仕様書
MULTI-S01 暗号

（株）日立製作所

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

目次

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

1. 準備

1.1. 変数、記号の表記について

本仕様書では、整数の演算子を以下のように定義する。

⊕：ビットごとの排他的論理和、変数の長さは文脈から明らか

⊗：有限体 $\mathbf{F}2^{64}$ 上の乗算（後述）

＋：整数加算

×：整数乗算

OR：ビット毎の論理和演算（32 ビット）

NOT：１変数演算子。ビット毎の論理反転（32 ビット）

ROTL$_k$(X)：32 ビットのバイナリ X に対する k ビット左巡回シフト

変数の長さを明示する場合、そのビット長を括弧付きスーパースクリプトとして表記する。たとえば、変数 X のビット長が 8 ビットである場合 $X^{(8)}$ と表記する。

また、列については、変数名にサブスクリプトとして引数を書くことで要素を表す。たとえば、変数列 X の引数 8 に相当する要素を X_8 と書き表す。列の引数は、特に明記しない限り、最初の要素を 1 とする。

1.2. ワード中のバイトデータの順序（エンディアン）

鍵、平文、暗号文、冗長データ、初期値はバイト単位の列として扱う。これらは 64 ビットのデータ型との変換の際、Big-Endian により変換される；たとえば、64 ビットのワード列 Y に、バイト文字列 X を格納する手順は以下のようになる。

$$Y_i = \sum_{j=1,\ldots,8} X_{8(i-1)+j} 2^{(64-8j)}$$

このアルゴリズムではバイト列を 64 ビットブロックへの埋め込みを行う。この場合、64 ビットのデータ型に埋め込まれるバイト列 X_i の順序は以下のようになる。

[MSB]$X1$ $X2$ $X3$ $X4$ $X5$ $X6$ $X7$ $X8$[LSB]

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

1.3. 有限体上の演算

MULTI-S01 では、有限体 $\mathbf{F}2^{64}$ 上の演算を用いる。ここでは、この演算の実装に必要な性質と演算方法を簡単に説明する。

本稿では 64 ビットの変数 A と B の $\mathbf{F}2^{64}$ の乗算（以下、演算子を記号⊗で示す）を考えるとき、変数、B はそれぞれの値に応じて、以下のようにして 63 次以下の x についての多項式（係数は 0 または 1）を対応づける。

- 1 ビット値 a_i={0,1}を A の上位から i 番目のビット値であると定義する。
  すなわち、$(a_1, a_2, a_3,...,a_{64})=A$。これら a_i を用いて A に対応づけられる多項式 $A(x)$を以下のように定義する。
  $A(x) = a_1x^{63} + a_2x^{62} + a_3x^{61} + ... + a_{63}x + a_{64} = \Sigma_{i=1...64}\ a_ix^{(64-i)}$
- b_i={0,1}についても A と同様に、多項式 $B(x)$を対応づける。
  $(b_1, b_2, b_3,...,b_{64})=B$
  $B(x) = b_1x^{63} + b_2x^{62} + b_3x^{61} + ... + b_{63}x + b_{64} = \Sigma_{i=1...64}\ b_ix^{(64-i)}$

ここで多項式の剰余演算のために記号を定義する。二つの多項式 $A(x)$、$P(x)$について $A(x)$ mod $P(x)$を、$A(x)$を $P(x)$で割ったときの剰余、と定義する。

本稿では $P(x)=x^{64}+x^4+x^3+x+1$ を使って、多項式の（剰余）乗算結果である $C(x)$を次のように定義する。

$$C(x) = A(x) \times B(x) \bmod P(x)$$

係数はすべて 2 の剰余（2 で割った余り）で考える。すなわち、0+0=0, 0+1=1, 1+1=0, 0×0=0, 0×1=0, 1×1=1（これらはすべて可換（第一引数と第二引数を入れ替えても同じ演算結果）である）。従って、加算は排他的論理和、乗算は論理積としてあつかうことができる。$P(x)$は 64 次の多項式なので $C(x)$は高々63 次多項式、項の数は高々64 項である。得られた多項式 $C(x)$に対応する 64 ビットの値 Cを、乗算の結果として扱う。つまり

- 1 ビット値 c_i={0,1}を多項式 $C(x)$の $64-i$ 次項の係数と定義する。すなわち、
  $C(x) = c_1x^{63} + c_2x^{62} + c_3x^{61} + ... + c_{63}x + c_{64} = \Sigma_{i=1...64}\ c_ix^{(64-i)}$
- Cを上位から $c_1, c_2, ..., c_{64}$ を並べた値とする。すなわち、
  $C=(c_1, c_2, c_3,...,c_{64})$

具体例

16 進数表記で A,Bを以下の値とし、$C=A \otimes B$を計算する。

A = `01234567 89abcdef`

B = `fedcba98 76543210`

これらを 2 進数表記すると以下のようになる。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

A = 0000 0001 0010 0011 0100 0101 0110 0111

1000 1001 1010 1011 1100 1101 1110 1111

B = 1111 1110 1101 1100 1011 1010 1001 1000

0111 0110 0101 0100 0011 0010 0001 0000

よって A, B に対応づけられる多項式 $A(x), B(x)$は以下のようになる。

$$A(x) = x^{56}+x^{53}+x^{49}+x^{48}+x^{46}+x^{42}+x^{40}+x^{38}+x^{37}+x^{34}+x^{33}+x^{32}$$
$$+x^{31}+x^{27}+x^{24}+x^{23}+x^{21}+x^{19}+x^{17}+x^{16}+x^{15}+x^{14}+x^{11}+x^{10}+x^{8}+x^{7}+x^{6}+x^{5}+x^{3}+x^{2}+x+1$$

$$B(x) = x^{63}+x^{62}+x^{61}+x^{60}+x^{59}+x^{58}+x^{57}+x^{55}+x^{54}+x^{52}+x^{51}+x^{50}+x^{47}+x^{45}+x^{44}+x^{43}+x^{41}$$
$$+x^{39}+x^{36}+x^{35}+x^{30}+x^{29}+x^{28}+x^{26}+x^{25}+x^{22}+x^{20}+x^{18}+x^{13}+x^{12}+x^{9}+x^{4}$$

$D(x)= A(x) \times B(x)$とする（剰余を取らない）。このとき $D(x)$は

$$D(x)=A(x)x^{63}+A(x)x^{62}+A(x)x^{61}+A(x)x^{60}+A(x)x^{59}+A(x)x^{58}+A(x)x^{57}+A(x)x^{55}$$
$$+A(x)x^{54}+A(x)x^{52}+A(x)x^{51}+A(x)x^{50}+A(x)x^{47}+A(x)x^{45}+A(x)x^{44}+A(x)x^{43}$$
$$+A(x)x^{41}+A(x)x^{39}+A(x)x^{36}+A(x)x^{35}+A(x)x^{30}+A(x)x^{29}+A(x)x^{28}+A(x)x^{26}$$
$$+A(x)x^{25}+\mathrm{A}(x)x^{22}+A(x)x^{20}+A(x)x^{18}+A(x)x^{13}+A(x)x^{12}+A(x)x^{9}+A(x)x^{4}$$
$$= x^{119}+x^{118}+x^{117}+x^{109}+x^{108}+x^{107}+x^{103}+x^{102}+x^{100}+x^{99}$$
$$+x^{94}+x^{93}+x^{91}+x^{87}+x^{83}+x^{78}+x^{76}+x^{71}+x^{69}+x^{68}$$
$$+x^{62}+x^{55}+x^{53}+x^{46}+x^{45}+x^{44}+x^{43}+x^{39}+x^{36}+x^{35}$$
$$+x^{29}+x^{27}+x^{23}+x^{22}+x^{19}+x^{12}+x^{7}+x^{6}+x^{5}+x^{4}$$

ここで $D(x)$の $P(x)= x^{64}+x^{4}+x^{3}+x+1$ による剰余を考える。明らかな計算法として以下の規則に従い、64 次以上の項を次数の低い項に置き換えることで計算できる。

$$x^{64}=x^{4}+x^{3}+x+1$$
$$x^{65}=x^{5}+x^{4}+x^{2}+x$$
$$x^{66}=x^{6}+x^{5}+x^{3}+x^{2}$$
...
$$x^{124}=x^{64}+x^{63}+x^{61}+x^{60} = (x^{4}+x^{3}+x+1)+ x^{63}+x^{61}+x^{60} = x^{63}+x^{61}+x^{60}+x^{4}+x^{3}+x+1$$
$$x^{125}=x \times x^{124}=x^{64}+x^{62}+x^{61}+x^{5}+x^{4}+x^{2}+x = (x^{4}+x^{3}+x+1)+x^{62}+x^{61}+x^{5}+x^{4}+x^{2}+x$$
$$= x^{62}+x^{61}+x^{5}+x^{3}+x^{2}+1$$
$$x^{126}=x \times x^{125}= x^{63}+x^{62}+x^{6}+ x^{4}+x^{3}+x$$

これらより、$C(x)= D(x) \bmod P(x)$は以下のようになる。

$$C(x)=x^{62}+x^{59}+x^{55}+x^{49}+x^{46}+x^{45}+x^{44}+x^{43}+x^{41}+x^{39}+x^{37}+x^{36}+x^{34}+x^{32}$$
$$+x^{30}+x^{28}+x^{27}+x^{26}+x^{24}+x^{23}+x^{20}+x^{18}+x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{5}$$

これを 63 次の項から係数だけをならべたデータに変化し、乗算の結果である cを得る。

c= 0100 1000 1000 0010 0111 1010 1011 0101

0101 1101 1001 0111 0110 1111 1010 0000(２進数表記)

= 48827ab5 5d976fa0(16 進数表記)

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

2. アルゴリズム

2.1. 全体概要

MULTI-S01 暗号は暗号化処理、復号化処理からなる。暗号化処理、復号化処理はそれぞれ、擬似乱数生成器とデータ攪拌部分のふたつの部分から構成される。ただし、暗号化処理のデータ攪拌部分と、復号化処理のデータ攪拌部分は逆変換の関係になっている。

擬似乱数生成器は秘密鍵 Kから鍵ストリーム A,B,Sを（処理するデータの長さに応じた長さだけ）生成する。

暗号化処理では、メッセージ M(m(≤ 2^{38}–128)ビット、より長いメッセージについては後で述べる)、冗長性 R (64 ビット)、秘密鍵 K (256 ビット)をそれぞれバイト列によるデータ($M^{(8)}_i$ (i =1,...,⌈m/8⌉), $R^{(8)}_i$ (i =1,..., 8), $K^{(8)}_i$(i =1,..., 32)として入力する。暗号化処理の出力は暗号文 Cであり、Cの長さは 64×(⌈m/64⌉+ 2)ビットで、バイト列として出力する。

これに対応する復号化処理では、暗号文 C (c ビット)、冗長性 R (64 ビット)、秘密鍵 K (256 ビット)をそれぞれバイト列によるデータ($C^{(8)}_i$ (i =1,...,⌈c/8⌉), $R^{(8)}_i$ (i =1,..., 8), $K^{(8)}_i$(i =1,..., 32)として入力する。復号化処理の出力は復号化結果 M'または改竄検出信号であり、メッセージが出力される場合には、これをバイト列として出力する。暗号化・復号化処理の内部は 64 ビットのブロックごとの処理で構成され、処理全体のブロックの数を n=⌈m/ 64⌉ + 2 とする。擬似乱数生成器は、Kを入力として、A (64 ビット)と B (64×(n+2)ビット)、S (64 ビット)を出力する。よって、暗号化処理のデータ攪拌部分は、M,R,A,B,Sを入力として Cを出力し、復号化処理のデータ攪拌部分は、C,R,A,B,Sを入力とし、復号化結果 M'または改竄検出信号を出力する。暗号化処理と復号化処理の全体の構成をそれぞれ図 2.1、図 2.2にモデルを示す。

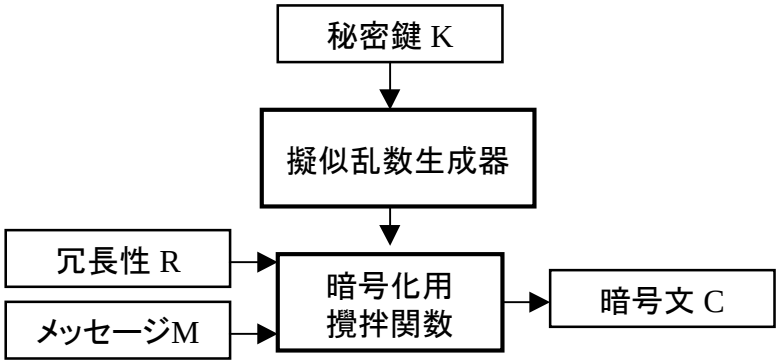

図 2.1 暗号化の場合の全体構造

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

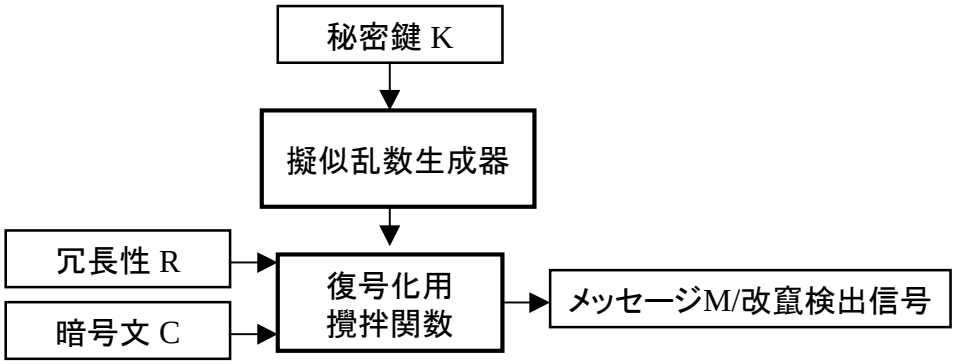

図 2.2 復号化の場合の全体構造

2.2. 機能

安全に生成・共有された秘密鍵を用いた場合、攻撃者は（秘密鍵を知らないため）暗号文のみの情報から平文についての情報を得ることができない(得るための計算量が非常に大きく攻撃の成功が非現実的)。さらに、冗長性が安全に共有された場合（ただし秘密である必要はない)、上の機能に加え暗号文の改竄検出機能も有する。これは「(秘密鍵を知らない）攻撃者の改竄が成功する確率が非常に小さい」ことを保証できる、ということである。

2.3. 擬似乱数生成器

擬似乱数生成器は、Kから A(非ゼロ),B,Sを生成する。nを処理ブロック数(後述)とすると、これらデータのビット長はそれぞれ A(64 ビット)、B(64×n ビット)、S(64 ビット)となる。n は以下の式で定義される。

(暗号化) $n=\lceil m/64 \rceil + 2$, ただし m はメッセージビット数

(復号化) $n=\lceil c/64 \rceil$, ただし c は暗号文ビット数

以下に具体的なAとBの生成法について示す。

生成法

入力: 秘密鍵 K

出力: 64 ビット非ゼロ擬似乱数 A

(64× n)ビット擬似乱数 B

64 ビット擬似乱数 S

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

(1) 擬似乱数生成器を鍵 Kで初期化

(2) 擬似乱数生成器により 64 ビットの擬似乱数 A を生成する

(3) もし A =0 ならば(2)に戻る

(4) 擬似乱数生成器により(64× n)ビットの擬似乱数 B を生成する

(5) 最後に 64 ビットの擬似乱数 S を生成する

2.4. 擬似乱数生成器 PANAMA

MULTI-S01 では、擬似乱数を生成するために、PANAMA 擬似乱数生成器を使う。よって、ここでは PANAMA の数学的な仕様説明を行う。厳密には、PANAMA とは、1998 年に J. Daemen と C. Clapp が提案した暗号モジュールであり、ストリーム暗号、およびハッシュ関数の構成方法として用いることができる。本稿ではハッシュ関数としての使用はしないので、PANAMA というときは単に PANAMA を使った擬似乱数生成器を意味する。

PANAMA は秘密鍵 K と乱数列番号 Q を入力とし、任意の長さの擬似乱数を生成する。乱数列番号 Q の使い方は後で述べる。PANAMA は 32 ビット幅の基本演算の組み合わせで構成される。以下、32 ビットを 1 ワードと呼ぶことにする。

PANAMA は、

・17 ワード(544 ビット)の主レジスタ a

・32×8 ワード(8192 ビット)の副レジスタ b

・レジスタ a の非線形変換 ρ

・レジスタ b の変換 λ

からなる暗号モジュールで、

(1) reset モード(レジスタのリセット)

(2) push モード(鍵情報などの入力)

(3) pull モード(レジスタの攪拌、および鍵ストリームの生成)

の 3 つのモードを持つ。擬似乱数生成を行う場合は次のスケジュールに従って PANAMA は動作する。

表 2.1擬似乱数生成 PANAMA のモードスケジュール

時間経過 t(ラウンド数)	モード	入力(256 ビット)	出力(256 ビット)
−34	reset	−	−
−33	push	K	−
−32	push	Q	−
−31, ..., 0	pull	−	−
1, 2, ..., i, ...	pull	−	擬似乱数列 Out_i

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

まず、モードについて解説する。ここでは PANAMA の原著に示される表記との混同がないように、列の番号を 0 から始まるものとして扱う。以下では、主レジスタ a の i 番目のワードを $a_i(0 \le i < 17)$と表す。また、副レジスタ b の j 番目のステージ(1 ステージ＝8 ワード)を $b^j(0 \le j < 31)$、b^j の i 番目のワードを $b^j_i(0 \le i < 8)$と表す。

reset モード

reset モードは、主レジスタ、副レジスタのリセットを行う。すなわち、a_i, b^j_i の値をすべて 0 にする。

push モード

push モードは、鍵情報などをレジスタに入力するモードで、8 ワード(256 ビット)を入力する。出力はない(図 2.3上)。

アルゴリズム

入力：入力情報 p(256 ビット)

出力：無し

(1) $x = b^{16}$

(2a)レジスタ b の更新($b = \lambda(b) = \lambda(b, p)$. 後述)

(2b)レジスタ a の更新($a = \rho(a) = \rho(a, p, x)$. 後述)

主レジスタ a の変換に用いる副レジスタ b の値 x(1 ステージ)は、b の更新以前のものであることに注意する。

pull モード

pull モードは、レジスタの攪拌、鍵ストリームの生成を行うモードである。入力情報はなく、ラウンドごとに 8 ワード(256 ビット)の乱数を生成する(図 2.3下)。

アルゴリズム

入力：ラウンド数 n

出力：擬似乱数(256 ビット)

$0 \le k < n$ について以下を繰り返す。

(1) $K = (a_9, ..., a_{16})$を出力

(2) $x = b^{16}$, $q = b^4$

(3a) レジスタ b の更新($b = \lambda(b) = \lambda(b, a)$)

(3b) レジスタ a の更新($a = \rho(a) = \rho(a, q, x)$)

push モードと同様、主レジスタ a の変換に用いる副レジスタ b の値 x(1 ステージ)は、b の更新以前のものであることに注意する。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

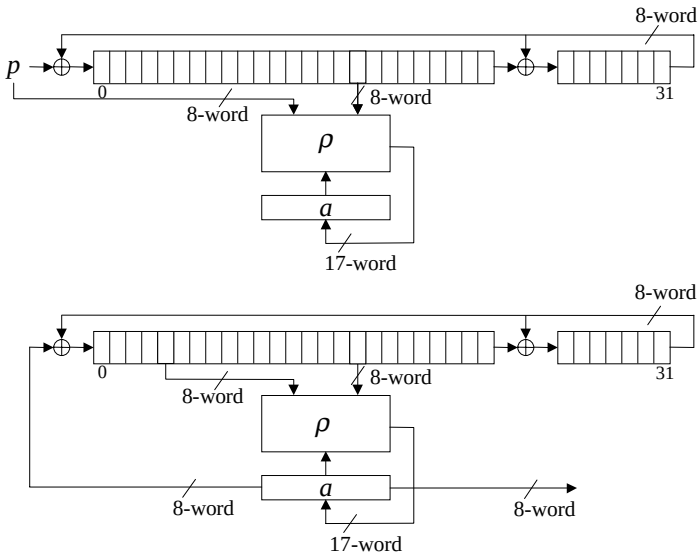

図 2.3　PANAMA **の** push **モード(上)と** pull **モード(下)**

副レジスタ *b* とその更新

副レジスタ *b* は 8 ワード(＝1 ステージ)を単位とする線形フィードバックシフトレジスタ(LFSR)の変形である。レジスタ *b* の更新は次のように行う(図 2.4参照)。

アルゴリズム

更新後のレジスタの状態を *d* で表す(*d*=λ(*b*))。λは次のように記述できる。

$j \neq 0, 25$ のとき、$d^j = b^{j-1}$

$d^0 = b^{31} \oplus q$

$d^{25}{}_i = b^{24}{}_i \oplus b^{31}{}_{i+2 \bmod 8}$ $(0 \le i < 8)$

q はモードによって変わる変数で、push モードの場合には入力情報 *p* である。pull モードの場合には主レジスタ *a* から、

$q_i = a_{i+1}, 0 \le i < 8$

で値を定める。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

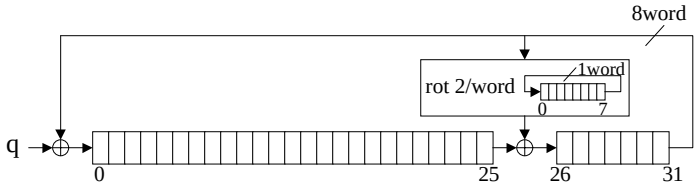

図 2.4　線形フィードバックシフトレジスタ *b* の更新

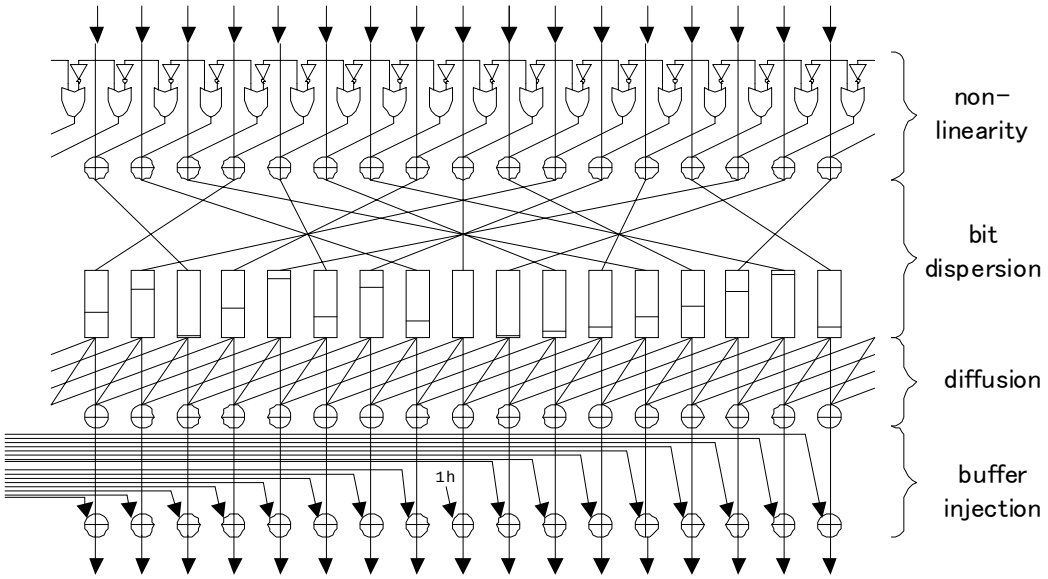

図 2.5　非線形変換 *ρ*

非線形変換 *ρ*

主レジスタ a の変換 ρ は、4 種類の変換 $\gamma, \pi, \theta, \sigma$ の合成変換で与えられる(図 2.5参照)。

$$\rho(a) = \sigma(\theta(\pi(\gamma(a))))$$

γ は、次の式で定義される非線形変換である。

$$\gamma(a_i) = a_i \oplus (a_{i+1} \text{ OR } (\text{NOT } a_{i+2})),\ 0 \le i < 17$$

π は、ワード単位での置換と 32 ビットの左巡回シフトからなり、以下のような式で定義される。

$$\pi(a_i) = \text{ROTL}_{i(i+1)/2 \bmod 32}(a_{7i \bmod 17}),\ 0 \le i < 17$$

θ は、3 つのワードの排他的論理和からなる。

$$\theta(a_i) = a_i \oplus a_{i+1} \oplus a_{i+},\ 0 \le i < 17$$

σ は、入力情報、および副レジスタ b のステージとの排他的論理和を行う変換である。

$$\sigma(a_0) = a_0 \oplus 1$$

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

$\sigma(a_{i+1}) = a_{i+1} \oplus l_i,\ 0 \le i < 8$

$\sigma(a_{i+9}) = a_{i+9} \oplus b^{16}{}_i,\ 0 \le i < 8$

lは、push モードでは入力情報 p、pull モードでは b^4 である。

2.5. PANAMA による A,B,S の生成

ここでは具体的に PANAMA を使った A,B,S の生成を定義する。生成する B の長さは n ブロック（１ブロックは 64 ビット）とする。パラメータ Q の使い方は次に説明する。

アルゴリズム

入力：鍵 K(256 ビット)

パラメータ Q(256 ビット)

出力：鍵ストリーム A, B, S

(1) PANAMA を初期化し K, Q を入力する

(2) 64 ビットの PANAMA の出力を取得、A とする

(3) もし A=0 ならば(2)に戻る

(4) 続けて 64n ビットの PANAMA の出力を取得、B とする

(5) 続けて 64 ビットの PANAMA の出力を取得、S とする

乱数列番号Qの使用法と注意

PANAMA への入力である Q は主に次の 2 つの目的で用いられる。

並列処理：擬似乱数の生成を複数のプロセッサに割り当て、これを Q で指定する

再同期：再同期（同期をしたい頻度でQを更新する）を可能とする

非常に長い平文に対する分割処理：3章で詳しく述べる

この変数 Q は秘密である必要もなく、乱数である必要もないが、暗号化・復号化で共有しておく必要がある。また、別の注意するべき点として、平文の暗号化ではいつもかならず新しい（今まで装置が発生したことがない）擬似乱数を使わなければならない。これは安全性に関する技術的理由によるものである。例えば、異なる平文、冗長符号に対して、同じ鍵、同じ変数 Q による暗号処理を行ってはならない。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

2.6. 暗号化処理のデータ攪拌部

暗号化処理のデータ攪拌部は、データパディング部と主要攪拌部から構成される。データパディング部では M,S,R から中間値 P(64n ビット)を生成し、主要攪拌部では P,A,B から C を生成する。

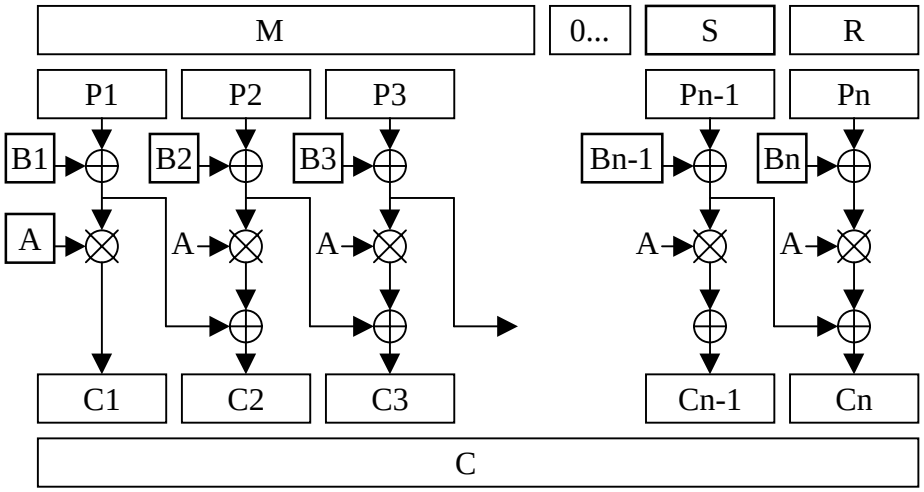

図 2.6 暗号化処理の主要攪拌部

データパディング部では、平文バイト列 $M^{(8)}{}_i$ (i=1,...,⌈m/8⌉)、鍵ストリーム $S^{(8)}{}_i$ (i=1,..., 8)、冗長データ $R^{(8)}{}_i$ (i=1,..., 8)から以下のようにして、中間値である 64 ビットデータの列 $P^{(64)}{}_i$ (i =1,...,n)を生成する。ただし、P の生成で必要でありながら、入力として与えられない $M^{(8)}{}_i$ は 0 として扱う(これは 64 ビットのレジスタに格納するための Mへの 0 パディングに等しい)。

$$P^{(64)}i = \sum_{j=1,...,8} M^{(8)}{}_{8(i-1)+j}\, 2^{(64-8j)} \quad (i = 1,...,n-2)$$

$$P^{(64)}{}_{n-1} = \sum_{j=1,...,8} S^{(8)}{}_j\, 2^{(64-8j)}$$

$$P^{(64)}{}_n = \sum_{j=1,...,8} R^{(8)}{}_j\, 2^{(64-8j)}$$

主要攪拌部では、上で定義した $P^{(64)}i$ (i = 1,...,n)と $F^{(64)}{}_0$ (=0), それに鍵ストリームである $A^{(64)}$, $B^{(64)}{}_i$ (i = 1,...,n)から以下のようにして暗号文 $C^{(64)}{}_i$ (i = 1,...,n)を生成する。

$$F^{(64)}{}_i = P^{(64)}{}_i \oplus B^{(64)}{}_i$$

$$C^{(64)}{}_i = (F^{(64)}{}_i \otimes A^{(64)}) \oplus F^{(64)}{}_{i-1} \quad (i = 1,...,n)$$

暗号文 $C'^{(64)}$をバイト列 $C^{(8)}$に変換したものを暗号文とする。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

図 2.6は上記の暗号化処理をブロック図で示したものである。

2.7. 復号化処理のデータ攪拌部

復号化処理のデータ攪拌部は、主要攪拌部と検査処理部から構成される。主要攪拌部では暗号文 C (64×n ビット：受信した暗号文が 64 ビットの倍数でない場合、処理を中止、改竄検出信号を出力する),鍵ストリーム A,B,S から中間値 P(64×n ビット)を生成し、検査処理部では P’と、事前に共有されている冗長性 $R^{(8)}{}_i$ (i=1,...,8)からメッセージ M’または改竄検出信号を出力する。

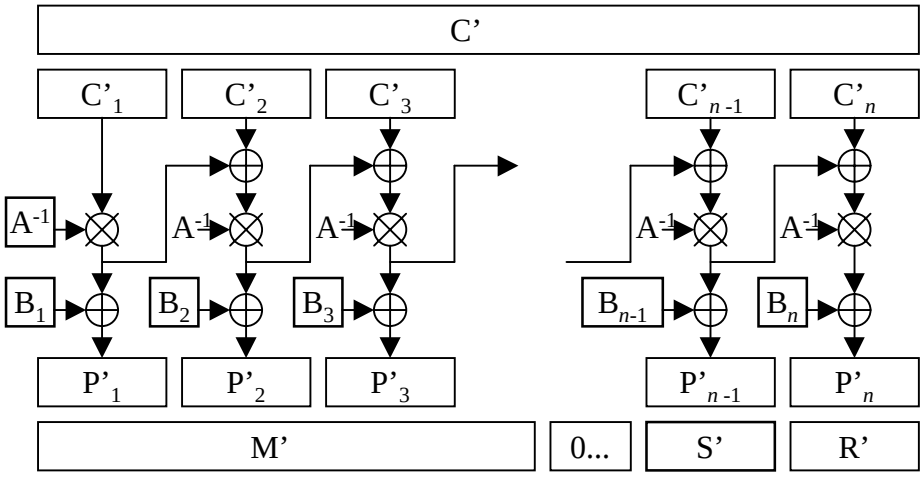

図 2.7 復号化処理の主要攪拌部

主要攪拌部では、バイト列データである $C'^{(8)}{}_i$,$S^{(8)}{}_i$をそれぞれ 64 ビットのデータ型の列 $C'^{(64)}{}_i$ (i=1,...,n), $S^{(64)}$に変換し、$A^{(64)}$,$B^{(64)}{}_i$ (i=1,...,n)を用いて以下の式により $P'^{(64)}{}_i$を生成する。

$$F'^{(64)}{}_0 = 0$$

$$F'^{(64)}{}_i = (C'^{(64)}{}_i \oplus F'^{(64)}{}_{i-1}) \otimes (A^{(64)})^{-1} \quad (i = 1,...,n)$$

$$P'^{(64)}{}_i = F'^{(64)}{}_i \oplus B^{(64)}{}_i \quad (i = 1,...,n)$$

検査処理部では、$P'^{(64)}{}_i$ (i = 1,...,n)を次のように分割する：

$$M'^{(64)}{}_i = P'^{(64)}{}_i \quad (i=1,...,n-2)$$

$$S'^{(64)} = P'^{(64)}{}_{n-1}$$

$$R'^{(64)} = P^{(64)}{}_n$$

ここで $R^{(8)}$を 64 ビットデータに変換したものを $R^{(64)}$とする。$S^{(64)}$=$S'^{(64)}$、かつ、$R^{(64)}$=$R'^{(64)}$のとき、$M'^{(64)}i$ (i=1,...,n–2)をバイト列に変換しメッセージとして出力する。そうでない場合は、改竄検出信号を出力する。図 2.7は上記の復号化処理をブロック図で示したものである。A^{-1} は（ $\mathbf{F}2^{64}$ における）A の逆数である。有限体の一般的な性質から $A^{-1} = A^q \oplus 1$ (ただし $q = 2^{64}-1$)。また、効率的な逆元を求めるアルゴリズムに Euclid の互除法を使った方法や、almost inverse algorithm[1]などがある。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

3. 大きなデータの処理

ブロック数が 2^{32} を超えるデータについては、2^{32} ブロック毎の処理となるようにメッセージを分割し（2^{38}–128 ビット）、それぞれの 2^{32} ブロックについて上記と同様の暗号処理、復号処理を行う。ただし、各々の 2^{32} ブロックには、異なる冗長性、異なる PANAMA 乱数を用いて処理を行う。PANAMA には、乱数列番号 Q があるので、これを用いることにより同一の鍵でも異なる乱数列を生成できる。

これまで説明した MULTI-S01 による暗号化処理を **M-S01**（K,Q,R,M）とする（K,Q,R,M はそれぞれ秘密鍵、乱数列番号、冗長性、メッセージ）。例えば、M を非常に長いメッセージとし、(2^{38}–128)ビットごとに分割したメッセージをμ_1,μ_2,..., μ_k とする。共有された秘密鍵を K とする。この場合の M の暗号文 C は

$$C = \textbf{M-S01}(K,0,0,\mu_1) \,||\, \textbf{M-S01}(K,1,1,\mu_2) \,||\, \ldots \,||\, \textbf{M-S01}(K,k,k,\mu_k)$$

となる。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved

参考文献

[1] Schroeppel, R., Orman, H., O'Malley, S., Spatscheck, O., "Fast Key Exchange with Elliptic Curve Systems," *Advances in Cryptology—CRYPTO'95, LNCS Vol.963, Springer-Verlag*, 1995.

本書に記載された会社名、製品名は、それぞれの会社の商標もしくは登録商標である。

Copyright © 2000, 2001 Hitachi, Ltd. All Rights Reserved