Stream Cipher *Enocoro*

Evaluation Report

Hitachi, Ltd.

2 February 2010

Contents

1 Introduction

This document is the evaluation report of a family of stream cipher *Enocoro*. A pseudorandom number generator, which underlies the stream cipher, characterizes the security and the implementation features of the stream cipher. Hereafter, we mainly analyze those properties of the pseudorandom number generator underlying the stream cipher *Enocoro*.

As is often the case with other cryptographic primitives, a pseudorandom number generator is considered reliable only if it is thoroughly evaluated both in terms of security and performance. This document purposes to provide our own evaluation results and other known results in order to claim that *Enocoro* is a "good cryptographic primitive". In addition, we wish that this document is a good introduction to researchers who intend to study *Enocoro*.

The document is organized as follows: Firstly, we will explain the design strategy of *Enocoro* in Section 2. Next, we survey the security evaluation results on *Enocoro* in Section 3. After that the hardware and software implementation aspects of *Enocoro*-128v2 are introduced in Section 4. Finally, we compare *Enocoro*-128v2 to other symmetric key primitives in Section 5.

Note that the readers are assumed to have sufficient knowledge on the specification of *Enocoro*-128v2. In addition, we use the notations defined in the specification document [20] and the terminologies which are familiar for symmetric key cryptographer without explanations. Please refer to the bibliographies for these information.

2 Design Rationale

The design of *Enocoro* intends to provide a light-weight symmetric key encryption in hardware. The origin of *Enocoro* is PANAMA proposed by Daemen and Clapp [7]. PANAMA adopts word-wise operations in order to achieve good performances in 32-bit platforms. Despite that *Enocoro* is intended to be a hardware oriented cipher, we also consider that its software implementations should not be ignored. In order to achieve two different requirements, we adopted byte-wise operations. This is a significant difference from other hardware oriented stream cipher such as ones listed in eSTREAM portfolio [9].

Another good point in adoption of a byte-wise design is that the security evaluation is much easier than that for a bit-wise design. Consider a 128-bit key cipher whose internal state is more than 256 bits. It is computationally infeasible to evaluate the resistance against various attacks by trying 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. There is a generic technique to simplify the various security evaluations for the block-wise structure, which is called a truncated evaluation. Especially, the truncated evaluations for 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.

In the choices of the underlying basic primitives such as an Sbox, we give preference the implementation cost over the cryptographic strength.

2.1 Internal State

In order to achieve the *light-weightness* in hardware implementation, we chose the size of the internal state of *Enocoro*-128v2 as small as possible. As a well known fact, a 128-bit key stream cipher must have at least 256 bits state in order to prevent the Time-Memory-Trade-Off attack. Therefore we chose $n_b = 32$, which means that the size of the internal state of *Enocoro*-128v2 is 272 bits.

2.2 Sbox

Enocoro adopts 8-bit Sbox as the only non-linear component. Different from AES [11] and MUGI [19], the Sbox consists of four 4-bit Sboxes and a 2 by 2 matrix over GF(2^4). This approach does not provide the best Sbox in terms of probabilistic properties such as the maximum differential probability and the maximum linear probability. On the other hand, the implementation cost (in hardware) of our Sbox is smaller than using the inverse mapping.

2.3 Structure of The Function ρ

PANAMA and MUGI adopted the SPN (Substitution-Permutation Network) in their non-linear transformations ρ. The ρ function of *Enocoro* has a different structure. It adopted the deformed SPS (Substitution-Permutation-Substitution) network in order to improve its non-linear property. The deformation enables the parallel processing of four Sboxes in hardware implementations.

2.4 Choice of Parameters

The algorithm of *Enocoro* has 11 parameters and they (except n_b which decides the size of internal state) decide the wire connections in the update function. These parameters for *Enocoro*-128v2 are the same as that for *Enocoro*-128v1.1 and they are chosen to optimize the security against the linear distinguishing attack.

2.5 Versions and Differences

Several recommended sets of parameters have been published to specify the algorithms of *Enocoro*. Though this document basically deals with *Enocoro*-128v2, we think that it is convenient for readers to summarize the published sets of parameters and to clarify their differences.

Table 1 shows the choices of parameters and other differences between the algorithms. All sets of parameters which have been published satisfy the following relations:

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

Therefore p_i and q_i are unique if $k_1, \ldots, k_4$ are given. In Table 1 we show only four parameters n_b, k_1, k_2, k_3, k_4.

Table 1: Published parameters of *Enocoro*

Key length (bits)	Version	Def. poly. of GF(2^8)	Parameters	Initialization	
			$(n_b; k_1, k_2, k_3, k_4)$	# of rounds	Counter
80	1.0 [30]	$x^8 + x^4 + x^3 + x + 1$	$20; 1, 4,\ 6, 16$	40	wo
128	1.0 [30]	$x^8 + x^4 + x^3 + x + 1$	$32; 1, 5, 15, 29$	64	wo
	1.1 [27]	$x^8 + x^4 + x^3 + x + 1$	$32; 2, 7, 16, 29$	64	wo
	2.0	$x^8 + x^4 + x^3 + x^2 + 1$	$32; 2, 7, 16, 29$	96	w

3 Security Evaluations

3.1 Basic Properties of Underlying Functions

The cryptographic properties of the 8-bit Sbox s_8 is summarized as follows:

- Maximum differential probability: $2^{-4.678}$
- Maximum linear probability: 2^{-4}
- Algebraic degree: 6

The branch number of the linear transform L is 3.

3.2 Time-Memory-Trade-Off Attack

A time-memory-trade-off (TMTO) attack on stream ciphers is firstly proposed by Babbage and Golić independently [1, 12]. Their attack indicates a new insight about the minimum requirement on the size of the internal state. The cost of a TMTO attack on stream ciphers is at least $O(2^{n/2})$, where n is the size of the internal state of the stream cipher. Therefore a stream cipher, whose key length is 128-bit, must have at least 256 bits internal state. *Enocoro*-128v2 has a 272 bits internal state so that the complexity of the naive TMTO attack is $2^{272/2} = 2^{136}$. Hence we believe that TMTO attacks do not threaten the security of *Enocoro*-128v2.

3.3 Guess and Determine Attack

A guess and determine attack (GD-attack) was proposed by Bleichenbacher 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 of the internal state at the beginning of the attack, then determines other words by using the relations coming from the update function and the output function. The output string tends to reveal some information on the internal state, this strategy provides a better attack than the exhaustive search of the internal state in general.

Ideguchi and Watanabe developed an evaluation tool that randomly chooses the guess words and iterates determine processes [22]. The best attack found by their tool for *Enocoro*-128v1.1 needs to guess 18 bytes (144 bits) so that the time complexity of the attack is about 2^{144}. The update function of *Enocoro*-128v2 is the same as that of *Enocoro*-128v1.1 so that their result is also valid for *Enocoro*-128v2. Therefore we believe that GD-attacks do not threaten the security of *Enocoro*-128v2.

3.4 Divide and Conquer Attack

A divide-and-conquer attack can be used if the internal state of the pseudo-random number generator can be divided into several parts, and their update functions do not take the other states as inputs. In other words, the attack is applicable if the update function *Next* can be written in the following form:

$$(a_1^{(t+1)}, \dots, a_N^{(t+1)}) = S^{(t+1)} = \mathit{Next}(S^{(t)}) = (f_1(a_1^{(t+1)}), \dots, f_N(a_N^{(t+1)})).$$

By definition of a PKSG, the internal state S is divided to two registers a and b. However, both their update functions ρ and λ take the other register as an additional input:

$$\begin{aligned} a^{(t+1)} &= \rho(a^{(t)}, b^{(t)}), \\ b^{(t+1)} &= \lambda(a^{(t)}, b^{(t)}). \end{aligned}$$

Therefore, we think that it is difficult to apply divide-and-conquer attacks to *Enocoro*-128v2.

3.5 Algebraic Attacks

In this decade, algebraic approaches to attack stream ciphers are highlighted in the security evaluation of stream ciphers. The XLS attack [6], and more generally, the application of Gröbner basis theory are basically the technique to reduce the complexity to solve the system of algebraic equations which describe the relation of the internal state and the output bits. However, these techniques have been mostly applied to stream ciphers based on linear feedback shift registers. The reason is that the number of monomials tends to explode if the update function is non-linear. We think this kind of attacks is not efficient because the update function of *Enocoro* is a non-linear function.

3.6 Linear Distinguishing Attack

A linear distinguishing attack is a kind of theoretical randomness testing of the output sequence which is an application of the linear attack [15] to stream ciphers. The efficient technique to evaluate the resistance against the linear distinguishing attack on the word-oriented stream ciphers was proposed by Sekine *et al.* [17]. So far, Muto *et al.* applied Sekine's approach in order to evaluate *Enocoro*-128v1.1 [27] and they concluded that at least 2^{144} outputs are necessary to detect the non-linear property of the output of *Enocoro*-128v1.1. The update function of *Enocoro*-128v2 is the same as that of *Enocoro*-128v1.1 so that their result is also valid for *Enocoro*-128v2. Therefore we believe that LDAs do not threaten the security of *Enocoro*-128v2.

3.7 Re-synchronization Attacks

A re-synchronization attack was proposed by Daemen *et al.* [8]. It is an generic framework of the attack in which the attacker concentrate on finding the weakness of the initialization function *Init*. Here we introduce the known results which studied the security of the initialization process of *Enocoro*, including *Enocoro*-80 and *Enocoro*-128v1.1.

The first research was done by Muto *et al.* [27]. They considered the application of the linear attack to the initialization process and studied the security of *Enocoro*-80 by the truncated evaluation. They reported that the distinguishing attack requires at least $2^{1}44$ known IVs.

Konosu *et al.* applied the same evaluation approach to *Enocoro*-128v1.1 [24]. They considered not only linear attack, but also differential attack. They concluded that the differential distinguishing attack on the initialization process requires $2^{177.8}$ chosen IVs and the linear attack requires 2^{216} known IVs.

Okamoto *et al.* pointed out that the key recovery techniques used in the attack on block ciphers, which have been used to reduce the data complexity, is applicable to the re-synchronization attacks [28]. They studied the differential attack using their idea to evaluate *Enocoro*-80 and concluded that at least $2^{70.2}$ chosen IVs are necessary for the attack.

We applied Okamoto's technique to evaluate the security of *Enocoro*-128v1.1 and v2 against the differential attack. We concluded that the attack on *Enocoro*-128v1.1 (64 rounds) requires at least $2^{102.9}$ chosen IVs [31]. Note that the specification of *Enocoro*-128v1.1 (and v2) do not allow to choose more than 2^{64} IVs. In addition, Okamoto's approach requires to guess a part of the secret key and the time complexity of the attack is more than 2^{128}. Therefore, this result does not threaten the security of *Enocoro*-128v1.1 nor that of v2.

Besides, we studied the case that the update function is applied 96 times in the initialization (Namely, the *Init* in *Enocoro*-128v2) and we confirmed that at least $2^{177.8}$ chosen IVs are necessary to attack in this case. Therefore we believe that the initialization process of *Enocoro*-128v2 is secure against the differential attack even if the there are advances in the key recovery techniques.

3.8 Weak Key and Related Key Attack

We will introduce a related key attack on *Enocoro*-128v1.1 proposed in [31]. This is the main reason why we updated *Enocoro*-128v1.1 to v2. The attack is applicable to 2^{112} out of 2^{128} keys and the complexity of the attack to recover a full 128-bit key is 2^{65}. It should be pointed out that the key pair used in the attack is hardly practical. Then we will explain the patch, which is one of the differences between *Enocoro*-128v1.1 and v2, and the reason that it prevents the attack.

3.8.1 Slid Pair

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 $S(K,I)^{(0)}$ 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 T-slid pair if $S(K,I)^{(0)}$ is equal to $S(K',I')^{(T)}$ at a round T. Obviously, keystreams generated by a slid pair satisfy $z(K,I)^{(t)} = z(K',I')^{(t+T)}$ for any t, where $z(K,I)^{(t)}$ is the output byte at time t, which is generated by the key K and the IV I.

It is easy to find a slid pair if the update function used in the random number generation and one used in the initialization function are the same, and *Enocoro*-80 and *Enocoro*-128v1.1 are the cases. We pointed out this property for *Enocoro*-80 and estimated that about 2^{32} trials are necessary to find a T-slid pair [18].

Enocoro-128v1.1 has the same feature as *Enocoro*-80. Let the initial inputs be (K,I) and T be an arbitrary positive integer. If values of $b_{24},\ldots,b_{31}$ and a_0,a_1 at round $T-96$ coincide to the initial constants $C_0,\ldots,C_9$, then (K,I) and (K',I'), where $K' = (b_0^{(T-96)},\ldots,b_{15}^{(T-96)})$ and $I' = (b_{16}^{(T-96)},\ldots,b_{23}^{(T-96)})$, are the T-slid pair. Therefore at least a T-slid pair is found with probability 2^{-80} for any T if the attacker randomly chooses K and I.

3.8.2 Weak Key of *Enocoro*-128v1.1

In the following, we explain the related key attack on *Enocoro*-128v1.1. The attack firstly checks if the target key is a weak key, then recovers it. The main observation is that, if the attacker chooses $C_0 \oplus C_5, C_1, \ldots, C_7$ as the IV values $I_0, I_1, \ldots, I_7$ and applies 8 round functions, then $b_{24},\ldots,b_{31}$ are equal to $C_0,\ldots,C_7$. Therefore the probability to find a 8-slid pair is expected to be 2^{-16}.

In our attack, the related key K' corresponding to K is given by $(K',I',C') = \mathit{Next}^8(K,I,C)$. Note that this relation is much less practical than the "key difference" scenario. For such key pairs, the key K is a weak key if C is equal to $C' = C'(K,I)$. The following algorithm shows the procedure to detect the weak key and to recover the secret key K.

In Algorithm 1, 64 bits value X is guessed in Step 1 and 8 bytes value

Algorithm 1 Algorithm to detect the weak key of *Enocoro*-128v1.1

```
Step 1. The attacker guesses the value of $I'$. Let the guessed value be $X$.
Step 2. He compares the keystream generated by two distinct inputs $(K, I)$ and $(K', X)$. If $(K, I)$ and $(K', X)$ are not the 8-slid pair, go back to Step 1.
Step 3: If they are the 8-slid pair, it is expected that the guess is correct. Then he recovers the 8 key bytes from $I'$ by using the relations $K_8 = I'_0 \oplus C_0, \ldots, K_{15} = I'_7 \oplus C_7$
Step 4: He recovers $K_0, \ldots, K_7$ by the exhaustive search.
```

$K_0, \ldots, K_7$ are guessed in Step 4, therefore the total time complexity of the attack is about 2^{65}.

3.8.3 Application of The Related Key Attack on *Enocoro*-128v2

The dominant reason for the vulnerability is that the initialization *aeinit* and the update function *aenext* use the same round function. The simple and efficient countermeasure is to attach a counter to the update function [5]. The update function for the initialization of *Enocoro*-128v2 has an 8-bit counter. We think that this change make it difficult to find a slid pair.

4 Implementation Aspects

4.1 Hardware Implementation

In this section, we describe our results on hardware implementation of *Enocoro.*

In a basic form, a *Enocoro* hardware circuit processes one round per clock cycle and consists of a 2-byte register for the state a, a 32-byte register for b, and combinational circuits both for the λ function and for the ρ function. Hereafter, we call this basic implementation.

We think that this basic implementation is highly efficient and well-balanced. Considering that the 8-bit Sbox s_8 consists of several applications of the 4-bit Sbox s_4, there can be two approaches of implementing s_8 in hardware: one is to have a 8-bit table in circuit and the other is to have a combinational circuit which performs logical operations. By taking the former approach, one can achieve the hardware implementation with a larger amount of gate counts and shorter gate delay. By taking the latter approach, one can achieve the hardware implementation with a smaller amount of gate counts and longer gate delay.

In order to reduce the number of gate counts for the basic implementation of *Enocoro*, it is effective to share S-boxes in the ρ function. The drawback in sharing the S-box circuit is the decrease of the throughput, which is caused by the fact that it takes more clocks to perform the same operations. For instance, if two or even one S-boxes is implemented, then the throughput is twice or four times lower than that of the basic implementation which implements 4 S-boxes.

On the other hand, in order to improve the throughput for the basic implementation of *Enocoro*, it is effective to sequentially have several rounds of the λ function and the ρ function in circuit. With this approach, one can achieve a high-throughput implementation of *Enocoro* which takes only one clock to process several rounds. For instance, if one has 4 rounds of λ function and ρ function in circuit, one can achieve a fast implementation of *Enocoro* which takes only one clock to generate 32 bits of pseudorandom numbers. However, the improvement on throughput is limited because of the increase of gate delay which makes the frequency lower.

In the following, we show the figures of our hardware implementation of *Enocoro*-128v2. The implementation architecture is the basic implementation where the two approaches are considered: One implementation has the Sbox in a table and the other has it in logical operations. We show our result on the ASIC implementation of *Enocoro*-128v2 with 90 nm standard library. The number of gate counts is 2-NAND gate equivalent. We show our result of FPGA implementations *Enocoro*-128v2 on ALTERA Stratix II. The number of slices is measured by converting the number of Adaptive Logic Module (ALM) into the number of Logic Element (LE) which is commonly used in the case of FPGA implementations. For each implementation, we adaptively set synthesis conditions so that one can take advantage of the characteristic each implementation has.

As shown the above, *Enocoro* has a characteristic that it is possible to implement it with a small number of gate counts.

Another characteristic of *Enocoro* is that its hardware module consumes low electric power per one-bit processing. In [23], there are power consump-

Table 2: ASIC implementation result of *Enocoro*-128v2

Sbox implementation approach	Synthesis condition	Gate count (Kgates)	Frequency (MHz)	Throughput (Gbps)
Table lookup	speed-optimised	8.7	1,250	10.0
Finite field operations	speed-optimised	4.7	1,052	8.4
	area-optimised	4.1	440	3.5

Table 3: FPGA implementation result of *Enocoro*-128v2

Sbox implementation approach	synthesis condition	gate count (LE)	Frequency (MHz)	Throughput (Gbps)
Table lookup	speed-optimised	795	149	1.2
Finite field operations	speed-optimised	530	140	1.1
	area-optimised	525	118	0.9

tion evaluations shown as follows: the power consumption during the encryption process of *Enocoro*-128v1.0 is 6.77 mW and the electrical energy per one-bit processing is 0.103 nW. These figures are less than 1/10 and 1/2 than the cases of AES and MUGI respectively.

Note that these results are on *Enocoro*-128v1.0. However, we expect that the results on *Enocoro*-128 v2 would not be very different from those of *Enocoro*-128v1.0 because the difference is caused by the more cycles spent on the initialization process in *Enocoro*-128v2.

4.2 Software Implementation

Here, we show the software performance of the *Enocoro*-128v2 on the generally available platform. Our target processor model is the Intel Core2 Duo E6600 2.4 GHz processor in 32-bit mode. Table 4 shows the detailed information of the evaluation environment.

We have measured the speed performance of a C code. We have used `gcc` compiler and `-O3` option. The code size was 974 bytes and the work area was 256 bytes.

Table 5 shows the speeds in processing to generate data of various lengths.

Table 4: Platform used for measurement

CPU	Memory	OS	Compiler
Intel Core2Duo E6600 (2.4 GHz)	2 GB	Ubuntu Linux 8.04 32-bit distribution	gcc 4.2.4

Table 5: Throughput speeds and execution time for the initialization

	Data size		
	16	256	1M
Speed (cycle/Byte)	318.6	64.9	46.3

The clocking of the execution time includes the initialization process, which takes 4870 cycles.

For each data length, we measured the speeds for 100 times. The results shown in Table 5 are the average values. In order to measure the execution time, we used the TSC register. The following code is the example for the measurement of the execution time of `init()` and `keystream()`.

```
#define RDTSC(X) asm volatile("rdtsc" : "=A" (X))
RDTSC(X);
init(&state, key, iv);
keystream(&state, out, ENOCORO_OUT_SIZE);
RDTSC(Y);
```

4.3 Resistance to Side-channel Attacks

The hardware module of *Enocoro* has a characteristic that it can process with low power consumption, compared to the AES module.

The side channel attacks such as DPA and CPA exploit biases varying power consumption of the target hardware while performing operations using secret keys. It might be more difficult to statistically analyze power consumption measurements for *Enocoro* than for AES. It is known that one can easily recover the key for known block ciphers like AES by guessing how intermediate values make a transition at some byte in the register from the last round to the ciphertext.

On the other hand, unlike the case of AES, it is difficult to apply the same approach to the basic implementation of *Enocoro* which generates one byte of pseudorandom number per cycle. It is known that it is effective to take some countermeasures for S-box to prevent power analysis in the case of block ciphers like AES. In the case *Enocoro* which has four Sbox applications per round, it might be possible to apply this approach with a lower cost than that of AES which has 16 Sbox applications per round.

5 Comparison to Preceding Algorithms

In this section we will compare *Enocoro*-128v2 and some other symmetric key ciphers listed in the e-Government Recommended Ciphers List and the stream ciphers in eSTREAM portfolio [9].

5.1 Security

The discussion of the merits of two different algorithms in terms of security is not a simple task because each algorithm has its own features. A novel idea to attack a new algorithm often arises from these differences. We think it is not realistic to compare the security of *Enocoro*-128v2 to the other known algorithms.

On the other hand, there is no serious vulnerability has been reported since *Enocoro* was published in 2007. In addition, we patched *Enocoro*-128v1.1 in order to prevent all possible weaknesses which have been suggested. We believe that the security of *Enocoro*-128v2 is comparably high to the other 128-bit ciphers.

5.2 Software Performances

The design of *Enocoro* is optimized to a small hardware implementation. We think that *Enocoro* does not have a concrete advantage against the other CRYPTREC ciphers.

5.3 Hardware Performances

Table 6 shows the ASIC implementation results. For the comparison, we chose AES [11] and MUGI [19] from CRYPTREC ciphers and Grain [14] and MICKEY [2] from eSTREAM Profile 2 ciphers, that have algorithms for a 128-bit key. The result on AES is referred from [16]. The results on eSTREAM ciphers are referred from the technical report by Good and Benaissa [13]

Table 6: The comparison of hardware implementations

Algorithm	Max. clock freq. (MHz)	Area (Kgates)	Throughput (Mbps)	Process (μm)
AES [16]	131.2	5.4	311	0.11
MUGI [29]	51.1	22.7	1,600	0.18
	186.2	46.0	11,900	
Grain-128 [13]	925.9	1.9	926	0.13
	581.4	2.5	4,651	
MICKEY-128 [13]	413.2	5.0	413	0.13
Enocoro-128v2	440.0	4.1	3,520	0.09

Table 6 shows that the implementation cost of *Enocoro*-128v2 is smaller than that of AES and MUGI. In addition, it is faster than AES. By comparison to eSTREAM portfolio ciphers, the cost of *Enocoro*-128v2 is larger than Grain-128 and smaller than MICKEY-128. The speed is also between Grain-128 and MICKEY-128.

Trademarks

- Altera and Stratix are registered trademarks of Altera Corporation in the United States and/or other countries.
- *Enocoro* is a registered trademark of Hitachi, Ltd. in Japan.
- Intel is a registered trademark and Core is the name of products of Intel Corporation in the U.S. and other countries.

- Linux is a registered trademark of Linus Torvalds in the U.S. and other countries.
- Ubuntu is a registered trademark of Canonical Ltd. in the United States and/or other countries.

Acknowledgements

This work was supported in part by a consignment research from the National Institute on Information and Communication Technology (NICT), Japan. In addition, most of the security evaluations have been done by Hiroki Furuichi, Toshinobu Kaneko, Kei Konosu, Kenichiro Muto, and Kazuto Okamoto of Science University of Tokyo. We would like to thank for their consecutive contributions.

References

[1] S. H. Babbage, “Improved exhaustive search attacks on stream ciphers,” *European Convention on Security and Detection*, IEE 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] M. Bellare and T. Kohno, “A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications,” *Advances in Cryptology–Eurocrypt 2003*, Lecture Notes in Computer Science, Vol. 2656, pp. 491–506, Springer-Verlag, 2003.

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

[5] C. De Cannière, Ö Küçük, and B. Preneel, “Analysis of Grain’s Initialization Algorithm,” *The State of the Art of Stream Ciphers, SASC 2008*, pp. 43–56, 2008.

[6] N. Courtois and J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations," `Advances in Cryptology, Asiacrypt'02`, Springer-Verlag, LNCS 2501, pp. 267–287, 2002.

[7] J. Daemen and C. Clapp, "Fast Hashing and Stream Encryption with PANAMA," *Fast Software Encryption, FSE'98*, Springer-Verlag, LNCS1372, pp. 60–74, 1998.

[8] 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.

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

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

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

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

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

[14] 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`

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

[16] 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.

[17] 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.

[18] D. Watanabe, K. Ideguchi, J. Kitahara, K. Muto, H. Furuichi, T. Kaneko, “Enocoro-80: A Hardware Oriented Stream Cipher,” *Second International Workshop on Advances in Information Security*, 2008.

[19] Hitachi, Ltd., “MUGI Pseudorandom Number Generator: Specification Ver. 1.2.” Available at `http://www.sdl.hitachi.co.jp/crypto/mugi/`.

[20] Hitachi, Ltd., “Pseudorandom Number Generator *Enocoro*: Specification Ver. 2.0.” Available at `http://www.sdl.hitachi.co.jp/crypto/enocoro/`.

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

[22] 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, January 2008 (*in Japanese*).

[23] J. Kitahara and D. Watanabe, “An electrical power evaluation of stream cipher Enocoro which is implemented in hardware,” *The 2008 Symposium on Cryptography and Information Security, SCIS 2008*, 2C2-3, January 2008 (*in Japanese*).

[24] K. Konosu, K. Muto, H. Furuichi, D. Watanabe and T. Kaneko, “Security evaluation of *Enocoro*-128 ver.1.1 against resynchronization attack,” IEICE Technical Report, ISEC2007-147, 2008 (*in Japanese*).

[25] K. Muto, D. Watanabe, and T. Kaneko, “A Study on Strength of Pseudorandom Number Generator *Enocoro*-80 against Linear Distinguish Attack,” *SITA2007*, 2007 (*in Japanese*).

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

[27] K. Muto, D. Watanabe, and T. Kaneko, “Security evaluation of *Enocoro*-80 against linear resynchronization attack,” *The 2008 Symposium on Cryptography and Information Security, SCIS 2008*, 4A1-1, January 2008 (*in Japanese*).

[28] K. Okamoto, K. Muto, and T. Kaneko, “Security evaluation of Pseudorandom Number Generator *Enocoro*-80 against Differential/Linear Cryptanalysis (II),” *The 2009 Symposium on Cryptography and Information Security, SCIS 2009*, 4B2-3, 2009 (*in Japanese*).

[29] T. Owada, S. Taira, Y. Igarashi, and J. Kitahara, “Hardware Implementation and Evaluation of MUGI,” *The 2005 Symposium on Cryptography and Information Security, SCIS 2005*, 1A3-5, 2005 (*in Japanese*).

[30] D. Watanabe and T. Kaneko, “A construction of light weight Panama-like keystream generator,” IEICE Technical Report, ISEC2007-78, 2007 (*in Japanese*).

[31] D. Watanabe, K. Okamoto, T. Kaneko, “A Hardware-Oriented Light Weight Pseudo-Random Number Generator *Enocoro*-128v2,” *The 2010 Symposium on Cryptography and Information Security, SCIS 2010*, 3D1-3, 2010 (*in Japanese*).