Enocoro-80: A Hardware Oriented Stream Cipher

Dai Watanabe, Kota Ideguchi, Jun Kitahara
Systems Development Laboratory,
Hitachi, Ltd.
1099 Ozenji, Asao-ku, Kawasaki,
Kanagawa, 215-0013, Japan.
dai.watanabe.td@hitachi.com

Kenichiro Muto, Hiroki Furuichi,
Toshinobu Kaneko
Tokyo University of Science
2641 Yamazaki, Noda, Chiba,
278-8510, Japan.

Abstract

Panama is a pseudo-random number generator proposed by Daemen and Clapp in 1998. It is designed to achieve high performance in software implementations, and the throughput is twice as fast as that of the AES block cipher. On the other hand, the size optimized implementation in hardware circuit is much larger than that of the AES. We are attempted to design a new hardware oriented PRNG which has a similar structure to Panama. In this paper, a family of PRNGs Enocoro is presented and especially a specific algorithm Enocoro-80 for 80-bit key is given. Enocoro-80 can be implemented with 2.7 Kgates in ASIC. The implementation results are comparable to other ciphers selected as the eSTREAM Profile 2 candidates (hardware oriented ciphers). As a result, we confirm that the design of Panama is suitable to not only a software oriented cipher, but also a hardware oriented cipher.

1 Introduction

A stream cipher is an encryption mechanism which uses pseudorandom bit strings (or keystreams). Stream ciphers are generally much faster than block ciphers, another encryption mechanism, so that they are widely used in various areas. A stream cipher is decomposed into two mechanisms. One is a pseudorandom number generator (PRNG) which generates keystreams of arbitrary length from secret keys of fixed length. The other is the mode of operations of those keystream generators; which provides the encryption mechanism by combining plaintexts and keystreams to generate ciphertexts. The typical way in the latter mechanism is so called binary additive stream cipher which combines plaintexts and keystreams by the xoring operation. Because of the simple encryption mechanism, a PRNG plays a main role in the research for stream ciphers.

A PRNG includes a finite state machine (FSM) and a linear feedback shift register (LFSR) is a class of FSMs whose update function being a linear map of canonical form. Due to its linear property, its behavior is well evaluated and it is widely used as an core module of many PRNGs. However, one claims the potential vulnerability also due to the linear property and it motivates the research for non-linear update functions. PANAMA is a PRNG designed by Daemen and Clapp in 1998 [5]. Its update function includes the same structure as a round function of a block cipher and it has a huge internal state. Its simple construction motivates to design the variants and MUGI, a word-wise variant of PANAMA was proposed by Watanabe *et. al.* in 2001 [20]. These two algorithms are mainly suited to software implementations. However, their internal states are so large that their hardware implementations tend to larger than hardware oriented ciphers. The basic structure of PANAMA is newer than LFSR-based ciphers, and there is no example suited to hardware implementations.

In this paper we are attempted to design a new hardware oriented PRNG whose structure is similar to PANAMA. The *Enocoro* is a PRNG family and a set of parameters recommended for an 80-bit key cipher is presented. The ASIC implementations requires 2.7 Kgates and it is comparable to other ciphers selected as the eSTREAM Profile 2 candidates. In addition, their software implementations in C language achieves better performance than most of Profile 2 ciphers. As a result, we confirm that the design of PANAMA is applicable not only for a software oriented cipher, but also for a hardware oriented cipher.

The organization of this paper is as follows. At first the basic notations used in this paper is defined in Section 2. Next, the specification of the new PRNG *Enocoro* is given in Section 3. Then the design strategy of *Enocoro* is presented in Section 6. In Section 4 and Section 5, some results on implementations and security issues are presented. Finally we conclude this paper in Section 7.

0-7695-3102-4/08 $25.00 © 2008 IEEE
DOI 10.1109/ARES.2008.84

2 Preliminary

2.1 Pseudorandom Number Generator

A PRNG consists of a finite state machine (FSM), an initialization function $Init$, and an output function Out. A FSM consists of a internal state (or a register) $S^{(t)}$ depending on the clock and its update function $Next$. The initialization function generates the initial internal state $S^{(0)}$ from the initial inputs such as a secret key K and an initial vector I. The output function generates output bits $Z^{(t)}$ from the internal state $S^{(t)}$ at each time t.

$$\begin{aligned} S^{(0)} &= Init(K, I), && (1)\\ Z^{(t)} &= Out(S^{(t)}), && (2)\\ S^{(t+1)} &= Next(S^{(t)}). && (3) \end{aligned}$$

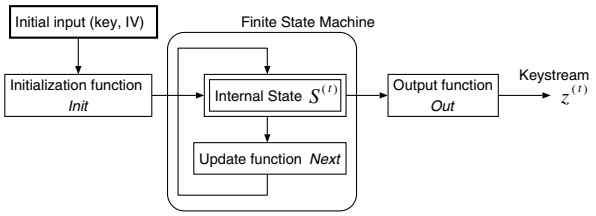

Figure 1. Pseudorandom Number Generator

A PANAMA-like keystream generator (PKSG) is a class of PRNGs and is a generalization of software oriented PRNG PANAMA. The internal state of a PKSG is separated into two: a *state* $a^{(t)}$ and a *buffer* $b^{(t)}$. The update functions are denoted by ρ and λ respectively and both functions take the other sub-internal state as a parameter. The whole update function $Next$ is a composition of ρ and λ.

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

In general, the state a is small compared to the buffer b and the buffer occupies most of the memory. In addition, the update function λ is a linear function. If λ does not take $a^{(t)}$ as a parameter, the PKSG is nothing different from a traditional LFSR-based PRNG called 'combiner with memory'.

2.2 Finite Field

$\varphi_4(x)$ and $\varphi_8(x)$ are the irreducible polynomials of degree 4 and degree 8 over $\mathrm{GF}(2)$. They are defined as

$$\begin{aligned} \varphi_4(x) &= x^4 + x + 1,\\ \varphi_8(x) &= x^8 + x^4 + x^3 + x + 1. \end{aligned}$$

In the specification of *Enocoro*, field multiplications over $\mathrm{GF}(2^4)$ and $\mathrm{GF}(2^8)$ are used. Above two polynomials are the definition polynomials of $\mathrm{GF}(2^4)$ and $\mathrm{GF}(2^8)$ respectively.

3 Specification

In this section, the specification of *Enocoro* PRNG family is given. The algorithm of *Enocoro* is specified by 11 parameters. Let the buffer size of *Enocoro* in bytes be n_b, the inputs from the buffer to the ρ function be $b_{k_1}, b_{k_2}, b_{k_3}, b_{k_4}$. The parameters which defines the λ function are denoted by $q_1, p_1, q_2, p_2, q_3, p_3$. It is denoted by $Enocoro(n_b; k_1, \ldots, k_4, q_1, p_1, \ldots, q_3, p_3)$ if the parameters should be specified.

3.1 Internal State

The state a consists of two bytes. The higher byte is denoted by a_0 and the lower byte is denoted by a_1. The buffer b consists of n_b bytes. They are denoted by $b_0, b_1, \ldots, b_{n_b-1}$ in rotation.

3.2 Function Rho

The update function of the state ρ of *Enocoro* takes $b_{k_1}, \ldots, b_{k_4}$ as external inputs. The ρconsists of referring Sboxes, the linear transformation L defined over $\mathrm{GF}(2^8)$, and XORings. The detailed transformation is defined as

$$\begin{aligned} u_0 &= a_0^{(t)} \oplus s_8[b_{k_1}^{(t)}],\\ u_1 &= a_1^{(t)} \oplus s_8[b_{k_2}^{(t)}],\\ (v_0, v_1) &= L(u_0, u_1),\\ a_0^{(t+1)} &= v_0 \oplus s_8[b_{k_3}^{(t)}],\\ a_1^{(t+1)} &= v_1 \oplus s_8[b_{k_4}^{(t)}]. \end{aligned}$$

3.2.1 Linear Transformation

The linear transformation L is chosen from the 2-by-2 matrix over $\mathrm{GF}(2^8)$ such that $(I_2|L)$ is a generator matrix of a maximum distance separable code. For 2-by-2 matrix, this condition is equivalent to satisfy the following two conditions:

- All elements are not zero.
- The discriminant of the matrix is not zero.

The linear transformation of *Enocoro* is defined as

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = L(u_0, u_1) = \begin{pmatrix} 1 & 1 \\ 1 & d \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}, \quad d \in \mathrm{GF}(2^8).$$

Especially, $d =$ `0x02` is adopted in *Enocoro*-80 which is specific algorithm described later.

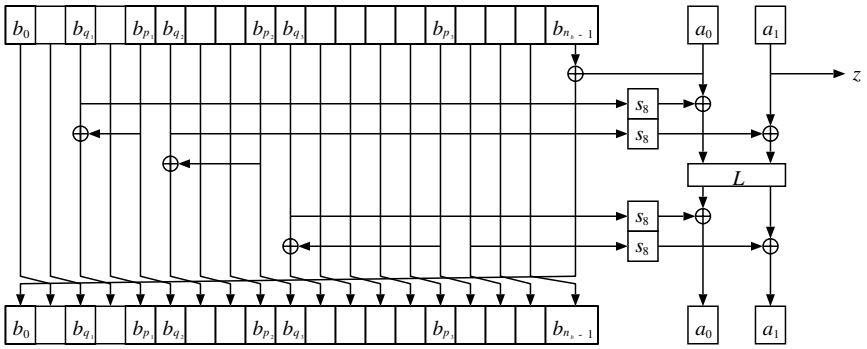

Figure 2. Outline of the structure of *Enocoro*

3.2.2 Sbox

The Sbox (substitution box) s_8 defines a permutation which maps 8-bit inputs to 8-bit outputs. It has also SPS structure and it consists of 4 small Sboxes s_4 which map 4-bit inputs to 4-bit outputs and a linear transformation l defined by a 2-by-2 matrix over $\mathrm{GF}(2^4)$. The Sbox s_4 is defined as

$$s_4[16] = \{1, 3, 9, 10, 5, 14, 7, 2, 13, 0, 12, 15, 4, 8, 6, 11\}.$$

The linear transformation l is defined as

$$l(x, y) = \begin{pmatrix} 1 & e \\ e & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \quad x, y, e \in \mathrm{GF}(2^4)$$

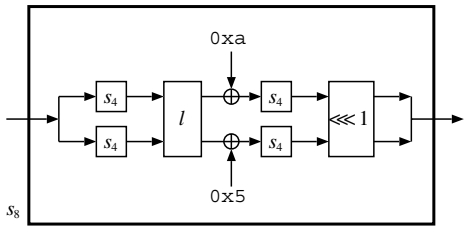

Figure 3. Sbox s_8

The Figure 3 shows how to construct the 8-by-8 Sbox s_8. The Sbox s_8 is defines as

$$\begin{aligned} y_0 &= s_4[s_4[x_0] \oplus e \cdot s_4[x_1] \oplus \mathtt{0xa}], \\ y_1 &= s_4[e \cdot s_4[x_0] \oplus s_4[x_1] \oplus \mathtt{0x5}]. \end{aligned}$$

$e = \mathtt{0x04}$ is used for *Enocoro*-80. The output is rotated 1 bit to the left at the end.

$$s_8[x] = (y_0 || y_1) \lll_8 1.$$

3.3 Function Lambda

The λ function of *Enocoro* consists of three feedbacks (XORings), XORing a_0 to the most right byte of the buffer b_{n_b-1}, and a byte-wise rotation of the buffer. The detailed transformation is defined as follows:

$$\begin{aligned} b_i^{(t+1)} &= b_{i-1}^{(t)}, \quad i \neq 0, q_1+1, q_2+1, q_3+1, \\ b_0^{(t+1)} &= b_{n_b-1}^{(t)} \oplus a_0^{(t)}, \\ b_{q_j+1}^{(t+1)} &= b_{q_j}^{(t)} \oplus b_{p_j}^{(t)}, \quad j = 1, 2, 3, \end{aligned}$$

where $p_i - q_i \neq p_j - q_j$ if $i \neq j$.

3.4 Output function Out

The output function of *Enocoro* outputs the lower byte of the state.

$$Out(S^{(t)}) = a_1^{(t)}.$$

3.5 *Enocoro*-80: Parameters and Initializations

The key length of *Enocoro*-80 is 80 bits and the IV length is 64 bits. The algorithm is specified as $Enocoro(20; 1, 4, 6, 16, 1, 3, 4, 5, 6, 15)$. The initial state $S^{(0)} = (a^{(0)}, b^{(0)})$ is defined as

$$\begin{aligned} b_i^{(-40)} &= K_i, \quad 0 \leq i < 10, \\ b_{i+10}^{(-40)} &= I_i, \quad 0 \leq i < 8, \\ b_{18}^{(-40)} &= \mathtt{0x66}, b_{19}^{(-40)} = \mathtt{0xe9}, \\ a_0^{(-40)} &= \mathtt{0x4b}, a_1^{(-40)} = \mathtt{0xd4}, \\ S^{(0)} &= Next^{40}(S^{(-40)}), \end{aligned}$$

where $Next^{40}(S^{(-40)})$ means the 40 times iteration of the update function $Next$.

3.6 Limitation

The length of the output with a key and an IV should be less than 2^{32} bytes.

4 Implementations

In this section, the hardware and software implementations of *Enocoro*-80 are presented.

4.1 Hardware Implementation

Table 1 shows the implementation results of *Enocoro*-80 and some eSTREAM Profile 2 ciphers in ASIC. The results on eSTREAM ciphers are referred from the technical report by Good and Benaissa [12] (Their result on AES is also referred from [18]). The cost and the throughput of *Enocoro* in ASIC design is almost same as other ciphers.

4.2 Software Implementation

Table 2 shows the implementation results of *Enocoro*-80 and some eSTREAM Profile 2 ciphers on Intel Pentium® 4 processor. The results on eSTREAM ciphers are referred from [8]. *Enocoro*-80 is the second fastest in this comparison table. This is mainly due to the suitability to software implementations of byte-wise operations.

Table 2. The comparison of software implementations [8]

Algorithm	Throughput (Cycle/byte)
DECIM v2	12845.4
Edon80	4061.0
F-FCSR-H	77.5
Grain80	57.3
MICKEY 2.0	1109.6
Pomaranch	1879.2
Trivium	8.4
Enocoro-80	**28.0**

5 Security

In this section, the security of *Enocoro*-80 against several attacks are considered.

5.1 Time-Memory-Trade-Off Attack

A time-memory-trade-off (TMTO) attack on stream ciphers are firstly proposed by Babbage and Golić independently [1, 11]. Their attacks indicate a new insight about the minimum requirement on the size of the internal state. The cost of TMTO attacks on stream ciphers is at least $O(2^{n/2})$, where n is the size of the internal state of the stream cipher. *Enocoro*-80 has a 176 bits internal state, it is more than twice as large as its key length. Hence TMTO attacks are expected to be difficult to apply.

5.2 Guess and Determine Attack

A guess and determine attack (GD-attack) is proposed by Bernbach and Patel [4]. GD-attack is a kind of optimized exhaustive search of the possible internal state corresponding to the given output sequence and it is especially effective for stream ciphers which adopt word-wise operations. The basic idea of the GD-attack is that the attacker guesses some words at the beginning of the attack, and determine other words by using the relations coming from the update function and the output function. The output string reveals the information of the internal state, this strategy provides a better attack than the exhaustive search of the internal state in general.

Our first approach is to guess some words randomly, and to iterate determine processes. The best attack in our trial requires 88 bits guess [14]. The second approach is to check all possible (byte-wise) guesses by using mathematical induction and it was proved that there is no GD-attack if the number of guess is less than 56 bits.

5.3 Linear Distinguishing Attack

A linear distinguishing attack is a kind of theoretical randomness testing of the output sequence which is based on the idea originated at the linear attack [15]. The efficient technique to evaluate the resistance against the linear distinguishing attack on the word-oriented stream ciphers was proposed by Sekine *et.al.* As a result of the evaluation, we confirmed that the linear distinguishing attack on *Enocoro*-80 requires at least 2^{108} bytes output [16].

5.4 Re-synchronization Attack

A re-synchronization attack is proposed by Daemen *et.al.* [6] and is an generic attacker's model in which the attacker concentrate to find the weakness of the initialization function $Init$. We applied the same technique as it is used in Section 5.3 to evaluate the resistance of *Enocoro* against the re-synchronization attack using linear attacks and we confirmed that the attack requires at least 2^{144} trials [17]. The re-synchronization attack based on differential attacks [3] is another example and the similar evaluation technique is applicable. We confirmed that the differential based re-synchronization attack requires at least about $2^{112.3}$ trials [10].

Table 1. The comparison of hardware implementations [12]

Algorithm	Max. clock freq. (MHz)	Area (Kgate)	Interface bits (bit/cycle)	Process (μm)
Grain80	724.6	1.3	1	0.13
	632.9	2.2	8	0.13
Trivium	358.4	2.6	1	0.13
	359.7	2.8	8	0.13
F-FCSR-H	392.2	4.8	8	0.13
Enocoro-80	**274.7**	**2.7**	**8**	**0.18**

5.5 Collision of the Internal State

The internal state of hardware oriented stream ciphers are much smaller that of software oriented ciphers and this may lead undesirable property as a cipher. The collisions of the internal state is an example.

Let $S_0(K, I)$ be the initial state corresponding to a secret key K and an IV I. Two initial values (K, I) and (K', I') are called a sliding pair if $S_0(K, I) = S_T(K', I')$ for a round T. Both *Enocoro*-80's initialization function and the update function are bijective so that the inverse image of $S_0(K, I)$ by the map $Init$ and T times iteration of $Next$ is easily calculated. If the lower two bytes of the buffer and the two bytes of the state of the inverse image are equal to the fixed constant, the inverse image is valid, i.e., an initial value (K', I') exists such that $S_0(K, I) = S_T(K', I')$. The constraint to find a sliding pair is just the 4 bytes constants. Hence, for any round t, there is an initial value (K', I') such that $S_0(K, I) = S_t(K', I')$ with probability 2^{-32}. The usage of *Enocoro*-80 allows to output up to 2^{32} bytes, so that the attacker can assume $S_0(K, I)$ as S_t and check its inverse image by the map $Init$ and t times iterations of $Next$ for any $0 < t < 2^{32}$. In other words, the attacker can generate 2^{32} candidates of the fragment of the pair. Therefore it is expected that a sliding pair exists for any initial value (K, I) on the average. However, the obtained pair (K, I) and (K', I') has no relation except the sliding property. We believe that this kind of property does not threaten the security of the cipher in practical usage.

6 Design Strategy

6.1 Byte-wise Operations

80-bit key ciphers have more than 160 bits internal state. However, it is computationally difficult to evaluate the resistance against various attacks with all possible initial values. In a modern block cipher design, most of the transformation consists of byte-wise or word-wise operations to be suited to software implementations. The security evaluations use this block-wise structure and the technique is called truncated evaluation. Especially truncated evaluations against differential and linear attacks provide satisfactory approximation of the lower bounds of the complexity of the attacks. This success motivates us to design a stream cipher consisting of byte-wise operations and to give the accurate security evaluations. As a result, we could evaluate a lot of candidates if they are resistant against the linear distinguishing attack before fixing parameters.

6.2 Structure of the function ρ

The function ρ of *Enocoro* does not have a substitution-permutation-network (SPN), but has a substitution-permutation-substitution (SPS) structure to make the update function $Next$ be stronger. Because of the additional substitution, the critical path of the SPS structure tends to longer than a SPN structure in general. Different from the standard SPS, the Sboxes of *Enocoro*'s ρ substitute not the state block data, but the inputs from the buffer. This transformation intends to avoid the performance of the SPS going down in hardware implementations. In addition, this transformation preserves the resistance against the linear distinguishing attack.

6.3 One Structure for Two Different Key Lengths

The AES algorithm allows three key lengths and the more round functions are applied to if the longer key size is used. But stream ciphers have to take into account TMTO attacks so that at least the state size should be at least twice as large as the key length. Besides, the state size enormously has an influence on the implementation size in hardware circuit so that the state size should be chosen according to the key length.

The question is if increasing only the state size improves the security. In advance of the design of *Enocoro*, we examined the relation between the size of the buffer the resistance against the linear distinguishing attack by using a toy cipher of MUGI whose internal state consists of 8-bit blocks

instead of 64-bit blocks, and the F-function is replaced by an 8-by-8 Sbox. In addition, the positions of feedbacks are parameterized as is the case with *Enocoro*.

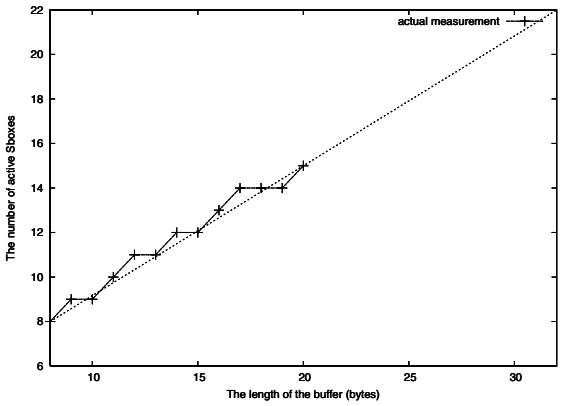

Figure 4. The resistance of MUGI's toy cipher against linear distinguishing attack

Figure 4 shows the maximum value of the resistant property of the toy cipher against the linear distinguishing attack according to the buffer size, where the resistance is expressed by the number of active Sboxes included in the linear path. The solid line shows the experimental results according to the buffer size $n_b = 8$ to 20. The dotted line means the estimate from the experimental result. The experimental result indicates that the resistant property increases in proportion to the buffer size. This result encouraged us to design a family of PRNGs.

6.4 Choice of Parameters

Checking all possible parameters is difficult, so that the parameters specifying the algorithm is chosen from which satisfy the following conditions:

$$q_i = k_i \quad \text{for} \quad 1 \leq i \leq 3,$$
$$p_i = k_{i+1} - 1 \quad \text{for} \quad 1 \leq i \leq 3.$$

Under the conditions, the inputs from the buffer to the function ρ specifies the feedbacks of the function λ.

Mostly considered security is the resistance against linear distinguishing attack. All possible values for k_1, k_2, k_3, k_4 are examined in the evaluation and one of the best set of parameters are selected.

7 Conclusion

In this paper we are attempted to design a new hardware oriented PRNG whose structure is similar to PANAMA and a family of PRNGs *Enocoro* is proposed. The 80-bit key algorithm *Enocoro*-80 can be implemented with 2.7 Kgates and it is comparable to eSTREAM Profile 2 candidates. In addition, their software implementations in C language achieves better performance than most of the candidates. As a result, we confirm that the design of PANAMA is suitable to not only a software oriented cipher, but also a hardware oriented cipher.

8 Acknowledgement

This work was supported in part by a consignment research from the National Institute on Information and Communication Technology (NICT), Japan.

References

[1] S. H. Babbage, "Improved exhaustive search attacks on stream ciphers," *European Convention on Security and Detection*, IEEE Conference publication No. 408, pp. 161–166, 1995.

[2] S. Babbage and M. Dodd, "The stream cipher MICKEY-128 2.0," available at `http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey128_p3.pdf`

[3] E. Biham, A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard," Springer-Verlag, 1993

[4] D. Bleichenbacher and S. Patel, "SOBER cryptanalysis," *Fast Software Encryption, FSE'99*, Springer-Verlag, LNCS 1636, pp. 305–316, 1999.

[5] J. Daemen, C. Clapp, "Fast Hashing and Stream Encryption with PANAMA," *Fast Software Encryption, FSE'98*, Springer-Verlag, LNCS 1372, pp.60–74, 1998.

[6] J. Daemen, R. Govaerts, J. Vandewalle, "Resynchronization weaknesses in synchronous stream ciphers," *Advances in Cryptology, Proceedings Eurocrypt'93*, Springer-Verlag, LNCS 765, pp. 159–169, 1994.

[7] eSTREAM, –The ECRYPT Stream Cipher Project–, `http://www.ecrypt.eu.org/stream/`.

[8] eSTREAM PHASE 3 Performance Figures Intel Pentium 4 revision 206, `http://www.ecrypt.eu.org/stream/phase3perf/2007a/pentium-4-a/`.

[9] FIPS 197, "Advanced Encryption Standard," National Institute of Standards and Technology, 2001. Available at `http://www.itl.nist.gov/fipspubs/`.

[10] H. Furuichi, K. Muto, D. Watanabe, T. Kaneko, "Security evaluation of Enocoro-80 against differential resynchronization attack," *The 2008 Symposium on Cryptography and Information Security, SCIS 2008*, 4A1-3, 2008 (*in Japanese*).

[11] I. Golić, "Cryptanalysis of alleged A5 stream cipher," *Advances in Cryptology, Eurocrypt'97*, Springer-Verlag, LNCS 1233, pp. 239–255, 1997.

[12] T. Good and M. Benaissa, "Hardware results for selected stream cipher candidates," available at `http://www.ecrypt.eu.org/stream/papersdir/2007/023.pdf`

[13] M. Hell, T. Johansson and W. Meier, "A Stream Cipher Proposal: Grain-128," available at `http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain128_p3.pdf`

[14] K. Ideguchi and D. Watanabe, "A Method of Security Evaluation of Guess and Determine Attacks," *The 2008 Symposium on Cryptography and Information Security, SCIS 2008*, 3A1-4, 2008 (*in Japanese*).

[15] M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology, Eurocrypt'93*, Springer-Verlag, LNCS 765, pp. 159–169, 1994.

[16] K. Muto, D. Watanabe, T. Kaneko, "A Study on Strength of Pseudorandom Number Generator Enocoro-80 against Linear Distinguish Attack," *The 30th Symposium on Information Theory and Its Application, SITA 2007*, 2007 (*in Japanese*).

[17] K. Muto, D. Watanabe, T. Kaneko, "Security evaluation of Enocoro-80 against linear resynchronization attack," *The 2008 Symposium on Cryptography and Information Security, SCIS 2008*, 4A1-2, 2008 (*in Japanese*).

[18] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," *Advances in Cryptology, ASIACRYPT 2001*, Springer-Verlag, LNCS 2249, pp. 230–254, 2001.

[19] H. Sekine, T. Nosaka, Y. Hatano, M. Takeda, and T. Kaneko, "A strength evaluation of a pseudorandom number generator MUGI against linear cryptanalysis," *IEICE Trans.* Vol. E88-A, No. 1, pp. 16-24, January 2005.

[20] D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi, and B. Preneel, "A new keystream generator MUGI," *Fast Software Encryption, FSE'02*, Springer-Verlag, LNCS 2365, pp. 179–194, 2002.

[21] D. Watanabe and T. Kaneko, "Some experiments on the mini-model of the pseudorandom number generator MUGI," *IEICE Technical Report*, ISEC2007-2, pp. 11-16, 2007 (*In Japanese*).

[22] D. Watanabe and T. Kaneko, "A construction of light weight Panama-like keystream generator," *IEICE Technical Report*, ISEC2007-78, pp. 33-40, 2007 (*In Japanese*).

A Test Vectors

$\mathrm{key}[10] = \{0\}$
$\mathrm{iv}[8] = \{0\}$
output =
`0xc9 0x22 0x79 0x45 0x6e 0xbe 0x3b 0xff`
`0xd8 0xd4 0x73 0x12 0x3e 0xce 0xb9 0x57`
...

$\mathrm{key}[10] =$
$\{$`0x00`, `0x01`, `0x02`, `0x03`, `0x04`, `0x05`, `0x06`, `0x07`, `0x08`, `0x09`$\}$
$\mathrm{iv}[8] =$
$\{$`0x00`, `0x10`, `0x20`, `0x30`, `0x40`, `0x50`, `0x60`, `0x70`$\}$
output =
`0x9b 0x0a 0x97 0x39 0x4b 0x58 0x72 0x73`
`0x3d 0xbf 0x9e 0xe5 0x0c 0x33 0x73 0x3e`
...