Enocoro-128v2の安全性評価

桑門秀典
神戸大学大学院工学研究科

平成23年1月31日

第1章　序論

Enocoro は日立製作所によって提案されたハードウエア向け疑似乱数生成器である．Enocoro の初期版は 2007 年に提案され，これまで改良と安全性解析が続けられてきた．そのため，提案者や第三者による Enocoro に関する解析結果が多く報告されている．CRYPTREC に応募された方式は，Enocoro の中で Enocoro-128v2 と呼ばれる方式である．

Enocoro-128v2 は，2010 年に発表された方式であり，Enocoro の中では最新の方式である．Enocoro-128v2 は，128 ビットの鍵，64 ビットの初期値 (IV) を入力として，最大で 2^{64} バイトの疑似乱数系列を生成する．内部状態は 272 ビット (34 バイト) あり，PANAMA 型と呼ばれる構造をしている．

Enocoro-128v2 の自己評価書 [5] では，下記の攻撃について述べられている．

Time-Memory-Trade-Off 攻撃	推測決定攻撃	分割統治攻撃
代数的な攻撃	線形確率に基づく識別攻撃	再同期攻撃
関連鍵攻撃		

自己評価書では，いずれの攻撃に対しても Enocoro-128v2 は十分な安全性を有すると結論づけている．自己評価書における安全性評価では，計算機実験による評価結果が述べられており，評価方法の詳細は参考文献 [6, 7, 9] を引用するにとどまっている．しかし，参考文献においても，その評価結果が正しいことを直ちに確認するはできない．また，Enocoro-128v2 は，他の Enocoro と基本的な構造が似ているため，これらの安全性評価結果を基にしているところもある．

本報告では，自己評価書における安全性評価手法の中でも特に重要な差分確率と線形確率に基づく安全性評価を中心に行った．また，自己評価書では扱っていない cube 攻撃 [4] に対する安全性評価も行った．その結果，自己評価書の内容を否定するような結果は得られず，Enocoro-128v2 はこれらの攻撃に対して安全な疑似乱数生成器である．

第2章　Enocoro-128v2の仕様

ハードウエア向け疑似乱数生成器 Enocoro は，多くパラメータをもつ．パラメータの値を特定したものが Enocoro-128v2 である．本報告書では，Enocoro-128v2 を対象とする．この章では，Enocoro-128v2 のアルゴリズムの述べる．本報告書では，特に断りがない限り，この章で定義する記号を用いる (大半の記号の定義は仕様書に準じている)．

疑似乱数生成器は，有限状態機械，初期化関数 Init，出力関数 Out の三つからなる．

- 有限状態機械は，時刻 t に依存する内部状態を $S^{(t)}$ と更新関数 Next から成る．
- 初期化関数 Init は，鍵 K と初期値 I を入力とし，内部状態の初期値を $S^{(0)}$ を出力する関数である．
- 出力関数 Out は，時刻 t の内部状態 $S^{(t)}$ から時刻 t の出力 $z^{(t)}$ を定める関数である．

なお，他の省では，時刻 t をラウンド t と呼ぶ場合もある．PANAMA 型疑似乱数生成器では，内部状態 $S^{(t)}$ をステート $a^{(t)}$ とバッファ$b^{(b)}$ に分けるので，内部状態の更新関数 Next もステートの更新関数 ρ とバッファの更新関数 λ に分けられる．

$$S^{(0)} = (a^{(0)}, b^{(0)}) = \mathrm{Init}(K, I) \tag{2.1}$$

$$z^{(t)} = \mathrm{Out}(S^{(t)}) \tag{2.2}$$

$$S^{(t+1)} = (a^{(t+1)}, b^{(t+1)}) = \mathrm{Next}(S^{(t)}) = (\rho(a^{(t)}, b^{(t)}), \lambda(a^{(t)}, b^{(t)})) \tag{2.3}$$

Enocoro-128v2 は，PANAMA 型疑似乱数生成器である．128 ビット (16 バイト) 鍵 K と 64 ビット (8 バイト) の初期値 (IV)I を入力として，8 ビット (1 バイト) の出力 (鍵ストリーム)$z^{(t)}$ を次々と出力する．Enocoro-128v2 の内部状態は，2 バイトのステートと 32 バイトのバッファからなる．時刻 t のステートを上位バイトから $a_0^{(t)}, a_1^{(t)}$ とし，時刻 t のバッファを上位バイトから $b_0^{(t)}, b_1^{(t)}, \ldots, b_{31}^{(t)}$ とする．

まず，状態の更新関数 Next について述べる．ステートの更新関数 ρ は，Sbox による変換 s_8 と GF(2^8) 上の線形変換から成る．

$$u_0 = a_0^{(t)} \oplus s_8(b_2^{(t)}) \tag{2.4}$$

$$u_1 = a_1^{(t)} \oplus s_8(b_7^{(t)}) \tag{2.5}$$

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \begin{pmatrix} \mathtt{01} & \mathtt{01} \\ \mathtt{01} & \mathtt{02} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \quad \text{over} \quad \mathrm{GF}(2^8) \tag{2.6}$$

$$a_0^{(t+1)} = v_0 \oplus s_8(b_{16}^{(t)}) \tag{2.7}$$

$$a_1^{(t+1)} = v_1 \oplus s_8(b_{29}^{(t)}) \tag{2.8}$$

なお，GF(2^8) の定義多項式は，$\phi_8(x) = x^8 + x^4 + x^3 + x^2 + 1$ である．8 ビットの Sbox 変換 s_8 は，4 ビットの Sbox 変換 s_4 と GF(2^4) 上の線形変換と巡回ビットシフトから成る．入力の 8

表 2.1: 4 ビット Sbox 変換 s_4

x_0	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
y_0	1	3	9	a	5	e	7	2	d	0	c	f	4	8	6	b

ビット x を上位 4 ビット x_0，下位 4 ビット x_1 に分け，

$$y_0 = s_4(s_4(x_0) \oplus \mathtt{4} \cdot s_4(x_1) \oplus \mathtt{a}) \tag{2.9}$$

$$y_1 = s_4(\mathtt{4} \cdot s_4(x_0) \oplus s_4(x_1) \oplus \mathtt{5}) \tag{2.10}$$

$$y = (y_0 \| y_1) \lll 1 \tag{2.11}$$

により，8 ビットの出力 y を計算する．ここで，GF(2^4) 上の乗算の定義多項式は $\phi_4(x) = x^4+x+1$ であり，$\lll 1$ は 1 ビット左巡回シフトを表す．また，本報告書では，16 進数をタイプライタ体 (0 ∼ f) で表記する．例えば，4 は，左を上位ビットとして 2 進数で 0100 である．Sbox 変換 s_4 は，4 ビットの入力 x_0 を表 2.1 にしたがって 4 ビットの入力 y_0 に変換する．

バッファの更新関数 λ は，

$$b_i^{(t+1)} = \begin{cases} b_{31}^{(t)} \oplus a_0^{(t)} & \text{if } \ i = 0 \\ b_2^{(t)} \oplus b_6^{(t)} & \text{if } \ i = 3 \\ b_7^{(t)} \oplus b_{15}^{(t)} & \text{if } \ i = 8 \\ b_{16}^{(t)} \oplus b_{28}^{(t)} & \text{if } \ i = 17 \\ b_{i-1}^{(t)} & \text{otherwise.} \end{cases} \tag{2.12}$$

である．

出力関数 Out は，ステートの下位バイト $a_1^{(t)}$ を出力する．

$$z^{(t)} = \mathrm{Out}(S^{(t)}) = a_1^{(t)} \tag{2.13}$$

次に，初期化関数 Init について述べる．16 バイトの鍵 K を K_i $(i = 0, 1, \ldots, 15)$ と 8 バイトの初期値 I を I_i $(i = 0, 1, \ldots, 7)$ として，内部状態 (ステートとバッファ)$S^{(-96)}$ を以下のように定める．

$$a_0^{(-96)} = \mathtt{88}, \quad a_1^{(-96)} = \mathtt{4c}, \tag{2.14}$$

$$b_i^{(-96)} = \begin{cases} K_i & \text{if } \ i = 0, 1, \ldots, 15 \\ I_{i-16} & \text{if } \ i = 16, 17, \ldots, 23 \\ \mathtt{66} & \text{if } \ i = 24 \\ \mathtt{e9} & \text{if } \ i = 25 \\ \mathtt{4b} & \text{if } \ i = 26 \\ \mathtt{d4} & \text{if } \ i = 27 \\ \mathtt{ef} & \text{if } \ i = 28 \\ \mathtt{8a} & \text{if } \ i = 29 \\ \mathtt{2c} & \text{if } \ i = 30 \\ \mathtt{3b} & \text{if } \ i = 31 \end{cases} \tag{2.15}$$

更新関数 Next とカウンタ ctr を用いて，内部状態 $S^{(-96)}$ から内部状態の初期値 $S^{(0)}$ を下記の手順で生成する．

1. $ctr = \mathtt{02}$
2. $t = -96, -95, \ldots, -1$ に対して

$$b_{31}^{(t)} = ctr \oplus b_{31}^{(t)} \tag{2.16}$$

$$ctr = \mathtt{02} \cdot ctr \text{ over } \mathrm{GF}(2^8) \tag{2.17}$$

$$S^{(t+1)} = \mathrm{Next}(S^{(t)}) \tag{2.18}$$

第3章　Sboxの評価

本章では，Enocoro-128v2 の安全性の基礎となる Sbox の評価を行う．Sbox の評価は，後述の安全性評価の際にしばしば引用される．

3.1　4ビット Sbox 変換 s_4 の評価

4 ビット Sbox 変換 s_4 は表 2.1 で定義される 4 ビットの置換であり，8 ビット Sbox 変換 s_8 の構成部品である．s_4 は，Enocoro-128v2 の非線形変換の主要部分であり，s_4 の最大差分確率，最大線形確率，最大次数は，以下のとおりである．

最大差分確率 p は，次式で定義される値である．

$$p = \frac{1}{2^4} \max_{\Delta x \neq x, \Delta y} \#\left\{x \in \{0,1\}^4 | s_4(x) \oplus s_4(x \oplus \Delta x) = \Delta y\right\} \tag{3.1}$$

$$= \frac{1}{2^4} \max_{\Delta x \neq x, \Delta y} N_d(\Delta x, \Delta y) \tag{3.2}$$

$N_d(\Delta x, \Delta y)$ の分布を表 3.1 に示す．この表から最大差分確率 p は 2^{-2} である．

表 3.1: $N_d(\Delta x, \Delta y)$ の分布.

$N_d(\Delta x, \Delta y)$	0	2	4
$(\Delta x, \Delta y)$ の組数	138	84	18

最大線形確率 q は，次式で定義される値である．

$$q = \max_{\Gamma x, \Gamma y \neq 0} \left(\frac{2 \cdot \#\left\{x \in \{0,1\}^4 | x \bullet \Gamma x = s_4(x) \bullet \Gamma y\right\}}{2^4} - 1 \right)^2 \tag{3.3}$$

$$= \max_{\Gamma x, \Gamma y \neq 0} \left(\frac{2 \cdot N_l(\Gamma x, \Gamma y)}{2^4} - 1 \right)^2 \tag{3.4}$$

ここで，$\bullet$ は 4 ビットベクトルの GF(2) 上の内積である．$N_l(\Gamma x, \Gamma y)$ の分布を表 3.2 に示す．この表から最大線形確率 q は 2^{-2} である．

表 3.2: $N_l(\Gamma x, \Gamma y)$ の分布.

$N_l(\Gamma x, \Gamma y)$	4	6	8	10	12
$(\Gamma x, \Gamma y)$ の組数	15	60	96	52	17

また，s_4 は，定義多項式を $\phi_4(x) = x^4 + x + 1$ とした GF(2^4) 上の関数として下記のとおり表

表 3.3: 8 ビット Sbox 変換 s_8.

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

現することができる.

$$y = \sum_{i=0}^{14} c_i x^i \quad \text{over GF}(2^4) \tag{3.5}$$

$$= 1 + d \cdot x^1 + 6 \cdot x^2 + 7 \cdot x^3 + e \cdot x^4 + 5 \cdot x^5 + 1 \cdot x^6 + 6 \cdot x^7 + 3 \cdot x^8 + a \cdot x^9 + 9 \cdot x^{10} + e \cdot x^{11} + b \cdot x^{12} + 5 \cdot x^{13} + 2 \cdot x^{14} \quad \text{over GF}(2^4) \tag{3.6}$$

ここで，係数 $c_i \in \mathrm{GF}(2^4)$ は，$\mathrm{GF}(2^4)$ 上のラグランジュ補間で求めた.

s_4 の入力 4 ビットを上位ビット (左) から x_0, x_1, x_2, x_3，出力を上位ビット (左) から y_0, y_1, y_2, y_3 とおく．このとき，出力の 1 ビット y_i は，入力 4 ビット x_0, x_1, x_2, x_3 の $\mathrm{GF}(2)$ 上の関数 $s_{4,i}$ で表現できる．$s_{4,i}$ を求めると，下記のようになり，これらの式から代数次数は 3 であることがわかる.

$$s_{4,0}(x_0, x_1, x_2, x_3) = x0 + x2 + x0 \cdot x1 + x0 \cdot x2 + x0 \cdot x3 + x1 \cdot x2 + x1 \cdot x3 + x0 \cdot x1 \cdot x2 + x0 \cdot x1 \cdot x3 + x0 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot x3 \text{ over GF}(2) \tag{3.7}$$

$$s_{4,1}(x_0, x_1, x_2, x_3) = x0 + x1 + x0 \cdot x1 + x0 \cdot x3 + x0 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot x3 \text{ over GF}(2) \tag{3.8}$$

$$s_{4,2}(x_0, x_1, x_2, x_3) = x3 + x0 \cdot x3 + x1 \cdot x2 + x0 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot x3 \text{ over GF}(2) \tag{3.9}$$

$$s_{4,3}(x_0, x_1, x_2, x_3) = 1 + x0 \cdot x1 + x0 \cdot x2 + x0 \cdot x3 + x1 \cdot x3 + x2 \cdot x3 + x0 \cdot x1 \cdot x2 + x0 \cdot x2 \cdot x3 + x1 \cdot x2 \cdot x3 \text{ over GF}(2) \tag{3.10}$$

3.2 8 ビット Sbox 変換 s_8 の評価

8 ビット Sbox 変換 s_8 は，表 3.3 と等価である．表 3.3 は，8 ビットの値 wz が与えられたとき，$s_8(wz)$ の値を表している．自己評価書 (p.5) によれば，8 ビット Sbox 変換 s_8 は，

- 最大差分確率: $2^{-4.678}$
- 最大線形確率: 2^{-4}
- 代数次数: 6

と記されている．

最大差分確率 p は，次式で定義される値である．

$$p = \frac{1}{2^8} \max_{\Delta x \neq x, \Delta y} \#\left\{x \in \{0,1\}^8 | s_8(x) \oplus s_8(x \oplus \Delta x) = \Delta y\right\} \tag{3.11}$$

$$= \frac{1}{2^8} \max_{\Delta x \neq x, \Delta y} N_d(\Delta x, \Delta y) \tag{3.12}$$

実際，表 3.4 の 11 組の差分 $(\Delta x, \Delta y)$ で，

表 3.4: 最大差分確率 p となる差分 (0x は省略).

Δx	d3	a8	99	8d	80	76	6b	4e	2f	22	02
Δy	9a	a8	87	1e	69	61	69	10	eb	09	be

$$N_d(\Delta x, \Delta y) = \#\left\{x \in \{0,1\}^8 | s_8(x) \oplus s_8(x \oplus \Delta x) = \Delta y\right\} \tag{3.13}$$

$$= 10 \tag{3.14}$$

となり，最大差分確率が $10/256 \approx 2^{-4.678}$ であることを確認した．自己評価書で述べられているように，この最大差分確率は最良ではない．また，$N_d(\Delta x, \Delta y)$ の分布を表 3.5 に示す．

表 3.5: N_d の分布.

$N_d(\Delta x, \Delta y)$	0	1	2	3	4	5	6	7	8	9	10
Enocoro-128v2	39716	0	19571	0	5041	0	832	0	109	0	11

最大線形確率 q は，次式で定義される値である．

$$q = \max_{\Gamma x, \Gamma y \neq 0} \left(\frac{2 \cdot \#\left\{x \in \{0,1\}^8 | x \bullet \Gamma x = s_8(x) \bullet \Gamma y\right\}}{2^8} - 1 \right)^2 \tag{3.15}$$

$$= \max_{\Gamma x, \Gamma y \neq 0} \left(\frac{2 \cdot N_l(\Gamma x, \Gamma y)}{2^8} - 1 \right)^2 \tag{3.16}$$

ここで，$\bullet$ は 8 ビットベクトルの GF(2) 上の内積である．

表 3.6: 最大線形確率 q となるマスク値.

Γx	b5	62	f7	c1	9a	34	26
Γy	9d	60	cd	50	46	7f	06
$N_l(\Gamma x, \Gamma y)$	160		96				

実際，表 3.6 の 2 組のマスク値 $(\Gamma x, \Gamma y)$ で

$$N_l(\Gamma x, \Gamma y) = \#\left\{x \in \{0,1\}^8 | x \bullet \Gamma x = s_8(x) \bullet \Gamma y\right\} \tag{3.17}$$

$$= 160, \tag{3.18}$$

表 3.7: N_l の分布.

$N_l(\Gamma x, \Gamma y)$	160	156	152	148	144	140	136	132	128	124
$(\Gamma x, \Gamma y)$ の組数	2	25	154	561	1793	4182	7968	10987	13993	10995
$N_l(\Gamma x, \Gamma y)$	120	116	112	108	104	100	96			
$(\Gamma x, \Gamma y)$ の組数	7856	4187	1779	618	146	29	5			

5 組のマスク値 $(\Gamma x, \Gamma y)$ で，$N_l(\Gamma x, \Gamma y) = 96$ となることを確認した．これらの値が最大線形確率 $q = 2^{-4}$ を与える．また，$N_l(\Gamma x, \Gamma y)$ の分布を表 3.7 に示す．

また，s_8 は，定義多項式を $\phi_8(x) = x^8 + x^4 + x^3 + x^2 + 1$ とした $\mathrm{GF}(2^8)$ 上の関数として表現することができる．

$$y = \sum_{i=0}^{254} c_i x^i \quad \text{over GF}(2^8) \tag{3.19}$$

上式の係数 $c_i \in \mathrm{GF}(2^8)$ は，$\mathrm{GF}(2^8)$ 上のラグランジュ補間で求められ，表 3.8 の値になる．例えば，定数項 63 は，$\mathrm{GF}(2^8)$ の要素を多項式で表現したとき，$x^7 + x^5 + x + 1$ を意味する．表 3.8 から，s_8 は，252 次の多項式であることがわかる．

s_8 の入力 8 ビットを上位ビット (左) から $x_0, x_1, \ldots, x_7$，出力を上位ビット (左) から $y_0, y_1, \ldots, y_7$ とおく．このとき，出力の 1 ビット y_i は，入力 8 ビット $x_0, x_1, \ldots, x_7$ のブール関数 $s_{8,i}$ となる．

$$y_0 = \sum (\text{表 3.9 の項}) \text{ over GF}(2), \quad y_1 = 1 + \sum (\text{表 3.10 の項}) \text{ over GF}(2),$$
$$y_2 = 1 + \sum (\text{表 3.11 の項}) \text{ over GF}(2), \quad y_3 = \sum (\text{表 3.12 の項}) \text{ over GF}(2),$$
$$y_4 = \sum (\text{表 3.13 の項}) \text{ over GF}(2), \quad y_5 = \sum (\text{表 3.14 の項}) \text{ over GF}(2)$$
$$y_6 = 1 + \sum (\text{表 3.15 の項}) \text{ over GF}(2), \quad y_7 = 1 + \sum (\text{表 3.16 の項}) \text{ over GF}(2)$$

表 3.9 から表 3.16 は，各ブール関数 $s_{8,i}$ に含まれる x_i の項を表す．例えば，表 3.9 において，1 行 2 列に 'o' は，x_1 の項が $s_{8,0}$ に含まれることを表す．また，4 行めは 1 列めと 4 列めに 'o' があるので，$x_0 \cdot x_3$ の項が $s_{8,0}$ に含まれることを表す．逆に，3 列めだけに 'o' がある行はないので，x_2 の項は $s_{8,0}$ に含まれない．表 3.9 から表 3.16 の最大次数が 6 であるから，代数次数が 6 である．代数次数が最大で 6 次になりうることは，s_4 の代数次数が 3 であり，s_8 は s_4 を二回使用していることから，直ちに確認できる．

自己評価書に述べられているように，8 ビット Sbox s_8 は，最大差分確率，最大線形確率，代数次数の点で最良ではない．これは，ハードウエア実装の効率を重視し，4 ビット Sbox s_4 を元に作られているためである．

表 3.8: GF(2^8) 上の多項式の係数.

i	c_i	i	c_i	i	c_i	i	c_i	i	c_i	i	c_i	i	c_i
0	63	40	bb	80	ae	120	fb	160	8e	200	8a	240	6a
1	f3	41	45	81	03	121	8b	161	68	201	a7	241	ae
2	fa	42	d1	82	07	122	13	162	57	202	c2	242	3e
3	ca	43	81	83	ba	123	31	163	d6	203	fd	243	fc
4	f7	44	0b	84	b5	124	6e	164	20	204	8b	244	f3
5	d9	45	4d	85	e6	125	96	165	a0	205	95	245	0f
6	16	46	13	86	12	126	c9	166	53	206	7c	246	47
7	cc	47	30	87	63	127	00	167	ac	207	c4	247	00
8	4b	48	af	88	95	128	35	168	36	208	f2	248	8a
9	9d	49	ed	89	64	129	68	169	da	209	a8	249	cf
10	b9	50	c6	90	0a	130	c4	170	6a	210	c0	250	47
11	1c	51	a1	91	93	131	19	171	ea	211	a5	251	00
12	c1	52	f1	92	59	132	e9	172	2d	212	87	252	03
13	2a	53	b0	93	50	133	53	173	eb	213	68	253	00
14	a7	54	36	94	71	134	db	174	83	214	a8	254	00
15	29	55	0a	95	f6	135	64	175	06	215	13		
16	c6	56	13	96	c2	136	98	176	bc	216	a7		
17	f8	57	f1	97	c6	137	c0	177	7b	217	82		
18	0f	58	c6	98	73	138	c5	178	14	218	e1		
19	cd	59	ff	99	d7	139	7a	179	4c	219	e4		
20	28	60	99	100	ec	140	39	180	4f	220	ef		
21	05	61	63	101	bc	141	7d	181	8d	221	74		
22	5f	62	5d	102	f0	142	a0	182	6a	222	79		
23	d6	63	95	103	66	143	57	183	7b	223	00		
24	b5	64	ad	104	c4	144	e5	184	1b	224	3b		
25	d3	65	13	105	f7	145	ce	185	47	225	9c		
26	b8	66	6d	106	8f	146	b4	186	aa	226	b6		
27	a4	67	b9	107	33	147	2a	187	95	227	61		
28	ae	68	45	108	9c	148	0e	188	64	228	d0		
29	1f	69	81	109	8d	149	77	189	1f	229	a5		
30	40	70	e3	110	0d	150	7d	190	2e	230	cd		
31	b7	71	ad	111	43	151	b8	191	00	231	5e		
32	22	72	ce	112	dd	152	fa	192	20	232	eb		
33	70	73	96	113	15	153	9e	193	51	233	b0		
34	ea	74	82	114	60	154	2a	194	ff	234	52		
35	ab	75	cc	115	19	155	17	195	b3	235	f9		
36	97	76	79	116	5a	156	59	196	88	236	59		
37	1d	77	ee	117	c2	157	35	197	25	237	3d		
38	c5	78	e4	118	a3	158	4a	198	3c	238	e1		
39	47	79	b7	119	71	159	90	199	49	239	00		

表 3.9: y_0 のブール関数の項.

0	1	2	3	4	5	6	7
	o						
				o			
					o		
o			o				
o					o		
o						o	
o							o
	o	o					
	o		o				
	o			o			
	o				o		
	o					o	
	o						o
		o	o				
		o		o			
			o			o	
			o				o
				o	o		
	o	o	o				
o	o			o			
o	o						o
o			o	o			
o			o		o		
o				o	o		
o						o	o
	o	o	o				
	o	o			o		
	o	o				o	
	o	o					o
	o		o	o			
	o		o			o	
	o		o				o
	o			o	o		
	o			o		o	

0	1	2	3	4	5	6	7
		o	o	o			
		o	o				o
		o		o		o	
		o				o	o
			o	o		o	
			o		o	o	
			o			o	o
			o		o		o
			o			o	o
				o	o	o	
				o	o		o
o	o	o				o	
o	o				o		
o	o					o	o
o			o	o		o	
o			o		o		o
o				o	o		o
o						o	o
	o	o	o				
	o	o			o		
	o	o				o	
	o		o	o			
	o		o			o	
	o			o	o		
	o					o	o
		o	o		o		
		o		o		o	
		o			o		o
			o		o		o
				o	o		o
	o	o	o				
	o	o			o		
	o	o				o	
	o		o	o			
	o		o			o	
	o			o	o		
	o					o	o

0	1	2	3	4	5	6	7
		o	o		o		o
		o		o	o	o	
			o	o	o	o	
			o		o	o	o
o	o	o		o	o		
o	o	o			o		o
o	o	o				o	o
o	o		o	o		o	
o	o		o		o		
o	o			o	o		
o		o	o	o	o		
	o	o	o	o			
	o	o	o	o		o	
	o	o		o			o
	o		o	o	o	o	
	o		o	o	o		o
	o			o	o	o	
	o				o	o	o
		o	o	o		o	
		o		o		o	o
		o			o	o	o
			o	o	o		o
	o	o	o	o		o	
	o	o		o	o		
	o	o		o		o	o
	o		o	o	o	o	
	o		o	o	o		o
	o	o	o	o	o	o	
	o	o	o	o		o	o
	o	o	o		o	o	o

表 3.10: y_1 のブール関数の項.

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
			o						o			o	o				o			o		o	o
				o					o			o			o		o				o	o	o
					o				o				o		o			o	o		o		o
						o				o	o	o						o		o	o	o	
o	o									o	o				o			o		o	o		o
o		o								o		o		o					o	o	o	o	
o			o							o			o	o					o	o	o		o
o				o							o	o		o					o		o	o	o
o						o					o		o		o	o	o	o		o	o		
	o	o										o		o	o	o	o	o		o			o
	o			o									o	o	o	o	o		o	o		o	
		o				o		o	o	o					o	o	o		o	o			o
			o	o				o	o		o	o				o	o		o		o	o	
			o				o	o	o		o				o	o	o		o			o	o
				o	o			o	o			o		o		o	o			o	o	o	
				o			o	o	o			o			o	o	o			o		o	o
					o	o		o	o				o	o		o		o	o	o	o		
					o		o	o	o					o	o	o		o	o	o			o
						o	o	o		o	o	o				o		o	o		o	o	
o	o		o					o		o	o			o		o		o	o		o		o
o	o				o			o		o		o	o			o		o		o	o		o
o		o	o					o		o				o	o	o		o			o	o	o
o		o		o				o			o	o		o		o			o	o	o	o	
o		o				o		o			o	o			o	o			o	o		o	o
o			o	o				o			o			o	o	o			o		o	o	o
o			o			o		o				o	o	o			o	o	o		o	o	
o			o				o	o				o	o		o		o	o	o		o		o
o				o	o			o				o		o	o		o	o		o		o	o
o				o		o		o					o	o	o		o		o	o		o	o
o				o			o		o	o	o		o					o	o		o	o	o
o					o	o			o	o	o			o		o	o	o		o		o	o
o					o		o		o	o		o	o			o	o		o	o	o	o	
	o	o	o						o	o		o			o	o	o		o	o		o	o
	o	o			o				o	o			o	o		o	o		o		o	o	o
	o	o				o			o	o			o		o	o		o	o	o		o	o
	o	o					o		o		o	o	o			o		o	o		o	o	o
	o		o				o		o		o	o			o								

表 3.11: y_2 のブール関数の項.

0	1	2	3	4	5	6	7
o							
		o					
			o				
				o			
					o		
						o	
							o
o			o				
o						o	
	o		o				
	o				o		
			o		o		
			o				o
				o		o	
						o	o
o		o			o		
o			o			o	
o			o				o
o				o	o		
o				o		o	
	o	o		o			
	o		o			o	
	o				o	o	
	o				o		o
	o					o	o
		o	o	o			
		o	o		o		
		o	o			o	
		o	o				o
		o		o	o		
		o				o	o
			o	o	o		
			o	o		o	
			o	o			o
			o		o		o
				o		o	o
					o	o	o

0	1	2	3	4	5	6	7
o	o	o		o			
o	o	o			o		
o	o	o				o	
o	o	o					o
o		o	o			o	
o		o		o			o
o			o	o	o	o	
o			o		o		o
	o	o	o	o	o		
	o			o	o	o	
	o			o	o		o
	o			o		o	
	o			o			o
	o			o		o	
		o	o	o	o	o	
		o	o	o	o		o
		o	o		o	o	o
o	o	o		o			o
o	o	o			o	o	o
o	o		o	o		o	o
o	o		o		o	o	o
o		o	o	o		o	o
	o	o	o	o		o	o

0	1	2	3	4	5	6	7
o	o		o			o	o
o		o	o	o		o	
o		o	o		o	o	
o		o	o		o		o
o		o	o			o	o
o		o		o		o	o
o			o	o	o	o	
o			o		o	o	o
	o	o	o	o	o		
	o	o	o	o		o	
	o	o	o		o		o
	o	o	o			o	o
	o	o		o	o		o
	o	o		o		o	o
		o	o	o	o	o	
		o	o	o	o		o
		o	o		o	o	o
o	o	o		o	o		o
o	o	o			o	o	o
o	o		o	o		o	o
o	o		o		o	o	o
o		o	o	o		o	o
	o	o	o	o		o	o

表 3.12: y_3 のブール関数の項.

0	1	2	3	4	5	6	7
o							
	o						
			o				
						o	
							o
o		o					
o				o			
o					o		
o						o	
	o				o		
	o					o	
		o		o			
		o			o		
		o				o	
		o					o
			o	o			
			o		o		
			o				o
				o	o		
				o		o	
					o	o	
o	o				o		
o		o		o			
o		o			o		
o		o				o	
o		o					o
o			o	o			
o				o		o	
o						o	o
	o	o				o	
	o	o					o
	o		o		o		
	o		o				o
	o			o	o		
	o			o		o	
	o				o		o

0	1	2	3	4	5	6	7
		o	o	o			
		o	o		o		
		o	o				o
		o			o		o
			o	o	o		
			o	o		o	
			o		o	o	
				o	o		o
				o	o	o	
					o	o	o
o	o	o		o			
o	o	o			o		
o	o	o				o	
o	o	o					o
o	o		o	o	o		
o	o		o		o	o	
o	o		o			o	o
o	o			o	o		
o	o				o	o	o
o		o	o	o		o	
o		o	o		o		o
o		o	o			o	o
o		o		o	o		o
o			o	o	o		
	o	o	o	o	o		
	o	o	o		o		o
	o	o	o			o	o
	o	o		o	o	o	
	o	o			o		o
	o		o	o	o		
	o		o			o	o
		o		o			
		o		o		o	
		o			o	o	

0	1	2	3	4	5	6	7
	o			o	o	o	
	o			o	o		o
	o				o	o	o
		o	o	o			o
		o	o		o	o	
		o		o	o	o	
		o		o	o		o
			o	o	o	o	
			o		o	o	o
o	o	o		o		o	
o	o	o		o			o
o	o	o			o	o	
o	o	o				o	o
o	o		o	o	o		
o	o		o		o	o	
o	o			o	o	o	
o	o				o	o	o
o		o	o	o		o	
o		o	o		o		o
o		o	o			o	o
o		o		o	o	o	
o		o		o	o		o
	o	o	o	o	o	o	
	o	o	o		o		o
		o	o	o	o	o	
o	o	o		o	o	o	
o	o	o		o	o		o
o	o		o	o	o	o	
o	o		o		o		o
	o	o	o	o	o	o	

表 3.13: y_4 のブール関数の項.

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
o									o			o			o			o	o		o	o	
		o							o				o	o				o	o			o	o
			o							o	o	o						o		o	o	o	
				o						o	o			o						o	o	o	o
					o					o		o	o							o	o		o	o
						o				o			o	o						o		o	o	o
o	o									o			o		o	o	o	o		o		o	
o			o							o				o	o	o	o	o			o	o	
o							o				o	o		o			o	o			o		o
	o		o									o	o	o		o	o		o	o	o		
	o			o								o	o		o	o	o		o	o			o
	o				o			o	o	o					o	o	o		o		o		o
		o	o					o	o		o	o				o	o		o			o	o
		o		o				o	o		o			o		o	o			o	o		o
		o				o		o	o		o				o	o	o			o		o	o
			o	o				o		o	o		o			o		o	o	o	o		
			o		o			o		o	o			o			o	o	o		o	o	
				o	o			o		o	o				o	o		o	o		o		o
				o			o	o		o		o		o		o		o	o			o	o
					o	o		o		o			o	o		o		o		o	o	o	
					o		o	o		o			o		o			o		o	o	o	o
								o			o	o	o			o		o			o	o	o
o	o	o						o			o	o	o			o		o			o	o	o
o	o		o					o			o		o	o		o			o	o	o	o	
o	o			o				o			o		o		o	o			o		o	o	o
o	o					o		o			o			o	o		o	o	o	o	o	o	
o		o	o					o				o	o	o			o	o		o	o		o
o		o					o	o				o	o		o		o	o		o		o	o
o			o		o			o				o		o			o	o			o	o	o
o			o			o			o	o	o		o				o		o	o	o	o	
o			o				o		o	o	o			o			o		o	o	o		o
o				o			o		o		o	o	o			o	o	o		o	o	o	
o					o		o		o		o	o		o			o	o		o		o	o
	o	o	o						o		o	o			o	o	o	o			o	o	o
	o	o			o				o		o		o		o	o	o		o	o	o	o	
	o	o				o			o		o			o	o	o	o		o	o		o	o
	o	o					o		o		o			o	o	o	o		o		o	o	o
	o		o							o	o	o	o			o		o	o	o	o	o	
	o		o			o				o	o	o		o		o		o	o		o	o	o
	o		o				o			o	o	o			o		o	o		o		o	o
	o		o							o	o	o			o	o	o	o		o	o	o	o

表 3.14: y_5 のブール関数の項.

0	1	2	3	4	5	6	7
o							
	o						
		o					
o	o						
o		o					
o			o				
o				o			
o						o	
	o	o					
	o		o				
		o	o				
		o		o			
		o			o		
		o				o	
				o	o		
				o			o
					o	o	
					o		o
o	o	o					
o	o		o				
o	o			o			
o	o					o	
o		o	o				
o		o		o			
o		o			o		
o		o					o
o			o	o			
o			o			o	
o			o				o
	o	o	o				
	o	o		o			
	o	o			o		
	o		o	o			
	o		o			o	
	o		o				o
	o			o	o		
	o				o		
	o					o	

0	1	2	3	4	5	6	7
		o	o				o
		o		o	o		
		o		o		o	
		o		o			o
		o			o	o	
		o				o	o
			o	o	o		
			o	o		o	
			o	o			o
				o	o	o	
				o		o	o
				o	o		o
o	o			o		o	
o	o			o	o		
o	o			o			o
o		o	o				
o		o		o			
o		o			o		
o		o					o
o			o	o			
o			o			o	
o			o				o
	o	o	o				
	o	o		o			
	o	o			o		
	o		o	o			
	o		o			o	
	o		o				o
	o			o	o		
	o				o		
	o					o	

0	1	2	3	4	5	6	7
		o	o		o		o
		o	o			o	o
		o		o	o		o
		o		o		o	o
		o			o	o	o
			o	o	o		o
o	o	o		o	o		
o	o	o		o		o	
o	o	o		o			o
o	o	o			o	o	
o	o	o			o		o
o	o		o	o	o		
o	o		o			o	
o		o	o	o			o
o		o	o			o	o
o		o		o			
o		o			o		
o		o				o	
o		o					o
	o	o	o	o	o		
	o	o	o	o		o	
	o	o	o	o			o
	o	o	o		o	o	
	o	o		o		o	
	o		o	o	o		
	o		o			o	
	o			o	o	o	
	o				o		
	o					o	

表 3.15: y_6 のブール関数の項.

0	1	2	3	4	5	6	7
o							
	o						
				o			
o	o						
o		o					
o			o				
o				o			
o					o		
	o	o					
	o					o	
	o						o
		o	o				
		o				o	
			o	o			
			o		o		
			o				o
				o			o
o	o	o					
o	o			o			
o	o				o		
o	o						o
o		o	o				
o		o		o			
o		o					o
o			o	o			
o			o		o		
o			o			o	
o			o				o
o				o		o	
o					o	o	
o					o		o
	o	o				o	
	o		o		o		
	o		o				o
	o				o	o	
	o				o		o
	o					o	o
		o	o	o			

0	1	2	3	4	5	6	7
		o	o		o		
		o		o			o
		o			o	o	
		o			o		o
			o	o		o	
			o	o			o
			o		o		o
				o	o		o
					o	o	o
				o		o	o
					o	o	o
o		o	o	o			
o		o	o				o
o		o	o			o	
o		o		o			o
o		o			o	o	
o			o			o	o
o	o	o	o	o			
o		o	o		o		
o		o			o	o	
o		o			o		o
o		o				o	o
o			o	o			
o			o			o	o
o			o				o
o				o		o	o
o					o	o	o
	o	o	o				o
	o			o	o		o
	o				o	o	o
	o					o	o
	o				o		o
		o	o	o			o

0	1	2	3	4	5	6	7
		o		o		o	o
		o			o	o	o
			o	o		o	o
o	o	o		o			o
o	o	o			o		o
o	o	o				o	o
o	o		o		o		o
o	o			o	o	o	
o	o				o	o	o
o		o	o	o		o	
o		o	o		o		o
o		o	o			o	o
o		o		o		o	o
o		o			o	o	o
	o	o	o	o			o
	o	o	o		o	o	
	o	o		o	o		o
	o	o		o		o	o
	o		o	o	o	o	
	o		o	o		o	o
	o		o		o		o
	o		o			o	o
		o	o	o		o	o
		o	o		o	o	o
o	o		o	o	o	o	
o		o	o	o		o	o
o		o	o		o		o
o		o	o	o		o	
	o	o	o	o	o	o	

表 3.16: y_7 のブール関数の項.

0	1	2	3	4	5	6	7
		o					
					o		
						o	
							o
o		o					
o			o				
o				o			
o						o	
o							o
	o		o				
	o			o			
	o				o		
	o					o	
	o						o
		o		o			
		o			o		
		o				o	
			o			o	
			o				o
				o	o		
				o			o
					o	o	
					o		o
o	o		o				
o	o				o		
o	o						o
o		o		o			
o		o			o		
o			o				o
o				o		o	
o				o			o
o						o	o
	o	o	o				
	o		o		o		
	o		o			o	
	o			o	o		
	o			o		o	
	o						o

0	1	2	3	4	5	6	7
	o					o	o
		o	o	o			
		o		o		o	
		o			o	o	
			o	o		o	
			o		o	o	
			o			o	o
				o	o	o	
				o		o	o
				o	o		o
					o	o	o
o		o					
o			o				
o				o			
o					o		
o						o	
o							o
	o		o				
	o			o			
	o				o		
	o					o	
	o						o
		o		o			
		o			o		
		o				o	
			o			o	
			o				o
				o	o		
				o		o	
				o			o
					o	o	
					o		o
o	o		o				
o	o				o		
o	o						o
o		o		o			
o		o			o		
o			o				o
o				o		o	
o				o			o
o						o	o
	o	o	o				
	o		o		o		
	o		o			o	
	o			o	o		
	o			o		o	
	o				o		o

0	1	2	3	4	5	6	7
	o	o				o	o
	o		o	o		o	
	o		o	o			o
	o		o		o		o
	o			o	o	o	
		o	o	o			o
		o	o		o	o	
		o	o		o		o
		o		o	o	o	
		o		o	o		o
			o	o		o	o
o	o	o		o	o		
o	o	o				o	
o	o	o			o	o	
o	o	o					o
o		o		o	o	o	
o	o		o		o		o
o	o		o			o	o
o	o			o	o	o	
o	o				o	o	o
	o	o	o	o	o	o	
	o	o	o		o		o
	o	o	o			o	o
	o	o		o	o	o	
	o	o	o		o	o	o
		o	o	o	o	o	
		o	o		o	o	o
		o	o		o		o
		o		o	o	o	
		o		o	o		o
		o			o	o	o
			o	o	o	o	
			o		o		o
			o			o	o
				o	o	o	o
				o		o	
					o		o

第4章　差分確率に基づく攻撃

4.1　疑似乱数生成器に対する差分確率

8 ビット変換に対する最大差分確率を式 (3.12) で定義した．Enocoro-128v2 の状態更新関数 Next は，34 バイト入力 $S^{(t-1)}$，34 バイト出力 $S^{(t)}$ の関数である．その最大差分確率を

$$DP_{\mathrm{Next}}(\Delta S^{(t-1)}, \Delta S^{(t)}) = \frac{\#\left\{S^{(t-1)} \in \{0,1\}^{272} | \mathrm{Next}(S^{(t-1)}) \oplus \mathrm{Next}(S^{(t-1)} \oplus \Delta S^{(t-1)}) = \Delta S^{(t)}\right\}}{2^{272}} \tag{4.1}$$

と定義する．$\Delta S^{(t-1)}$ を入力差分，$\Delta S^{(t)}$ を出力差分と呼ぶ．時刻 $t = 0$ の内部状態 $S^{(0)}$ を入力とし，時刻 $t = R$ の内部状態 $S^{(R)}$ を出力とする関数を考え，その関数に対する最大差分確率を同様に定義することができる．その最大差分確率の近似値として各時刻での最大差分確率の積を考え，最大差分特性確率 $DCP(R)$ と呼ぶ．

$$DCP(R) = \max_{\Delta S^{(t-1)}} \prod_{t=1}^{R} DP_{\mathrm{Next}}(\Delta S^{(t-1)}, \Delta S^{(t)}) \tag{4.2}$$

ここで，$(\Delta S^{(0)}, \Delta S^{(1)}, \ldots, \Delta S^{(R)})$ を差分パスという．

$DP_{\mathrm{Next}}(\Delta S^{(t-1)}, \Delta S^{(t)})$ を求めることが計算量的に難しいため，$DCP(R)$ を求めることも難しい．そこで，差分 $\Delta S^{(t)}$ を 8 ビット毎にわけ，その 8 ビットが非ゼロ (active) か否 (non-active) かだけを評価することを考える (truncate 差分解析)．この場合，272 ビットの内部状態の差分 $\Delta S^{(t)}$ は，34 ビットの差分 $\Delta T^{(t)}$ に縮約される．差分を 8 ビット毎に分けた理由は，Enocoro-128v2 の処理が 8 ビット単位であるからである．Sbox の入力差分が非ゼロ (active) であるとき，その Sbox を active Sbox という．状態更新関数 Next の非線形関数は s_8 のみであるから，s_8 の最大差分確率を p，ラウンド R までの active Sbox の数の最小値を $AS^{(R-1)}$ とおくと，

$$DCP(R) \leq p^{AS^{(R)}} \tag{4.3}$$

となる．Enocoro-128v2 の場合，$p = 2^{-4.678}$ なので，$AS^{(R)} \geq 28$ となると差分確率に基づく攻撃に対して安全である．

4.2　自己評価による評価

自己評価書によれば，攻撃に必要な既知 IV 数は約 $2^{140.3}$ 個である．これは，Enocoro-128v2 の IV の総数が 2^{64} であることから，差分確率に基づく攻撃は不可能であることを意味する．

また，s_8 の最大差分確率が $2^{-4.678}$ なので，$t = -96$ から $t = 15$ までの間に，active Sbox が 30 個の truncate 差分パスが存在することを意味する．active Sbox の数は，計算機実験により得られた値であり，truncate 差分パスは自己評価書や参考文献では示されていない．

4.3 active Sbox 数が最小となる差分パスの探索

差分パスの探索開始は，鍵と IV を代入した時点 $t = -96$ から始める．このとき，$b_0^{(-96)}$, $b_2^{(-96)}$, $\ldots$, $b_{31}^{(-96)}$, $a_0^{(-96)}$, $a_1^{(-96)}$ の truncate 差分を

$$\Delta T^{(-96)} = (\Delta_0^{(-96)}, \Delta_1^{(-96)}, \ldots, \Delta_{33}^{(-96)}) \tag{4.4}$$

とおく．$\Delta T^{(-96)}$ は，二通りの場合を考える．

初期差分条件 1 $\Delta_0^{(-96)}$, $\Delta_1^{(-96)}$, $\ldots$, $b_{19}^{(-96)}$ のうち少なくとも一つは 1，$\Delta_{20}^{(-96)}$, $\Delta_{21}^{(-96)}$,$\ldots$, $\Delta_{33}^{(-96)}$ はすべて 0 とする．

初期差分条件 2 $\Delta_0^{(-96)}, \Delta_1^{(-96)}, \ldots, \Delta_{33}^{(-96)}$ のうち，少なくとも一つは 1 とする．

初期差分条件 1 は，攻撃者が鍵と IV に差分を任意に設定できる状況に対応する．初期差分条件 2 は，攻撃者が仕様では固定値となっている部分にも差分を任意に設定できる状況に対応する．

状態 $S^{(-96)}$ から状態 $S^{(-95)}$ へ遷移する過程において，

- 差分が分岐点を通過するとき: 分岐では，入力された値がそれぞれにそのまま出力されるので，入力された差分もそのまま出力される．truncate 差分についても，入力された truncate 差分が 1 ならば，出力される truncate 差分は 1，入力された truncate 差分が 0 ならば，出力される truncate 差分は 0 である．

- 差分が XOR 演算点を通過するとき: XOR 演算点の二つ入力を x, y，出力を z とおく．x が差分 (truncate 差分ではない)δ_x，y が差分 δ_y をもつとき，出力 z の差分 δ_z は $\delta_x \oplus \delta_y$ となる．$\delta_x, \delta_y, \delta_z$ の truncate 差分を $\Delta_x, \Delta_y, \Delta_z$ とおくと，

$$\Delta_z = \begin{cases} 0 & \text{if } \Delta_x = \Delta_y = 0 \\ 1 & \text{if } \Delta_x = 0, \Delta_y = 1 \\ 1 & \text{if } \Delta_x = 1, \Delta_y = 0 \\ 0, 1 & \text{if } \Delta_x = \Delta_y = 1 \end{cases} \tag{4.5}$$

  となる．$\Delta_x = 1, \Delta_y = 1$ のときには，出力差分が 0,1 のどちらにもなり得るので，攻撃者は都合のよい方 (つまり，active Sbox の総数が最小になるような) を選ぶことができる．

- 差分が線形変換 L を通過するとき: 線形変換 L は，式 (2.6) で与えられ，分岐数が 3 である．つまり，L に $u_0 \oplus \delta_0, u_1 \oplus \delta_1$ が入力されると，

$$\begin{pmatrix} \mathtt{01} & \mathtt{01} \\ \mathtt{01} & \mathtt{02} \end{pmatrix} \begin{pmatrix} u_0 \oplus \delta_0 \\ u_1 \oplus \delta_1 \end{pmatrix} = \begin{pmatrix} u_0 \oplus u_1 \oplus \delta_0 \oplus \delta_1 \\ u_0 \oplus \mathtt{02} \cdot u_1 \oplus \delta_0 \oplus \mathtt{02} \cdot \delta_1 \end{pmatrix} \tag{4.6}$$

$$= \begin{pmatrix} v_0 \oplus \delta_0 \oplus \delta_1 \\ v_1 \oplus \delta_0 \oplus \mathtt{02} \cdot \delta_1 \end{pmatrix} \tag{4.7}$$

  となるので，u_0, u_1, v_0, v_1 の truncate 差分を Δ_{u_0}, Δ_{u_1}, Δ_{v_0}, Δ_{v_1} とおくと，

  - $\Delta_{u_0} = \Delta_{u_1} = 0$ のとき，$\Delta_{v_0} = \Delta_{v_1} = 0$.
  - $\Delta_{u_0} = 0, \Delta_{u_1} = 1$ のとき，$\Delta_{v_0} = 1, \Delta_{v_1} = 1$.
  - $\Delta_{u_0} = 1, \Delta_{u_1} = 0$ のとき，$\Delta_{v_0} = 1, \Delta_{v_1} = 1$.

– $\Delta_{u_0} = 1, \Delta_{u_1} = 1$ のとき，$\Delta_{v_0} = 0, \Delta_{v_1} = 1$，または $\Delta_{v_0} = 1, \Delta_{v_1} = 0$，または $\Delta_{v_0} = 1, \Delta_{v_1} = 1$.

となる．$\Delta_{u_0} = 1, \Delta_{u_1} = 1$ のとき，攻撃者は，都合のよい出力差分を選ぶことができる．

このように，XOR 演算点と線形変換部において，truncate 差分の通過の仕方が一意ではないため，状態 $S^{(-95)}$ の truncate 差分

$$\Delta T^{(-95)} = (\Delta_0^{(-95)}, \Delta_1^{(-95)}, \ldots, \Delta_{33}^{(-95)}) \tag{4.8}$$

は一意ではなく，それぞれの trucate 差分に対する active Sbox の数も異なる．

状態 $S^{(-95)}$ から状態 $S^{(-94)}$，状態 $S^{(-94)}$ から状態 $S^{(-93)}$，などについても同様に考え，状態 $S^{(-96)}$ の truncate 差分 $\Delta T^{(-96)}$ から状態 $S^{(t)}$ の差分 $\Delta T^{(t)}$ になるために必要な active Sbox の数の最小値を求める (この探索には，Viterbi アルゴリズムが有効である)．

Enocoro-128v2 は，鍵長が 128 ビットなので，各差分の出現確率が 2^{-128} からの偏差を調べる必要がある．したがって，128 ビット以上の出力が必要なので，探索を 16 ラウンド以上繰り返す必要がある．しかし，ラウンド数に対して，active Sbox の数の最小値は非減少関数なので，$\Delta T^{(15)}$ を求めればよい．

$AS^{(t)}$ は，差分の初期状態から $t + 96$ ラウンド進んだとき，truncate 差分が通過する Sbox の最小個数である．truncate 差分では，8 ビット単位の差分が非ゼロか否かだけに着目しているので，truncate 差分パスがつながっているとき，1 ビット単位の差分パスがつながっているとは限らない．逆に，1 ビット単位の差分パスがつながっているときは，必ず truncate 差分パスはつながっている．したがって，$AS^{(t)}$ による最大差分特性確率で，$DCP(R) > 2^{-128}$ であることは，1 ビット単位の差分確率に基づく攻撃に対して脆弱であるとは限らないが，$DCP(R) < 2^{-128}$ であれば，1 ビット単位の差分確率に基づく攻撃に対して安全である．

4.4 評価結果

前節で説明した探索法により求めた $AS^{(t)}$ を表 4.1，表 4.2 に示す．計算機資源の制約のため，具体的な差分パスを求めることはできなかった．表 4.1 の結果より

$$DCP(16) \leq p^{AS^{(16)}} = 2^{-4.678*58} = 2^{-271.324} < 2^{-128} \tag{4.9}$$

である．攻撃者が固定値の部分に差分を与えられるとしても，表 4.2 から，$DCP(16) \leq 2^{-257.290}$ である．したがって，差分確率を利用して，Enocoro-128v2 の出力系列と真性乱数系列を鍵の総当たり攻撃より効率良く識別することはできない．

自己評価書では，$DCP(16) \leq 2^{-140.3}$ と述べられている．これは，最小 active Sbox の数が 30 個の差分パスが存在することを意味する．上記の結果との相違の原因は，文献 [8] で述べられている，カウントしない active Sbox を考慮しているからである．文献 [8] では，

1. 鍵が Sbox に入力される場合
2. IV が Sbox に入力される場合
3. 鍵と IV の XOR が初めて Sbox に入力される場合

には，active Sbox にカウントしないことを提案している．1 の場合にはその Sbox の出力を等価鍵とみなすこと，2 の場合には IV の値が既知であるため Sbox の出力値も既知になることがカウントしない理由であるから，カウントしないことによる計算量の増加はない．一方，3 の場合には，その Sbox の入力を総当たりすることを想定しているので，active Sbox にカウントしない場合には計算量が余分に必要になる．

Enocoro-128v2 の状態遷移を考えると，$S^{(-66)}$ になるまで，1,2,3 のいずかが起こり得る．したがって，1,2,3 が起きた場合に active Sbox にカウントしないことは，初期化のラウンド回数が最大 30 回少なくなることに相当する．また，このときは，ΔT^{-66} はすべての部分に差分を与えることができることになる．

初期化のラウンド回数を 30 回少なくした場合の active Sbox の最小個数は，表 4.2 の $AS^{(-14)} = 39$ に対応する．これは攻撃者に有利な見積りになっているが，それでも 8 ビット truncate 差分に基づく最大差分特性確率は 2^{-128} を下回っているため，Enocoro-128v2 は差分確率に基づく攻撃に対して安全である．今回の実験では，自己評価書で述べられている $DCP(16) \leq 2^{-140.3}$ を得るための active Sbox の数え方を明らかにすることはできなかった．

表 4.1: 初期差分条件 1 での各ラウンドでの active Sbox の最小数.

t	$AS^{(t)}$
-95	0
-94	0
-93	0
-92	0
-91	0
-90	0
-89	0
-88	0
-87	0
-86	0
-85	0
-84	0
-83	1
-82	1
-81	1
-80	1
-79	1
-78	1
-77	1
-76	2
-75	2
-74	3
-73	3
-72	3
-71	6
-70	6
-69	8
-68	8
-67	8
-66	8
-65	8
-64	9
-63	9
-62	11
-61	11

t	$AS^{(t)}$
-60	12
-59	13
-58	15
-57	15
-56	15
-55	16
-54	17
-53	18
-52	18
-51	18
-50	19
-49	20
-48	20
-47	21
-46	22
-45	22
-44	22
-43	22
-42	22
-41	22
-40	22
-39	23
-38	23
-37	24
-36	24
-35	25
-34	27
-33	27
-32	27
-31	28
-30	29
-29	30
-28	30
-27	30
-26	32

t	$AS^{(t)}$
-25	34
-24	34
-23	35
-22	35
-21	37
-20	37
-19	37
-18	38
-17	38
-16	38
-15	39
-14	39
-13	40
-12	40
-11	40
-10	43
-9	43
-8	43
-7	43
-6	44
-5	44
-4	45
-3	45
-2	46
-1	47
0	47
1	49
2	50
3	50
4	51
5	51
6	51
7	51
8	52
9	52

t	$AS^{(t)}$
10	53
11	53
12	55
13	55
14	55
15	57
16	58

表 4.2: 初期差分条件 2 での各ラウンドでの active Sbox の最小数.

t	$AS^{(t)}$
-95	0
-94	0
-93	0
-92	0
-91	0
-90	0
-89	0
-88	0
-87	0
-86	0
-85	0
-84	0
-83	1
-82	1
-81	1
-80	1
-79	1
-78	1
-77	1
-76	2
-75	2
-74	3
-73	3
-72	3
-71	4
-70	5
-69	7
-68	8
-67	8
-66	8
-65	8
-64	9
-63	9
-62	10
-61	11

t	$AS^{(t)}$
-60	12
-59	13
-58	14
-57	15
-56	15
-55	16
-54	17
-53	18
-52	18
-51	18
-50	19
-49	20
-48	20
-47	21
-46	22
-45	22
-44	22
-43	22
-42	22
-41	22
-40	22
-39	23
-38	23
-37	24
-36	24
-35	25
-34	26
-33	27
-32	27
-31	28
-30	29
-29	30
-28	30
-27	30
-26	31

t	$AS^{(t)}$
-25	32
-24	34
-23	35
-22	35
-21	36
-20	37
-19	37
-18	38
-17	38
-16	38
-15	39
-14	39
-13	40
-12	40
-11	40
-10	41
-9	42
-8	43
-7	43
-6	44
-5	44
-4	45
-3	45
-2	46
-1	47
0	47
1	48
2	48
3	49
4	49
5	50
6	50
7	51
8	51
9	52

t	$AS^{(t)}$
10	52
11	53
12	53
13	54
14	54
15	55
16	55

第5章　線形確率に基づく攻撃

5.1　疑似乱数生成器に対する線形確率

8ビット変換に対する最大線形確率を式(3.16)で定義したのと同様に，Enocoro-128v2の状態更新関数Nextの最大線形確率を定義できる．$S^{(t)} = \mathrm{Next}(S^{(t-1)})$に注意して，

$$LP_{\mathrm{Next}}(\Gamma S^{(t-1)}, \Gamma S^{(t)}) = \frac{\#\left\{S^{(t-1)} \in \{0,1\}^{272} | S^{(t-1)} \bullet \Gamma S^{(t-1)} = S^{(t)} \bullet \Gamma S^{(t)}\right\}}{2^{272}} \tag{5.1}$$

$\Gamma S^{(t-1)}$を入力マスク，$\Gamma S^{(t)}$を出力マスクと呼ぶ．ラウンド$t = 0$の内部状態$S^{(0)}$を入力とし，ラウンド$t = R$の内部状態$S^{(t)}$を出力とする関数を考え，その関数に対する最大線形確率を同様に定義することができる．その最大線形確率の近似値として各時刻の最大線形確率の積を考え，それを最大線形特性確率$DLP(R)$と呼ぶ．

$$LCP(R) = \max_{\Gamma S^{(t-1)}} \prod_{t=1}^{R} LP_{\mathrm{Next}}(\Gamma S^{(t-1)}, \Gamma S^{(t)}) \tag{5.2}$$

ここで，$(\Gamma S^{(0)}, \Gamma S^{(1)}, \ldots, \Gamma S^{(R)})$を線形パスという．

$LP_{\mathrm{Next}}(\Gamma S^{(t-1)}, \Gamma S^{(t)})$を求めることが計算量的に難しいため，$LCP(R)$を求めることも難しい．そこで，マスク値$\Gamma S^{(t)}$を8ビット毎にわけ，その8ビットがオールゼロ(non-active)か否(active)かだけを評価することを考える(truncate線形解析)．この場合，272ビットの内部状態のマスク$\Gamma S^{(t)}$は，34ビットのマスク$\Gamma T^{(t)}$に縮約される．マスク値を8ビット毎に分けた理由は，Enocoro-128v2の処理が8ビット単位であるからである．Sboxの出力マスクが非ゼロ(active)であるとき，そのSboxをactive Sboxという．状態更新関数Nextの非線形関数はs_8のみであるから，s_8の最大線形確率をq，ラウンドRまでのactive Sboxの数の最小値をAS_{min}とおくと，

$$LCP(R) \leq q^{AS_{min}} \tag{5.3}$$

となる．Enocoro-128v2の場合，$q = 2^{-4}$なので，$AS_{min} \geq 32$となると線形確率に基づく攻撃に対して安全である．

truncate線形解析の問題点は，ラウンド数が事前に決めることができないことである．仕様上，Enocoro-128v2の最大ラウンド数は2^{64}であるが，2^{64}以下のすべてのラウンドに対して，逐一AS_{min}を求めていくことは難しい．そこで，参考文献では，「閉じるtruncate線形パス」を定義し，探索するラウンド数の制限しつつ，AS_{min}を正確に評価する方法を提案している．

5.2　自己評価による解析結果

自己評価書によれば，$p = 2^{-4}, AS_{min} = 36$である．したがって，$LCP \leq 2^{-144} \leq 2^{-128}$となることから，線形確率に基づく攻撃に対して安全である．active Sboxの数AS_{min}は，計算機実験により得られた値であり，truncate線形パスは自己評価書では示されていない．

5.3 active Sbox 数が最小となる線形パスの探索

状態 $S^{(t)}$ から状態 $S^{(t+1)}$ に遷移する際，XOR 演算，分岐，線形変換 L における truncate マスクの伝搬について考える．

- マスクが分岐点を通過するとき: 分岐点の入力を x，出力を y, z とおき，そのマスク (truncate マスクではない) を $\gamma_x, \gamma_y, \gamma_z$ とおく．出力 y, z は x に等しいので，

$$\gamma_x = \gamma_y + \gamma_z \tag{5.4}$$

  であれば，任意の x に対して，

$$x \bullet \gamma_x = y \bullet \gamma_y + y \bullet \gamma_z \tag{5.5}$$

  が成立する．$\gamma_x, \gamma_y, \gamma_z$ の truncate マスクを $\Gamma_x, \Gamma_y, \Gamma_z$ とおくと，

  - $\Gamma_x = 0$ のとき，$\Gamma_y = 0$ かつ $\Gamma_z = 0$，または，$\Gamma_y = 1$ かつ $\Gamma_z = 1$ である．
  - $\Gamma_x = 1$ のとき，$\Gamma_y = 0$ かつ $\Gamma_z = 1$，または，$\Gamma_y = 1$ かつ $\Gamma_z = 0$，または，$\Gamma_y = 1$ かつ $\Gamma_z = 1$ である．

  となる．つまり，入力された truncate マスクに対して，出力 truncate マスクは一意ではなく，攻撃者が都合のよい出力 truncate マスクを選ぶことができる．

- マスクが XOR 演算点を通過するとき: XOR 演算点の入力を x, y，出力を $, z$ とおき，そのマスク (truncate マスクではない) を $\gamma_x, \gamma_y, \gamma_z$ とおく．このとき，$\gamma_x = \gamma_y$ であれば，γ_z を γ_x に選べば，任意の x, y に対して

$$x \bullet \gamma_x + y \bullet \gamma_y = z \bullet \gamma_z \tag{5.6}$$

  が成立する．$\gamma_x, \gamma_y, \gamma_z$ の truncate マスクを $\Gamma_x, \Gamma_y, \Gamma_z$ とおくと，

  - $\Gamma_x = \Gamma_y = 0$ のとき，$\Gamma_z = 0$ である．
  - $\Gamma_x = \Gamma_y = 1$ のとき，$\Gamma_z = 1$ である．

  つまり，入力 truncate マスクに対して，出力 truncate マスクは一意である．

- マスクが線形変換 L を通過するとき: 線形変換 L は，

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \begin{pmatrix} \mathtt{01} & \mathtt{01} \\ \mathtt{01} & \mathtt{02} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \quad \text{over} \quad \mathrm{GF}(2^8) \tag{5.7}$$

  である (式 (2.6))．u_0, u_1, v_0, v_1 のマスク (truncate マスクではない) をそれぞれ $\gamma_0, \gamma_1, \omega_0, \omega_1$ とおく．今，入力側では $u_0 \bullet \gamma_0 + u_1 \bullet \gamma_1$ の値，出力側では $v_0 \bullet \omega_0 + v_1 \bullet \omega_1$ の値を考え

る．式 (5.7) を用いて，出力側の式を変形をすると，

$$v_0 \bullet \omega_0 + v_1 \bullet \omega_1 = u_0 \bullet (\omega_0 + \omega_1) + u_1 \bullet \omega_0 + (\mathtt{02} \cdot u_1) \bullet \omega_1 \tag{5.8}$$

$$= u_0 \bullet (\omega_0 + \omega_1) + u_1 \bullet \omega_0 + \begin{cases} (u_1 \ll 1) \bullet \omega_1 \\ \qquad (u_1 \text{の最左ビットが} 0 \text{のとき}) \\ (u_1 \ll 1) \bullet \omega_1 + \phi \bullet \omega_1 \\ \qquad (u_1 \text{の最左ビットが} 1 \text{のとき}) \end{cases} \tag{5.9}$$

$$= u_0 \bullet (\omega_0 + \omega_1) + u_1 \bullet (\omega_0 + (\omega_1 \gg 1)) + \begin{cases} 0 \\ \qquad (u_1 \text{の最左ビットが} 0 \text{のとき}) \\ \phi \bullet \omega_1 \\ \qquad (u_1 \text{の最左ビットが} 1 \text{のとき}) \end{cases} \tag{5.10}$$

となる．ここで，ϕ は $\mathrm{GF}(2^8)$ の定義多項式である．次に，

$$u_0 \bullet \gamma_0 + u_1 \bullet \gamma_1 = v_0 \bullet \omega_0 + v_1 \bullet \omega_1 \tag{5.11}$$

が成立する条件を考える．$\gamma_0, \gamma_1, \omega_0, \omega_1$ の truncate マスクを $\Gamma_0, \Gamma_1, \Omega_0, \Omega_1$ とおく．

- $\Gamma_0 = \Gamma_1 = 0$ のとき: $\Gamma_0 = \Gamma_1 = 0$ は $\gamma_0 = \gamma_1 = 0$ なので，任意の u_0, u_1 に対して，$\omega_0 = \omega_1 = 0$ で式 (5.11) が成立する．つまり，$\Omega_0 = \Omega_1 = 0$ である．この場合，入出力 trucate マスクの重みの和は 0 である．
- $\Gamma_0 = 0, \Gamma_1 = 1$ のとき: $\Gamma_0 = 0, \Gamma_1 = 1$ である γ_0, γ_1 が与えられたとき，任意の u_0, u_1 に対して，式 (5.11) が成立する ω_0, ω_1 が存在するならば，$\Omega_0 = \Omega_1 = 1$ である．

  ある u_0, u_1 に対してのみ式 (5.11) が成立する ω_0, ω_1 を考えたとしても，$\omega_0 = \omega_1$ から，$\Omega_0 = \Omega_1 = 1$ になる ($u_0 = u_1 = \mathtt{00}$ の場合を除く)．

  いずれの場合も，$\Gamma_0 = 0, \Gamma_1 = 1$ のときは，入出力 trucate マスクの重みの和は 3 である．

  (例) $\gamma_0 = \mathtt{00}, \gamma_1 = \mathtt{96}$ のとき，$\omega_0 = \omega_1 = \mathtt{e4}$ で式 (5.11) が成立する．しかし，$\gamma_0 = \mathtt{00}, \gamma_1 = \mathtt{97}$ のとき，任意の u_0, u_1 に対して，式 (5.11) が成立する ω_0, ω_1 は存在しない．
- $\Gamma_0 = 1, \Gamma_1 = 0$ のとき: $\Gamma_0 = 1, \Gamma_1 = 0$ である γ_0, γ_1 が与えられたとき，任意の u_0, u_1 に対して，式 (5.11) が成立する ω_0, ω_1 が存在するならば，$\Omega_0 = \Omega_1 = 1$ である．この場合，入出力 trucate マスクの重みの和は 3 である．

  ただし，$\gamma_0 = \mathtt{01}, \gamma_1 = \mathtt{00}$ かつ u_1 の最左のビットが 0 ならば，$\omega_0 = \mathtt{00}, \omega_1 = \mathtt{01}$ で式 (5.11) が成立する．つまり，$\Omega_0 = 0, \Omega_1 = 1$ になる．確率的ではあるが，入出力 trucate マスクの重みの和が 2 になることがある．
- $\Gamma_0 = 1, \Gamma_1 = 1$ のとき: $\Gamma_0 = 1, \Gamma_1 = 0$ である γ_0, γ_1 が与えられたとき，任意の u_0, u_1 に対して，式 (5.11) が成立する ω_0, ω_1 が存在するならば，Ω_0, Ω_1 の少なくとも一方は 1 である．つまり，入出力 trucate マスクの重みの和は 3 以上である．重みの値や重みが 3 の場合，Ω_0, Ω_1 のどちらを 1 にするかは攻撃者が選ぶことができる．

既存研究 [6, 7] では，入力 truncate マスクの重みの和が 0 でない場合，線形変換部 L の入出力 truncate マスクの重みの和は 3 以上として評価を行っている．この場合，線形変換部

L で truncate マスクの線形パスが必ずつながる．上記のように，線形変換部 L の入出力 truncate マスクの重みの和が 2 になることも確率的にありえる．そのため，それが 2 以上とした評価も行う．

線形パスの探索開始は，初期化が終わった時点 $t = 0$ から始める．$b_0^{(0)}, b_2^{(0)}, \ldots, b_{31}^{(0)}, a_0^{(0)}, a_1^{(0)}$ の truncate マスクを

$$\Gamma T^{(0)} = (\Gamma_0^{(0)}, \Gamma_1^{(0)}, \ldots, \Gamma_{33}^{(0)}) \tag{5.12}$$

とおく．ただし，$\Gamma_0^{(0)}, \Gamma_1^{(-0)}, \ldots, \Gamma_{33}^{(0)}$ はすべて 0 とする．

状態 $S^{(t)}$ から状態 $S^{(t+1)}$ へ遷移する過程において，入力 truncate マスク $\Gamma T^{(t)}$ が分岐点，線形変換部を通過するため，出力 truncate マスク $\Gamma T^{(t+1)}$ は上述のとおり一意とは限らない．同じ出力 truncate マスクになる場合は，active Sbox の数が最小の遷移の着目する．$\Gamma T^{(0)}$ からオールゼロの $\Gamma T^{(t)}$ になる線形パスの active Sbox の数 $AS^{(t-1)}$ を求める．その線形パスを閉じたパスと呼ぶ．

5.4 評価結果

線形変換部の入出力マスクの重みの和が 3 以上の場合の各ラウンドの active Sbox の最小個数の結果を表 5.1，線形変換部の入出力マスクの重みの和が 2 以上の場合の結果を表 5.2 に示す．表中の ‘–’ は，そのラウンドでは閉じたパス ($\Gamma T^{(t)}$ がオールゼロになる線形パス) が存在しないことを示す．計算機資源の制約のため，active Sbox の個数が最小になる線形パスを具体的に求めることはできなかった．

それぞれの場合において，最大線形特性確率は，$2^{-144}, 2^{-132}$ 以下になり，Enocoro-128v2 は線形確率に基づく攻撃に対して安全である．線形変換部の入出力マスクの重みの和が 3 以上の場合の最大線形特性確率が 2^{-144} 以下であることは，自己評価書の結果と一致する．また，線形変換部の入出力マスクの重みの和が 2 以上の場合は，そのビット単位での線形パスが存在する確率がさらに低くなるが，その場合でも，鍵の総当たりの攻撃の方が効率が良くなる．

表 5.1: 線形変換部の入出力マスクの重みの和が 3 以上の場合

t	$AS^{(t)}$
1	–
2	–
3	–
4	–
5	–
6	–
7	–
8	–
9	–
10	–
11	–
12	–
13	–
14	–
15	42
16	45
17	42
18	40
19	38
20	39
21	40
22	37
23	39
24	39
25	38
26	37
27	36 *
28	36 *
29	39
30	40
31	37
32	39
33	39
34	42
35	39

t	$AS^{(t)}$
36	40
37	38
38	39
39	39
40	40
41	39
42	40
43	39
44	39
45	42
46	41
47	41
48	41
49	41
50	42
51	42
52	41
53	42
54	41
55	42
56	42
57	44
58	44
59	42
60	44
61	45
62	45
63	45
64	44
65	45
66	45
67	45
68	45
69	47
70	48

t	$AS^{(t)}$
71	47
72	48
73	48
74	48
75	48
76	48
77	48
78	48
79	50
80	50
81	50
82	50
83	50
84	50
85	50
86	51
87	51
88	52
89	52
90	53
91	53
92	53
93	53
94	54
95	54
96	54
97	54
98	55
99	55
100	55

表 5.2: 線形変換部の入出力マスクの重みの和が 2 以上の場合

t	$AS^{(t)}$
1	–
2	–
3	–
4	–
5	–
6	–
7	–
8	–
9	–
10	–
11	–
12	–
13	–
14	–
15	38
16	41
17	39
18	37
19	36
20	35
21	36
22	34
23	36
24	34
25	36
26	35
27	34
28	35
29	33 *
30	35
31	35
32	34
33	35
34	35
35	34

t	$AS^{(t)}$
36	34
37	33
38	36
39	35
40	35
41	34
42	35
43	35
44	34
45	34
46	33 *
47	34
48	36
49	35
50	35
51	37
52	36
53	35
54	36
55	36
56	37
57	37
58	37
59	37
60	38
61	39
62	40
63	39
64	39
65	39
66	38
67	40
68	40
69	41
70	40

t	$AS^{(t)}$
71	41
72	42
73	42
74	42
75	42
76	42
77	42
78	43
79	44
80	43
81	44
82	43
83	44
84	45
85	45
86	45
87	46
88	45
89	46
90	46
91	45
92	45
93	47
94	47
95	47
96	47
97	48
98	48
99	47
100	49

第6章　代数的な攻撃

6.1　代数的な性質に着目した攻撃

自己評価書では，線形フィードバックシフトレジスタを使ったフィルタ型生成器やコンバイナ型生成器に適用される XSL 攻撃や Grobner 基底を用いた代数的攻撃を Enocoro-128v2 に適用することは難しいと述べている．

最近，疑似乱数生成器 (ストリーム暗号) の代数的な性質に着目した攻撃として，cube 攻撃が提案された [4]．Trivium や Grain-128 に cube 攻撃を適用した場合の安全性解析が報告されており，従来の攻撃法よりも有効であることが示されている [1][2][4]．cube 攻撃の原理は，4 章で述べた差分確率を拡張した高階差分攻撃と類似しており，関数の次数が低い場合に有効である．自己評価書では，cube 攻撃に対する安全性評価の記述はない．

cube 攻撃を用いた鍵復元攻撃の原理を説明する．3 ビットの鍵を k_1, k_2, k_3，2 ビットの初期値を v_1, v_2 とし，1 ビット出力の疑似乱数生成器を考える．このとき，P の出力が GF(2) 上の多項式

$$P_{k_1,k_2,k_3}(v_1, v_2) = v_1v_2k_1 + v_1v_2k_2 + v_2k_2k_3 + v_1v_2 + v_2 + k_1k_3 + v_1k_3 + 1 \tag{6.1}$$
$$= v_1v_2(k_1 + k_2 + 1) + v_2k_2k_3 + v_2 + k_1k_3 + v_1k_3 + 1 \tag{6.2}$$
$$= v_1(v_2k_1 + v_2k_2 + v_2 + k_3) + v_2k_2k_3 + v_2 + k_1k_3 + 1 \tag{6.3}$$

で表現できているとする．このとき，$(v_1, v_2) \in \{0, 1\}^2$ について，出力のすべて和をとると，

$$\sum_{(v_1,v_2)\in\{0,1\}^2} P_{k_1,k_2,k_3}(v_1, v_2) = k_1 + k_2 + 1 \tag{6.4}$$

となる．つまり，鍵 k_1, k_2, k_3 が固定された疑似乱数生成器 P がブラックボックスとして与えられたとき，初期値を変化させて出力の和を計算すると，鍵ビットに関する情報 $k_1 + k_2 + 1$ が得られる．$k_1 + k_2 + 1$ は，cube (v_1, v_2) に対する superpoly と呼ばれ，この例のように，1 次式の superpoly を max term と呼ぶ．これは，式 (6.2) の () の多項式であることに注意しよう．次に，$v_2 = 0$ として，$v_1 \in \{0, 1\}$ についてすべて和をとると，

$$\sum_{v_1\in\{0,1\}} P_{k_1,k_2,k_3}(v_1, 0) = k_3 \tag{6.5}$$

となり，鍵ビット k_3 が求められる．これは，式 (6.3) の () の多項式に $v_2 = 0$ を代入した式である．このように，疑似乱数生成器の多項式 $P_{k_1,k_2,k_3}(v_1, v_2)$ が既知の場合，初期値を適切に選び，和をとることで，鍵に関する情報を得ることができる．

しかし，疑似乱数生成器の場合，鍵が固定された多項式 $P_{k_1,k_2,k_3}(v_1, v_2)$ が攻撃者に明示的に与えられることはない．したがって，攻撃者は，まず，P を推定する必要がある．この場合，鍵も未知数と考え，多項式 $P(k_1, k_2, k_3, v_1, v_2)$ とおく．$P(k_1, k_2, k_3, v_1, v_2)$ は，形式的に

$$P(k_1, k_2, k_3, v_1, v_2) = v_1v_2 \cdot g(k_1, k_2, k_3) + h(k_1, k_2, k_3, v_1, v_2) \tag{6.6}$$

と書くことができる．ここで，$g(k_1,k_2,k_3)$ は，k_1,k_2,k_3 の関数であるが，攻撃者にとってその形は不明である．$h(k_1,k_2,k_3,v_1,v_2)$ は，v_1v_2 を因数として含まない多項式であり，やはり，攻撃者にとってその形は不明である．まず，攻撃者は，例えば $(k_1,k_2,k_3)=(1,0,0)$ として，

$$u_{1,0,0}=\sum_{(v_1,v_2)\in\{0,1\}^2} P(1,0,0,v_1,v_2) \tag{6.7}$$

の値を求める．P のアルゴリズムは攻撃者に既知であるから，この計算は可能である．このとき，

$$u_{1,0,0}=g(1,0,0) \tag{6.8}$$

が成立していることに注意しよう．また，$(k_1,k_2,k_3)=(0,1,0)$ とすれば，

$$u_{0,1,0}=\sum_{(v_1,v_2)\in\{0,1\}^2} P(1,0,0,v_1,v_2) \tag{6.9}$$

$$=g(0,1,0) \tag{6.10}$$

である．このようにして式 (6.8) や式 (6.10) のような式を多く集めることで，鍵の情報を推定できる．例えば，$g(k_1,k_2,k_3)$ が線形であれば，つまり，cube (v_1,v_2) に対する superpoly が max term であれば，それらは線形の連立方程式となるので，$g(k_1,k_2,k_3)$ が容易に求められる．

この例からわかるように，疑似乱数生成器を多項式で表現したとき，その次数が低く，かつ max term が存在する場合に，cube 攻撃による鍵回復攻撃が可能になる．Trivium の場合，初期化ラウンド数が少ない場合，12 変数の cube に対して max term が多く見つかっており，効率のよい鍵回復攻撃が可能になる [4].

また，疑似乱数生成器 P の次数を d としたとき，d 次の多項式の集合の中から一様ランダムに選ばれた関数 G と P を cube 攻撃を用いて識別する，いわゆる乱数識別問題に適用することも報告されている [2].

6.2 cube 攻撃の攻撃アルゴリズム

本章では，Enocoro-128v2 に対して cube 攻撃により鍵回復を行うアルゴリズムを述べる．Enocoro-128v2 は，8 ビット出力の疑似乱数生成器であるが，着目するのは，その中の 1 ビットである．出力 1 ビットに着目したときの Enocoro-128v2 の関数を $f(\mathbf{k},\mathbf{v},\mathbf{u})$ とおく．ここで，$\mathbf{k}$ は，鍵バイトに対応し，

$$\mathbf{k} = b_0\|b_1\|\ldots\|b_{15},\quad b_i\in\{0,1\}^8 \tag{6.11}$$

$$= k_0\|k_1\|\ldots\|k_{127},\quad k_i\in\{0,1\} \tag{6.12}$$

とおく．$\mathbf{v},\mathbf{u}$ は，IV に対応するが，対応するビットはその都度定める．

1. $\mathbf{k}_0,\mathbf{k}_1,\ldots,\mathbf{k}_{255}$ を

$$\mathbf{k}_0=0^{128},\quad \mathbf{k}_1=10^{127},\quad \mathbf{k}_2=\mathbf{k}_1\gg 1=010^{126}, \tag{6.13}$$

$$\mathbf{k}_{2i+1}=\mathbf{k}_{2i}\oplus\mathbf{k}_{2i-1},\quad \mathbf{k}_{2i}=\mathbf{k}_{2i-2}\gg 1\ \ (i=2,3\ldots,127) \tag{6.14}$$

で定める．ここで，$\gg$ は右シフトを表す．0^x は 0 が x 個連続することを意味する．

2. IV の 64 ビットのなかから，t ビットを選ぶ．その t ビットの位置を $I_\eta = \{\eta_0, \eta_1, \ldots, \eta_{t-1}\}$ とおく．I_η に対応する cube を

$$\mathbf{v}_\eta = (v_{\eta_0}, v_{\eta_1}, \ldots, v_{\eta_{t-1}}) \tag{6.15}$$

とおく．cube に含まれない残りの $64-t$ ビット $\mathbf{u}$ はすべて 0 とする．

(a) $i = 0, 1, \ldots 255$ に対して

$$s_{\eta,i} = \sum_{\mathbf{v}_\eta \in \{0,1\}^t} f(\mathbf{k}_i, \mathbf{v}_\eta, \mathbf{0}) \text{ over GF(2)} \tag{6.16}$$

を計算する．

(b) $f(\mathbf{k}, \mathbf{v}_\eta, \mathbf{0})$ の出力ビットの線形性を次の方法で検査する．すべての $i \in \{1, 2, \ldots, 127\}$ に対して，

$$s_{\eta,0} + s_{\eta,2i-1} + s_{\eta,2i} = s_{\eta,2i+1} \text{ over GF(2)} \tag{6.17}$$

が成立するかどうか調べる．もし上式が成立しない i がある場合，cube $(v_{\eta_0}, v_{\eta_1}, \ldots, v_{\eta_{t-1}})$ の superpoly は線形でないので，2. に戻る．すべての i に対して，成立するならば，superpoly は線形 (つまり，maxterm) の可能性が極めて高い．

(c) $c_{\eta,i} \in \{0, 1\}$ を未知の係数として，superpoly を $sp_{\eta,j}$ とおくと，

$$sp_{\eta,j} = \sum_{i=0}^{127} c_{\eta,i} k_i + c_{\eta,128} \tag{6.18}$$

このとき，$\mathbf{k}_i$ の決め方から，

$$\begin{pmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \mathbf{k}_4 \\ \mathbf{k}_8 \\ \vdots \\ \mathbf{k}_{254} \\ \mathbf{k}_0 \end{pmatrix} \begin{pmatrix} c_{\eta,0} \\ c_{\eta,1} \\ c_{\eta,2} \\ c_{\eta,3} \\ \vdots \\ c_{\eta,127} \\ c_{\eta,128} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ & & \ldots & & \\ 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \begin{pmatrix} c_{\eta,0} \\ c_{\eta,1} \\ c_{\eta,2} \\ c_{\eta,3} \\ \vdots \\ c_{\eta,127} \\ c_{\eta,128} \end{pmatrix} \tag{6.19}$$

$$= \begin{pmatrix} s_{\eta,1} \\ s_{\eta,2} \\ s_{\eta,4} \\ s_{\eta,8} \\ \vdots \\ s_{\eta,254} \\ s_{\eta,0} \end{pmatrix} \tag{6.20}$$

である．したがって，$sp_{\eta,j}$ の係数 $c_{\eta,i}$ 決定できる．

3. 上記 2 から 5 を繰り返し，線形独立な $sp_{\eta,j}$ を 128 個求める．ここで，線形独立とは，superpoly の 128 個の係数をベクトルと見たとき，それらが線形独立であることを意味する．これら 128 個の superpoly を改めて，$sp_0, sp_1, \ldots sp_{127}$，とし，$sp_i$ に対応する cube を $\mathbf{v}_i$ とおく．

4. 未知の鍵が固定されたEnocoro-128v2の1ビット出力の関数を $f_{\mathbf{k}}(\mathbf{v}, \mathbf{u})$ とおく．$i = 0, 1, \ldots, 127$ に対して，

$$w_i = \sum_{\mathbf{v}_i \in \{0,1\}^t} f_{\mathbf{k}}(\mathbf{v}_i, 0^{64-t}) \text{ over GF}(2) \tag{6.21}$$

を計算する．

5. $i = 0, 1, \ldots, 127$ に対して

$$w_i = sp_i(\mathbf{k}) \tag{6.22}$$

である．sp_i は線形なので，この連立方程式を解くと，鍵 $\mathbf{k}$ が得られる．

この鍵回復攻撃において，cubeのサイズ t はパラメータである．$f(\mathbf{k}, \mathbf{v}, \mathbf{u})$ が明示的に分からないので，t の値を事前に決めることは難しい．t が大きくなると，式(6.16)の計算時間が指数的に増加するため，あまり大きくすることはできない．攻撃に必要な計算量のほとんどは，128個の $sp_{\eta,j}$ を計算量である．

6.3 評価結果

$t = 31$ とした場合，${}_{64}\mathrm{C}_{31}$ $(\approx 1.77 \times 10^{18})$ 個の $\mathbf{v}_\eta$ が存在する．そのうちランダムに選んだ約12,300個の $\mathbf{v}_\eta$ に対して，初期化ラウンドを64から96までの間で線形なsuperpolyを計算機で探索を行ったが，線形なsuperpolyを見つけることはできなかった．

s_8 の次数は6であり，表3.9から表3.16から分かるように，s_8 の各出力ビットの多項式はかなり密である．したがって，cube攻撃は，Enocoro-128v2に対して有効な攻撃方法ではと思われる．

第7章　結論

本報告では，Enocoro-128v2 の Sbox の評価を述べた後，差分確率/線形確率に基づく攻撃に対する安全性評価の方法と計算機実験による評価結果を述べた．差分確率/線形確率に基づく攻撃に対する安全性については，自己評価書 [5] 及び参考文献 [6, 7, 9] に記載されている評価結果を否定するような結果は得られなかった．したがって，これらの攻撃に対して Enocoro-128v2 は安全である．

次に，代数的な性質を利用した攻撃として，cube 攻撃について述べた．cube 攻撃は，最近提案された新しい攻撃方法であり，自己評価書では触れられていない．本報告では，cube 攻撃の原理及び Enocoro-128v2 への適用方法について詳述した．Enocoro-128v2 への cube 攻撃は，あるパラメータに対して計算機実験を行ったが，出力系列と初期値の間に鍵回復攻撃に役立つような関係式を見つけることができなかった．他のパラメータの cube 攻撃について計算機実験を行うことはできなかったが，Sbox のブール多項式の項が密であることから，他のパラメータの cube 攻撃が有効でないことが予想できる．

以上より，Enocoro-128v2 には，差分確率/線形確率/代数次数を利用した攻撃法に対して，脆弱性は見当たらない．

関連図書

[1] J.-P. Aumasson, I. Dinur, L. Henzen, W. Meier, and A. Shamir, "Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128," SHARCS'09 - Special-purpose Hardware for Attacking Cryptographic Systems, 2009. http://www.131002.net/data/papers/ADHMS09.pdf.

[2] J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir, "Cube testers and key recovery attacks on reduced-round MD6 and Trivium," Fast Software Encryption, FSE 2009, Lecture Notes in Computer Science, vol. 5665, pp. 1–22, 2009.

[3] M. Bellare and T. Kohno, "A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications," Advances in Cryptology - EUROCRYPT 2004, Lecture Notes in Computer Science, vol. 2656, pp. 491–506, 2003. http://www-cse.ucsd.edu/users/mihir/papers/rka.pdf.

[4] I. Dinur and A. Shamir, "Cube attacks on tweakable black box polynomials," Cryptology ePrint Archive, Report 2008/385, 2008. http://eprint.iacr.org/.

[5] 株式会社日立製作所, 疑似乱数生成器 Enocoro 評価書, http://www.cryptrec.go.jp/topics/cryptrec_20101001_callforattack.html, 2010.

[6] 鴻巣慧, 武藤健一郎, 古市洋希, 渡辺大, 金子敏信, "Enocoro-128 ver.1.1 の再同期攻撃耐性評価," 電子情報通信学会技術研究報告, Vol. 107, No. 503, pp. 7–13, 2008.

[7] 武藤健一郎, 渡辺大, 金子敏信. "Enocoro-128 の LDA 耐性評価と改良," 2008 年暗号と情報セキュリティシンポジウム予稿集, No. 4A1-1, 2008.

[8] 岡本和人, 武藤健一郎, 金子敏信, "疑似乱数生成器 Enocoro-80 の差分／線形攻撃耐性評価（II）," 2009 年暗号と情報セキュリティシンポジウム予稿集, No. 4B2-3, 2009.

[9] 渡辺大, 岡本和人, 金子敏信, "軽量ハードウェア向け疑似乱数生成器 Enocoro-128v2," 2010 年暗号と情報セキュリティシンポジウム予稿集, No. 3D1-3, 2010.

[10] 五十嵐保隆, 岡本和人, 金子敏信, "関連鍵攻撃による Enocoro の弱鍵復元の検討," 電子情報通信学会技術研究報告, Vol. 109, No. 446, pp. 275–280, 2010.