである。

入出力仕様 $x_{(8)} = s^{-1}(y_{(8)})$

関数定義

$(s^{-1}(\mathtt{00})\ s^{-1}(\mathtt{01})\ s^{-1}(\mathtt{02})\ \ldots\ s^{-1}(\mathtt{0F})\ s^{-1}(\mathtt{10})\ s^{-1}(\mathtt{11})\ \ldots\ s^{-1}(\mathtt{FF})) =$

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

3.3.5 MDS_{L} 関数

MDS_{L} 関数では、64 ビット・データを 2 分割して得られる 32 ビット・サブデータに同一の mds_{L} 関数を適用する。

入出力仕様 $Y_{(64)} = MDS_{\mathrm{L}}\left(X_{(64)}\right)$

関数定義

$X_{1(32)} \| X_{2(32)} = X_{(64)}$,

$Y_{(64)} = mds_{\mathrm{L}}(X_{1(32)}) \| mds_{\mathrm{L}}(X_{2(32)})$.

復号に用いる MDS_{L}^{-1} 関数は以下のとおりである。

入出力仕様 $X_{(64)} = MDS_{\mathrm{L}}^{-1}\left(Y_{(64)}\right)$
関数定義
$Y_{1(32)} \| Y_{2(32)} = Y_{(64)}$,
$X_{(64)} = mds_{\mathrm{L}}^{-1}(Y_{1(32)}) \| mds_{\mathrm{L}}^{-1}(Y_{2(32)})$.

3.3.6 mds_{L} 関数

mds_{L} 関数は GF(2^8) を要素を元とする 4 行 4 列の行列による線形変換である。引数 $X_{(32)}$ を 4 分割した $x_{1(8)}$, $x_{2(8)}$, $x_{3(8)}$, $x_{4(8)}$ も GF(2^8) の元とみなして計算する。

入出力仕様 $Y_{(32)} = mds_{\mathrm{L}}(X_{(32)})$
関数定義
$x_{1(8)} \| x_{2(8)} \| x_{3(8)} \| x_{4(8)} = X_{(32)}$,
$Y_{(32)} = y_{1(8)} \| y_{2(8)} \| y_{3(8)} \| y_{4(8)}$,

$$\begin{pmatrix} y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \end{pmatrix} = \begin{pmatrix} \mathtt{C4} & \mathtt{65} & \mathtt{C8} & \mathtt{8B} \\ \mathtt{8B} & \mathtt{C4} & \mathtt{65} & \mathtt{C8} \\ \mathtt{C8} & \mathtt{8B} & \mathtt{C4} & \mathtt{65} \\ \mathtt{65} & \mathtt{C8} & \mathtt{8B} & \mathtt{C4} \end{pmatrix} \begin{pmatrix} x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \end{pmatrix} .$$

復号に用いる mds_{L}^{-1} は以下のとおりである。

入出力仕様 $X_{(32)} = mds_{\mathrm{L}}^{-1}(Y_{(32)})$
関数定義
$y_{1(8)} \| y_{2(8)} \| y_{3(8)} \| y_{4(8)} = Y_{(32)}$,
$X_{(32)} = x_{1(8)} \| x_{2(8)} \| x_{3(8)} \| x_{4(8)}$,

$$\begin{pmatrix} x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \end{pmatrix} = \begin{pmatrix} \mathtt{82} & \mathtt{C4} & \mathtt{34} & \mathtt{F6} \\ \mathtt{F6} & \mathtt{82} & \mathtt{C4} & \mathtt{34} \\ \mathtt{34} & \mathtt{F6} & \mathtt{82} & \mathtt{C4} \\ \mathtt{C4} & \mathtt{34} & \mathtt{F6} & \mathtt{82} \end{pmatrix} \begin{pmatrix} y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \end{pmatrix} .$$

ここで、8 ビット・データ $x_{(8)}$ と行列要素 a (16 進表現の整数) は、次式によって GF(2^8) の元 (多項式表現) と関係付ける。

$$x_{(8)} \Leftrightarrow \sum_{i=1}^{8} x_{i(1)} z^{8-i} ,$$
$$a = \sum_{i=0}^{7} a_i 2^i \Leftrightarrow \sum_{i=0}^{7} a_i z^i .$$

GF(2^8) の原始多項式は $z^8 + z^6 + z^5 + z + 1$ である。

3.3.7 MDS_{H} 関数

MDS_{H} 関数は、64 ビット・データを 8 分割して得られる 8 ビット・サブデータ $x_{i(8)} (1 \leq i \leq 8)$ に対する行列の積として与えられる。ここで、$x_{i(8)}$ は GF$(2)^8$ の元とみなす。

暗号化に用いる MDS_{H} 関数は以下のとおりである。

入出力仕様 $Y_{(64)} = MDS_{\mathrm{H}}\left(X_{(64)}\right)$
関数定義
$x_{1(8)} \| x_{2(8)} \| \cdots \| x_{8(8)} = X_{(64)}$,
$Y_{(64)} = y_{1(8)} \| y_{2(8)} \| \cdots \| y_{8(8)}$,

$$\begin{pmatrix} y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \\ y_{5(8)} \\ y_{6(8)} \\ y_{7(8)} \\ y_{8(8)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \\ x_{5(8)} \\ x_{6(8)} \\ x_{7(8)} \\ x_{8(8)} \end{pmatrix} .$$

復号の際に用いる MDS_{H}^{-1} 関数は以下のとおりである。

入出力仕様 $X_{(64)} = MDS_{\mathrm{H}}^{-1}(Y_{(64)})$
関数定義
$y_{1(8)} \| y_{2(8)} \| \cdots \| y_{8(8)} = Y_{(64)}$,
$X_{(64)} = x_{1(8)} \| x_{2(8)} \| \cdots \| x_{8(8)}$,

$$\begin{pmatrix} x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \\ x_{5(8)} \\ x_{6(8)} \\ x_{7(8)} \\ x_{8(8)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \\ y_{5(8)} \\ y_{6(8)} \\ y_{7(8)} \\ y_{8(8)} \end{pmatrix} .$$

3.3.8 $P^{(n)}$ 関数

鍵スケジュールで用いる $P^{(n)}$ 関数は、4 個の n ビット変数 $x_{i(n)}$ $(i = 1, 2, 3, 4)$ の連接 $X_{(4n)}$ に対する次式の線形変換である。各要素は $\mathrm{GF}(2)^n$ の元とみなす。

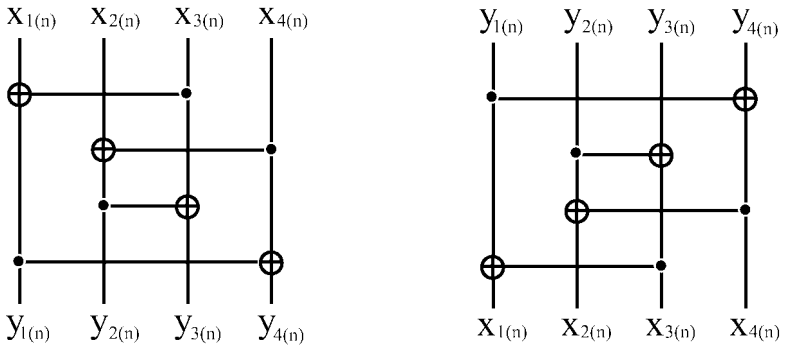
入出力仕様 $Y_{(4n)} = P^{(n)}\left(X_{(4n)}\right)$
関数定義
$x_{1(n)} \| x_{2(n)} \| x_{3(n)} \| x_{4(n)} = X_{(4n)}$,
$Y_{(4n)} = y_{1(n)} \| y_{2(n)} \| y_{3(n)} \| y_{4(n)}$,

$$\begin{pmatrix} y_{1(n)} \\ y_{2(n)} \\ y_{3(n)} \\ y_{4(n)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_{1(n)} \\ x_{2(n)} \\ x_{3(n)} \\ x_{4(n)} \end{pmatrix} .$$

復号に用いる $P^{(n)^{-1}}$ は以下のとおりである。

入出力仕様 $X_{(4n)} = P^{(n)^{-1}}\left(Y_{(4n)}\right)$
関数定義
$y_{1(n)} \| y_{2(n)} \| y_{3(n)} \| y_{4(n)} = Y_{(4n)}$,
$X_{(4n)} = x_{1(n)} \| x_{2(n)} \| x_{3(n)} \| x_{4(n)}$,

$$\begin{pmatrix} x_{1(n)} \\ x_{2(n)} \\ x_{3(n)} \\ x_{4(n)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{1(n)} \\ y_{2(n)} \\ y_{3(n)} \\ y_{4(n)} \end{pmatrix} .$$



$P^{(n)}$ 関数　　$P^{(n)^{-1}}$ 関数

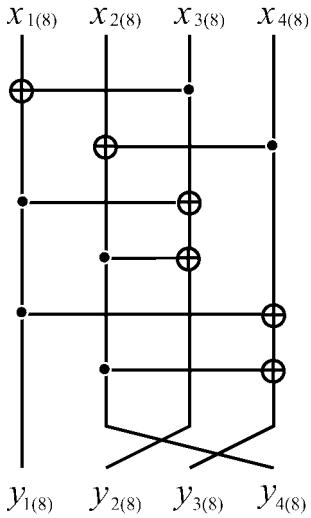
図 6: $P^{(n)}$ 関数と $P^{(n)^{-1}}$ 関数

3.3.9 M_5 関数

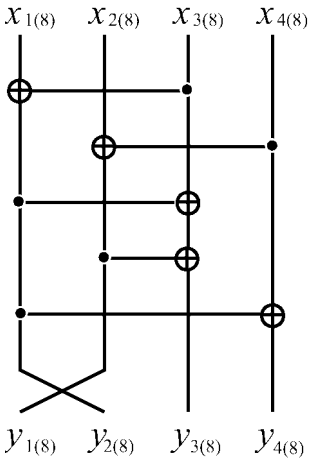
鍵スケジュールの平文サイドで用いる M_5 関数は、32 ビット変数 $X_{(32)}$ に対する以下の線形変換である。

入出力仕様 $Y_{(32)} = M_5\left(X_{(32)}\right)$
関数定義
$x_{1(8)} \| x_{2(8)} \| x_{3(8)} \| x_{4(8)} = X_{(32)}$,
$Y_{(32)} = y_{1(8)} \| y_{2(8)} \| y_{3(8)} \| y_{4(8)}$,

$$\begin{pmatrix} y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \end{pmatrix} .$$



M_5 関数



M_{B} 関数

図 7: M_5 関数と M_{B} 関数

3.3.10 M_{B} 関数

鍵スケジュールで用いる M_{B} 関数は、32 ビット変数 $X_{(32)}$ に対する以下の線形変換である。

入出力仕様 $Y_{(32)} = M_{\mathrm{B}}\left(X_{(32)}\right)$
関数定義
$x_{1(8)} \| x_{2(8)} \| x_{3(8)} \| x_{4(8)} = X_{(32)}$,
$Y_{(32)} = y_{1(8)} \| y_{2(8)} \| y_{3(8)} \| y_{4(8)}$,

$$\begin{pmatrix} y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \end{pmatrix} .$$

3.3.11 F_σ 関数

鍵スケジュールで用いる F_σ 関数は、32 ビット変数 $X_{(32)}$ を入力とする 32 ビット出力の非線形関数である。

入出力仕様 $Y_{(32)} = F_\sigma\left(X_{(32)}\right)$
関数定義
$x_{1(8)} \| x_{2(8)} \| x_{3(8)} \| x_{4(8)} = X_{(32)}$,
$Y_{(32)} = P^{(8)}\left(s(x_{1(8)}) \| s(x_{2(8)}) \| s(x_{3(8)}) \| s(x_{4(8)})\right)$.

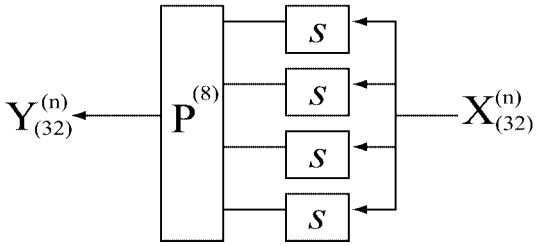


図 8: F_σ 関数

参考文献

[1] E.Biham and A.Shamir, "Differential cryptanalysis of DES-like cryptosystems," Journal of Cryptology, 4, No.1, pp.3-72, 1991.

[2] M.Matsui, "Linear cryptanalysis method for DES cipher," Eurocrypt'93, LNCS 765, pp.386-397, 1994.

[3] J.Daemen, L.R.Knudsen, V.Rijmen, "The block cipher Square," Fast Software Encryption, LNCS 1267, pp.149-165, 1997.

[4] J.Daemen, V.Rijmen, "AES Proposal: Rijndael," http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaeldocV2.zip

[5] 大熊・村谷・佐野・川村, "Specification and assessment of the cipher Hierocrypt," ISEC2000-7, 2000.

[6] K.Ohkuma, H.Muratani, F.Sano, and S.Kawamura, "The block cipher Hierocrypt," SAC 2000, LNCS 2012, pp.72–88, 2000.

[7] S.Hong, S. Lee, J. Lim, J. Sung, and D. Cheon, "Provable Security against Differential and Linear Cryptanalysis for the SPN Structure," FSE 2000, LNCS 1978, pp.273–283, 2000.

[8] H.Shimizu, private communication, 2000.

[9] L.R.Knudsen, T.A.Berson, "Truncated Differentials of SAFER," FSE'96, LNCS 1039, pp.15-25, 1996.