Self-Evaluation Report
MULTI-S01
(revised for 2001 submittion)

Hitachi, Ltd.

1. Introduction

This article summarizes the results of the research done to evaluate security of the MULTI-S01 encryption algorithm.

MULTI-S01 uses the pseudorandom number generator (PRNG) PANAMA. Thus, the security of MULTI-S01 is based on that of PANAMA. In this article, we discuss three topics.

1. Security of MULTI-S01 with assumption of security by PANAMA;
   - ✓ Data Confidentiality
   - ✓ Data Integrity
2. Security of PANAMA PRNG;
   - ✓ Empirical Randomness Test
   - ✓ Cryptographic Security Evaluation
3. Implementation Evaluation (i.e., performance)
   - ✓ Software Implementation
   - ✓ Hardware Implementation

These results show that MULTI-S01 is a secure and efficient encryption algorithm.

The latest information about MULTI-S01 is available at the following URL at any time:

http://www.sdl.hitachi.co.jp/crypto/s01/index.html

2. Data Confidentiality and Data Integrity

2.1. Introduction

In this section, we investigate the security of MULTI-S01, assuming the security of PANAMA. MULTI-S01 supplies two security fanctionalities as a symmetric key cipher.

□ Message Confidentiality

Cryptographic characteristics prevent anyone without the secret key from deriving plaintext information from ciphertext.

□ Message Integrity

Cryptographic characteristics prevent anyone without the secret key from generating a *valid* alteration of the ciphertext. *Valid* alteration means generating a different ciphertext whose plaintext passes the receiver's verification process.

Approaches based on currently known techniques

We briefly review currently known methods to achieve confidentiality and integrity. There are two distinct ways to provide these two securities: symmetric key cryptography and asymmetric key cryptography.

Table 1: Cryptographic Techniques for Achieving Confidentiality and Integrity

	Confidentiality	Integrity
Symmetric-key	Block cipher Stream cipher	Message authentication code (MAC)
Asymmetric-key	Public-key cipher	Digital signature

In this article, we discuss only symmetric-key cryptography and do not mention techniques based on asymmetric-key cryptography.

Block ciphers and stream ciphers are well known ways to provide data confidentiality. Six major modes of a block cipher, ECB, CBC, CFB, OFB, PCBC, and counter mode, are used only for data confidentiality. There are two distinct types of stream ciphers, synchronous and asynchronous, both of which provide only data confidentiality as well.

Data integrity is provided by making use of a message authentication technique. Generating a message authentication code (MAC) is one message authentication technique. There are several known ways to generate MACs from cryptographic primitives; DES-MAC (CBC-MAC is the generic term), MMH, and HMAC are based on the security of a block cipher, a PRNG, and both of a PRNG and hash function, respectively.

In many applications, encryption is expected to provide a security channel where transmitted data are secured from tapping or malicious alteration. In such circumstances, we need both data confidentiality and data integrity. A simple solution is to combine two techniques; one for confidentiality and one for integrity. However, a simple combination does not work well.

One disadvantage is an increase in the key length because the keys for two mechanisms should be independent of each other. Otherwise, generally, using just one key between two mechanisms, confidentiality and integrity, does not guarantee the security of both, even if the mechanisms provide very good security independently. For instance, CBC mode and CBC-MAC do not have any serious flaws on their own, but when they are used under the same key, the integrity guaranteed by CBC-MAC is obviously violated. The two mechanisms must therefore use independent keys. This increases the length of the shared secrets, i.e., the key length.

Moreover, in some circumstance, combined encryption may degrade performance for a long message. If encryption and MAC generation are processed separately, the whole message must be stored. Otherwise, two processes, one for encryption and one for MAC generation, must run at the same time, resulting in more memory consumption.

Another disadvantage of implementing both encryption and MAC generation, is an increase in the implementation cost (i.e., program size and gate count) without a particular idea of sharing resources between encryption and MAC generation.

There are several known methods for providing both securities. Two block ciphers with a variable block length, BEAR and LION, use both PRNG and a hash function. However, BEAR and LION essentially require the same amount of memory as needed for plaintext. For a large amount of data, BEAR and LION are not efficient.

Two other modes of operation, iaPCBC and RPC, are message-authenticating encryption. With iaPCBC mode, a block cipher is processed sequentially so that there is less parallel computation. RPC is highly suitable for parallel computation, but the length of the ciphertext increases proportionally. For a large ciphertext, RPC is not efficient without parallel computation.

We have evaluated the security and efficiency of a message-authenticating stream cipher, in which parallel computation and pre-computation are applicable.

One of the biggest advantages of the proposed cipher is that it guarantees two securities simultaneously; confidentiality and integrity. Moreover our another objective of design is to design an encryption with highly parallel computation. We briefly describe our approaches for our goals, data integrity and parallel computation.

Message Integrity

Similar to CBC mode, the mechanism we designed randomizes data in blocks and feedbacks to the next block. A randomized difference thus propagates to the last block if

someone modifies the ciphertext.

Performance

To take advantage of parallel calculation and pre-computation of the stream cipher, we designed a mechanism so that the PRNG is independent of the plaintext and ciphertext.

In this article we describe the details of our proposed scheme.

2.2. Encryption Model

In this section, we define our encryption model. The procedure of the proposed encryption scheme is as follows.

Encryption

Input:	Message	M ($64 \times n$ bits)
	Redundant data	R (64 bits)
	Secret key	A ($\neq 0$, 64 bits)
		B_i ($1 \leq i \leq n+2$, each 64 bits)
		S (64 bits)
Output:	Ciphertext	C (($n+2$) $\times$ 64 bits)

Process E1: (Message Padding)

Generate padding P for the following encryption process as follows. Here, $A||B$ is a concatenation of bit strings A and B.

$$P = M \,||\, S \,||\, R.$$

P_i represents the i th 64-bit block of data P ($1 \leq i \leq n+2$).

Process E2: (Encryption)

Generate ciphertext blocks C_i as follows.

$$F_i = P_i \oplus B_i, \quad (1)$$
$$C_i = A \otimes F_i \oplus F_{i-1}. \quad (2)$$

Here, $F_0 = 0$, and symbols $\oplus$ and $\otimes$ represent addition and multiplication in finite field $\mathbf{F}2^{64}$, respectively.

Decryption

Input:	Ciphertext	C ($n' \times 64$ bits)
	Redundant data	R (64 bits)
	Secret key	A ($\neq 0$, 64 bits)
		B_i ($1 \leq i \leq n'$, each 64 bits)
		S (64 bits)
Output:	Message	Reception error or message M ($64 \times (n-2)$ bits)

Process D1: (Decryption)

Generate P_i as follows. Set $F_0 = 0$.

$$F_i = A^{-1} \otimes (C_i \oplus F_{i-1}), \quad (3)$$
$$P_i = F_i \oplus B_i. \quad (4)$$

Process D2: (Redundancy Data Test)
Check the last two blocks of P_i. Set R' to be the last block of P_i. Similarly, set S' to be the next-to-last block of P_i, i.e., $R'=P_{n'}$, and $S'=P_{n'-1}$. If and only if $R=R'$ and $S=S'$, output the rest of P_i ($n'-2$ blocks) as message data. Otherwise, output reception error signal.

We have proven the security of data confidentiality and data integrity for this scheme.

Theorem 1 (Security of MULTI-S01)

Assuming that the PRNG PANAMA is secure, MULTI-S01 provides perfect confidentiality and integrity. The probability of successful forgery is no more than $(n+1)/2^{64}$.

The proof of the theorem is given in the following sections. The following sections discuss confidentiality and integrity separately.

2.3. Perfect Confidentiality of MULTI-S01 Cipher

Let P be the concatenated data of M, S, and R. The necessary and sufficient condition for perfect confidentiality is $\Pr(P|C)=\Pr(P)$, where C represents the ciphertext, and $\Pr(X)$ is the probability of event X. $\Pr(X|Y)$ is the conditional probability of X gibrn event Y. To prove Theorem 1, we give a following corollary.

Corollary 1 (Number of equivalent keys for a plaintext)
Let P be the plaintext and C the ciphertext. For a certain fixed pair of P and C, there exists $2^{64}-1$ secret key pairs of (A, B) that encrypt P to C.

Proof:
In accordance with equations (1) and (2), we obtain the recursive equation for B_i:

$$B_1 = (C_1 \oplus F_0) \otimes A^{-1} \oplus P_1,$$
$$B_i = (C_i \oplus P_{i-1} \oplus B_{i-1}) \otimes A^{-1} \oplus P_i.$$

For an arbitrary A ($\neq 0$), B is determined by the equations above. Therefore, the number of key streams that encrypt P to C is at least the number of possible A values, i.e., $2^{64}-1$.
In the following we prove that B is determined uniquely for a fixed value of A, so the number of (A, B) pairs that encrypt P to C is exactly $2^{64}-1$.
For two secret key pairs, (A, B') and (A, B''), we examine the ciphertext of plaintext P.

$$F'_{i+1} = P_i \oplus B'_i,$$
$$C'_i = (F'_{i+1} \otimes \mathrm{A}) \oplus F'_i,$$
$$F''_{i+1} = P_i \oplus B''_i,$$
$$C''_i = (F''_{i+1} \otimes \mathrm{A}) \oplus F''_i,$$

Let j be the index such that $j = \min_i (i: B'_i \neq B''_i)$. From equations (1) and (2), $F'_j = F''_j$, $B' \neq B''$, and $F'_{j+1} = F''_{j+1}$. Therefore, $C''_j \neq C''_j$. Because of this, the two secret key pairs do not encrypt any plaintext to identical ciphertext.

In brief, the number of secret keys that maps an arbitrarily fixed plaintext to a ciphertext is $2^{64}-1$.

Based on this corollary, MULTI-S01 has perfect confidentiality.

Theorem 2 (Confidentiality of MULTI-S01)
An encryption scheme defined based on equations (1) and (2) provides perfect confidentiality.

Proof:
Let Pr(P) be the probability that P (padded message) is given as a plaintext. We evaluate Pr(C |P), the probability that P is mapped to C. Let l be the length (in bits) of P. In this case, the ciphertext is also l bits in length. Note that the number of possible secret keys is $(2^{64}-1)2^l$. According to corollary 1, $2^{64}-1$ secret keys out of a possible $(2^{64}-1)2^l$ map P to C. If secret keys are randomly chosen, for an arbitrary fixed plaintext P, the probability that P is mapped to C is:

$$\Pr(C|P) = (2^{64}-1)/\{(2^{64}-1)2^l\} = 1/2^l.$$

Therefore, the probability that the pair of a plaintext and corresponding ciphertext are (P,C) is:

$$\Pr(P,C) = \Pr(C|P)\Pr(P) = \Pr(P)/2^l.$$

The probability of a ciphertext Pr(C) is the sum of Pr(P,C) over whole possible P. Note that $\Sigma_P \Pr(P) = 1$.

$$\Pr(C) = \Sigma_P \Pr(P,C) = \Sigma_P \Pr(P)/2^l = 1/2^l.$$

Then, we apply Bayes theorem and we get:

$$\Pr(P|C) = \Pr(P,C) / \Pr(C) = \Pr(P).$$

This is the necessary and sufficient condition for perfect confidentiality.

Remark (Confidentiality of a plaintext being partially disclosed)
It is likely that an adversary knows part of the plaintext, due to redundant data or redundancy in the language itself. Even if part of the plaintext is known, it does not affect the disclosure of other information in this scheme. For instance, let I be all the information an adversary knows. This I reduces the possible plaintext space, P. We can think of the limitation on the space of P as a probability distribution, so that an adversary who knows the ciphertext does not know any more than the probability distribution of the plaintext.

2.4. Integrity of MULTI-S01 Cipher

First, we define an attack against message integrity.

Definition of Attacker
An adversary knows a known-plaintext consisting of message M, redundant data R, and corresponding ciphertext C. His goal is to generate a different ciphertext whose last two blocks pass the receiver's redundancy data test.

Under this definition, we evaluated the success probability of an adversary's ciphertext alteration, separately discussing (1) alteration of ciphertext of the same length, (2) alteration

of shorter ciphertext, and (3) alteration of longer ciphertext.

2.4.1. Case 1: Alteration of Ciphertext of the Same Length

First, note that an adversary cannot determine the value of A in secret key pair (A, B). Because of Corollary 1, an adversary given known-plaintext has $2^{64}-1$ secret key candidates, each of which has a different value of A.

Let C' be an altered ciphertext, and P' be the corresponding plaintext (without any data removed, i.e., redundancy and padding). From this definition, P' can be defined as:

$$
\begin{aligned}
F'_0 &= 0, \\
F'_i &= (C'_i \oplus F'_{i-1}) \otimes A^{-1}, \\
P'_i &= F'_i \oplus B_i .
\end{aligned}
$$

Similarly, the original plaintext is defined using the original ciphertext:

$$
\begin{aligned}
F_0 &= 0, \\
F_i &= (C_i \oplus F_{i-1}) \otimes A^{-1}, \\
P_i &= F_i \oplus B_i .
\end{aligned}
$$

For both cases, index i runs from 1 to n'. Eliminating feedback values F_i and F'_i, we have for each,

$$
\begin{aligned}
P'_{n-1} &= B_{n'-1} \oplus C'_{n'-1} A^{-1} \oplus C'_{n'-2} A^{-2} \oplus C'_{n'-3} A^{-3} \oplus C'_{n'-4} A^{-4} \oplus \ldots \oplus C'_1 A^{-n'+1} \\
&= B_{n'-1} \oplus (\oplus_{i=0\ldots n'-2} C'_{n'-i-1} A^{-(i+1)}), \\
P_{n-1} &= B_{n'-1} \oplus C_{n'-1} A^{-1} \oplus C_{n'-2} A^{-2} \oplus C_{n'-3} A^{-3} \oplus C_{n'-4} A^{-4} \oplus \ldots \oplus C_1 A^{-n'+1} \\
&= B_{n'-1} \oplus (\oplus_{i=0\ldots n'-2} C_{n'-i-1} A^{-(i+1)}),
\end{aligned}
$$

where $\oplus_i X_i$ represents an exclusive-or sum of X_i over the specified range of i. We define $\delta_i = C_i \oplus C'_i$. Note that any alteration can be represented uniquely by a sequence of δ_i. As noted, the objective of an adversary is to generate a ciphertext such that its plaintext passes the receiver's redundancy test. To pass the test, the plaintext should have the same random number and redundancy paddings as the last two blocks. Therefore, the condition $P_{n'-1} = P_{n''-1}$ is the necessary condition for successful alteration. Creating ciphertext C' is equivalent to knowing one of the corresponding δ sequences. In addition, because of the unique determination of δ from C' (and vice versa), the ability to knowing any valid ciphertext is equivalent to know any solution δ of

$$
0 = \oplus_{i=0\ldots n'-2} \delta_{n'-i-1} A^{-(i+1)} .
$$

According to Corollary 1, an adversary has no information about the value of A. Therefore, an adversary must determine δ values without knowing A. If we consider the equation to be an the equation about A, A has at most $n'-2$ roots for $A \neq 0$. Hence, the best way to determine δ is to do so so that the equation has $n'-2$ distinct roots. As a result, the success probability is upper-bounded by the probability that randomly chosen A has $n'-2$ distinct roots, $(n'-2) / (2^{64}-1)$, where n' is the number of blocks in the ciphertext. The number of corresponding plaintext n's is determined by $n = n'-2$.

2.4.2. Case 2: Alteration of Shorter Ciphertext

Let n'' be the number of blocks in the altered (shortened) ciphertext ($n'' < n'$ and $n'' > 2$). As in Case 1, P, P', and n' denote the original plaintext, the plaintext corresponding to the altered ciphertext, and the number of blocks in the original ciphertext, respectively.

The necessary and sufficient condition for a successful forgery is

$$P'_{n''} = P_{n'} (= R), \text{ and } P'_{n''-1} = B_{n''+1} ,$$

where $B_{n''+1}$ corresponds to secret padding S. For our security evaluation, we consider the necessary condition to be $P'_{n''-1} = B_{n''+1}$, i.e., $P'_{n''-1} \oplus B_{n''+1} = 0$. Referring to Equations (3) and (4) for decryption, we have

$$\begin{aligned}
P'_{n''-1} &= B_{n''-1} \oplus (\oplus_{i=0\ldots n''-2}\, C'_{n''-i-1} A^{-(i+1)}), \\
B_{n''+1} &= P_{n''+1} \oplus (\oplus_{i=0\ldots n''}\, C'_{n''-i+1} A^{-(i+1)}), \\
B_{n''-1} &= P_{n''-1} \oplus (\oplus_{i=0\ldots n''-2}\, C'_{n''-i-1} A^{-(i+1)}).
\end{aligned}$$

We prove the equation for $P'_{n''-1} \oplus B_{n''+1}$, eliminating all other B_i variables:

$$\begin{aligned}
P'_{n''-1} \oplus B_{n''+1} &= B_{n''-1} \oplus P_{n''+1} \\
&\quad \oplus (\oplus_{i=0\ldots n''-2}\, C'_{n''-i-1} A^{-(i+1)}) \\
&\quad \oplus (\oplus_{i=0\ldots n''}\, C'_{n''-i+1} A^{-(i+1)}) \\
&= P_{n''-1} \oplus P_{n''+1} \\
&\quad \oplus (\oplus_{i=0\ldots n''-2}\, C'_{n''-i-1} A^{-(i+1)}) \\
&\quad \oplus (\oplus_{i=0\ldots n''}\, C'_{n''-i+1} A^{-(i+1)}) \\
&\quad \oplus (\oplus_{i=0\ldots n''-2}\, C'_{n''-i-1} A^{-(i+1)}) \\
&= P_{n''-1} \oplus P_{n''+1} \\
&\quad \oplus (\oplus_{i=0\ldots n''-2}\, \delta_{n''-i-1} A^{-(i+1)}) \\
&\quad \oplus (\oplus_{i=0\ldots n''}\, C'_{n''-i+1} A^{-(i+1)}).
\end{aligned}$$

Therefore, the determination of *C'* such that the following equation holds for an unknown randomly chosen A is the necessary condition for a successful attack.

$$0 = P_{n''-1} \oplus P_{n''+1} \oplus (\oplus_{i=0\ldots n''-2}\, \delta_{n''-i-1} A^{-(i+1)}) \oplus (\oplus_{i=0\ldots n''}\, C'_{n''-i+1} A^{-(i+1)}).$$

Note that for a non-zero variable A, this equation has at most $n''+1$ distinct roots. Obviously, the best way to determine δ values is to fix them so that the above equation has $n''+1$ distinct roots. In this case, the probability of the case that an adversary can control the redundancy is $(n''+1) / (2^{64}-1)$. This probability upper-bounds that of a successful forgery.

2.4.3. Case 3: Alteration of Longer Ciphertext

Let n'' be the number of blocks in altered (lengthened) ciphertext ($n'' > n'$). As above, P, P', and n' denote the original plaintext, the plaintext corresponding to the altered ciphertext, and the number of blocks in the original ciphertext, respectively.

The necessary and sufficient condition for a successful forgery is

$$P'_{n''} = P_{n'} (= R), \text{ and } P'_{n''-1} = B_{n''+1} ,$$

where $B_{n''+1}$ corresponds to secret padding S. For our security evaluation, we consider the

necessary condition to be $P'_{n''-1} = B_{n''+1}$., i.e., $P'_{n''-1} \oplus B_{n''+1} = 0$, the same as in Case 2. Referring to Equations (3) and (4) for decryption, we have

$$P'_{n''-1} = B_{n''-1} \oplus (\bigoplus_{i=0...n''-2} C'_{n''-i-1} A^{-(i+1)}),$$
$$B_{n''+1} = P_{n''+1} \oplus (\bigoplus_{i=0...n''} C'_{n''-i+1} A^{-(i+1)}).$$

We prove the equation for $P'_{n''-1} \oplus B_{n''+1}$:

$$P'_{n''-1} \oplus B_{n''+1} = B_{n''-1} \oplus P_{n''+1}, \oplus (\bigoplus_{i=0...n''-2} C'_{n''-i-1} A^{-(i+1)}), \oplus (\bigoplus_{i=0...n''} C'_{n''-i+1} A^{-(i+1)}).$$

Therefore, determination of *C'* such that the following equation holds for an unknown randomly chosen *A* is the necessary condition for a successful forgery,

$$0 = B_{n''-1} \oplus P_{n''+1} \oplus (\bigoplus_{i=0...n''-2} C'_{n''-i-1} A^{-(i+1)}) \oplus (\bigoplus_{i=0...n''} C'_{n''-i+1} A^{-(i+1)}).$$

Note that for non-zero variable *A*, the above equation has at most *n"*+1 distinct roots. Obviously, the best way to determine δ values is to fix them so that this equation has *n"*+1 distinct roots. However, an adversary cannot predict the constant term of the above equation because $B_{n''-1}$ is chosen randomly. In this case, the probability of an adversary controlling the redundancy is much less than $(n''+1) / (2^{64}-1)$ and upper-bounds that of a successful forgery.

3. The Security of Pseudorandom Number Generator PANAMA

PANAMA is a cryptographic module designed by J. Daemen and C. Clapp [8]. It can be used both as a cryptographic hash function and as a stream cipher. While many cryptanalysts have attended to PANAMA because the designers of PANAMA are very famous cryptanalysts, as far as I know, there have been no reports analyzing Panama PRNG it since it was proposed. As for the hash mode of Panama, it was reported that the hash collision can be generated with the fewer computational complexity[18]. The major report about the security of PANAMA can be found in the summary of Daemen's research [7].

The output of a pseudorandom number generator (PRNG) must be indistinguishable from a truly random number sequence. This requirement is very ambiguous. There are three conditions for a PRNG to provide security:

(1) The output sequence of a PRNG must have a long period.
(2) The sequence must pass statistical randomness tests.
(3) There must be no theoretical flaw.

A secure PRNG requires a long period matching the length of the secret parameter. It is difficult to evaluate the period of the output sequences of PANAMA, but its huge internal state and the complex behavior of non-linear transformation γ imply that the output sequences of PANAMA have periods that are long enough.

In this paper the statistical randomness tests, condition (2), are chi square tests on some statistical properties. The output sequences of PRNG should satisfy such elemental properties as mono-bit frequency. We describe the results of these tests in section 3.1.

Condition (3) addresses the other various properties. We focus on the linear correlation property of PANAMA. We describe about this and other properties in section 3.2.

3.1. Randomness Test

Hereafter, we use PANAMA only as a PRNG. In this section we present the results of our statistical randomness tests.

The tests we used to examine the randomness of the output sequences of PANAMA can be found in FIPS 140-1 [9]. They were designed for examining short sequences so they are not suited to sequences used for stream ciphers. We thus used not only the tests FIPS but also frequency tests for long sequences.

3.1.1.FIPS 140-1 Randomness Tests

Three randomness tests can be found in FIPS 140-1 for examining 20,000-bit data streams:

(1) Monobit frequency test

(2) Poker test (four-bit frequency test)

(3) Run test (including detection of a long run)

We applied the tests to 2^{21}-bit (256K-byte) consecutive output streams from PANAMA because a 20,000-bit stream is too short to use for a stream cipher. We used 2^{12} sets of initial data of PANAMA (the key and the diversification parameter) generated by the random number generator in the standard C library. The method used for the frequency tests was that described by Knuth [15], and the method used for the run test was described by Menezes et al [17].

Table 2 shows the results. The values in the table represent how many of the initial data streams regarded as wrong in each test and the rejection probability. For instance, 39 of 4096 output sequences generated from different initial data streams were distinguished from truly random sequences with probability 99% on the mono-bit test. The parenthetic values represent the ratio of the number of rejected initial data streams to the number of tested ones.

Table 2: Result of randomness tests (FIPS 140-1)

Rate of rejection	0.05	0.01	0.001
Mono-bit test (/4096)	152 (0.037)	39 (0.0095)	3 (0.0007)
Poker test (/4096)	172 (0.042)	33 (0.0081)	3 (0.0007)
Run test (/4096)	196 (0.048)	43 (0.0105)	4 (0.001)

Table 2 shows that the ratio of the number of rejected initial data streams to the number of tested ones nearly equaled the rate of rejection in each test. On equal terms with above tests the longest run was 32 long. The expected value that a run of length 32 exists in a 2^{21}-bit stream is 2^{-12}.

3.1.2.Further Test for Long Sequences

The discussion in the previous section concludes that the MULTI-S01 cipher has to change random number A for each 2^{32} blocks (2^{38} bits) in the plaintext. We assumed that the length of the plaintext is less than 2^{32} blocks (2^{38} bits) and applied several tests to the corresponding long streams. The algorithm of PANAMA consists of 32-bit operations, but it is difficult to check the 32-bit frequency because doing so requires much calculation. We thus checked only the 1-, 2-, and 8-bit frequencies. The method we used was described by Knuth

[15].

Additionally, we applied these tests to 2^{17}-, 2^{21}-, 2^{25}-, and 2^{29}-bit-length sequences to observe the effect of the sequences length. We fixed 512 sets of initial data generated by the random number generator in the standard C library and used them as the initial data.

Table 3, Table 4, and Table 5 shows the results of the frequency tests. The values in the tables represent how many of the initial data were regarded as wrong in each test and the rejection probabilities. For instance, in the mono-bit frequency test to 2^{29}-bit-length sequences, 5 of 512 output sequences generated from the initial data were distinguished from truly random sequences with probability 99%.

Table 3: Results of randomness test (1-bit frequency)

Rate of rejection / data length (bit)	0.05	0.01
2^{17}	17 (/512)	3 (/512)
2^{21}	24 (/512)	8 (/512)
2^{25}	27 (/512)	4 (/512)
2^{29}	21 (/512)	5 (/512)
2^{33}	26 (/512)	5 (/512)

Table 4: Results of randomness test (2-bit frequency)

Rate of rejection / data length (bit)	0.05	0.01
2^{17}	20 (/512)	4 (/512)
2^{21}	22 (/512)	3 (/512)
2^{25}	29 (/512)	4 (/512)
2^{29}	21 (/512)	6 (/512)
2^{33}	22 (/512)	4 (/512)

Table 5: Results of randomness test (8-bit frequency)

Rate of rejection / data length (bit)	0.05	0.01
2^{17}	29(/512)	10(/512)
2^{21}	21(/512)	3(/512)
2^{25}	17(/512)	7(/512)
2^{29}	18(/512)	5(/512)
2^{33}	26(/512)	4(/512)

The ratio of the number of rejected initial data to the number of tested ones nearly equaled to the rate of rejection in each test, and there was no initial data such that generated a stream distinguishable from truly random sequences for all of the plaintext lengths. These results mean that PANAMA is a good enough PRNG for MULTI-S01.

3.2. Cryptographic Security

Because there are various types of stream PRNGs, it is very difficult to apply general

attacks despite there being many methods for estimating the security of PRNGs. We restrict our evaluations of the attacks on PRNGs to ciphertext only attacks. "An attack on the PRNGs" can be one of two types:

(i) To obtain the secret information (seed) from the output sequences,

(ii) To predict unknown output sequences from previous sequences.

Linear complexity is commonly used characteristic for measuring the security of PRNGs. This complexity is defined as the length of the shortest LFSR that represents the given PRNG. However, only a recursive algorithm can calculate the linear complexity from an output sequence, so it is impractical to calculate the linear complexity of a PRNG that has a long period. For instance, a PRNG with a 256-bit-length seed is secure if its linear complexity is more than 2^{255}. We therefore did not try to calculate the linear complexity of PANAMA. Also, we did not investigate several attacks specialized in the PRNGs based on LFSR.

In this paper we consider only the differential and linear correlation property of PANAMA and limit the attacks on PANAMA to those of the type (ii). To consider type (i) attacks is unrealistic because the initialization of PANAMA makes it too difficult for adversaries to guess the seed. The linear correlation property was investigated by Daemen [7], so we only summarize the results.

3.2.1.Difference Propagation and Linear Correlation

Difference propagation

The difference of two elements of $\mathbf{F}2^n$ a, a^* is given by $a \oplus a^* (= a')$. Suppose f is a function defined over $\mathbf{F}2^n$. $b' = f(a) \oplus f(a^*)$ is the output difference associated with a'. We say that the input difference a' propagates to the output difference b'. In general, it is difficult to apply the differential attack to output sequences of PRNGs.

Linear correlation

The correlation between two Boolean function f and g is given by

$C(f,g) := 2 \cdot \Pr(f(a) = g(a)) - 1$.

We say that f and g are correlated if $C(f,g) \neq 0$. Consider a non-linear transformation of $\mathbf{F}2^n$ φ and linear Boolean function f and g. Then correlation coefficient $C(f \circ \varphi, g)$ means the linear approximation probability of φ.

3.2.2.Non-linear Properties of ρ

Non-linear property of γ

Two properties were proven by Daemen [7].

(1) The linear correlation probability of γ exponentially decreases in proportion to the increase in the hamming weight of the output mask.

(2) The difference propagation probability of γ exponentially decreases in proportion to the increase in the hamming weight of the input difference.

Diffusion property of θ

A linear transformation θ, is used to stir the location of bits. It consists of two parts. The first part transforms in units of a word, and the second part permutates the locations of the bits inside a word. The first part is an invertible transformation of 17 words and diffuses the words well in a linear transformation defined as the summation of three words. Table 6 is the hamming weight distribution table of θ introduced by Daemen [7].

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	-	-	-	-	51	-	85	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	17	-	153	-	374	-	136	-	-	-	-	-	-	-	-
4	-	-	-	-	119	-	561	-	1071	-	578	-	51	-	-	-	-	-
5	-	-	-	34	-	561	-	1802	-	2465	-	1173	-	153	-	-	-	-
6	-	-	-	-	374	-	1870	-	4624	-	3910	-	1445	-	153	-	-	-
7	-	-	-	170	-	1394	-	5440	-	7208	-	4233	-	935	-	68	-	-
8	-	-	68	-	748	-	4454	-	8806	-	7344	-	2584	-	289	-	17	-
9	-	17	-	289	-	2584	-	7344	-	8806	-	4454	-	748	-	68	-	-
10	-	-	68	-	935	-	4233	-	7208	-	5440	-	1394	-	170	-	-	-
11	-	-	-	153	-	1445	-	3910	-	4624	-	1870	-	374	-	-	-	-
12	-	-	-	-	153	-	1173	-	2465	-	1802	-	561	-	34	-	-	-
13	-	-	-	-	-	51	-	578	-	1071	-	561	-	119	-	-	-	-
14	-	-	-	-	-	-	-	-	136	-	374	-	153	-	17	-	-	-
15	-	-	-	-	-	-	-	-	-	-	-	85	-	51	-	-	-	-
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	17	-	-	-
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1

Table 6: Hamming weight distribution table of θ

This table shows the correlation between the hamming weights of the input and output data. The numbers in the columns are the hamming weights of the input 17-bit data and the hamming weights of the output 17-bit data. The element in row *i* and column *j* denotes the frequency that the input hamming weight is *i* and the output hamming weight is *j*. For instance, there are 561 patterns (out of 2380 patterns) in which the hamming weight of the input data is 4 and that of the output data of θ is 6.

3.2.3.Resistance Property of Linear Cryptoanalysis

We separate ρ into key addition functions σ and ρ' consisting of γ, π, and θ.

$$\rho' = \theta \circ \pi \circ \gamma,$$
$$\rho = \rho' + K.$$

$a^{(t)}$, $a^{(t)}_{\mathrm{H}}$, and $a^{(t)}_{\mathrm{L}}$ denote the outputs of ρ in round t, their 1 to 8 word, and 9 to 16 word .

Some linear correlations between $a^{(t)}_{\mathrm{L}}$ and $\rho'(a^{(t)})_{\mathrm{L}}$ with probability $1/2 \pm 2^{-3}$ were found by searching the input and output masks such that only one bit was 1 and the other 31 bits were 0 in each word. This is valid because ρ' has no operation such that two or more bits of one word affect each only one bit of the output. We denote input and output masks Γ_1 and Γ_2 that produce any one of the above correlations as

$$\Gamma_1 \bullet a_L^{(t)} = \Gamma_2 \bullet a_L^{(t+1)}.$$

This and the result introduced in 3.2.2 leads above linear representation is the best linear approximation of ρ' with known data.

$$\Gamma_1 \bullet a_L^{(t)} + \Gamma_2 \bullet a_L^{(t+1)} = \Gamma_2 \bullet K^{(t)} \quad (\text{prob} = 2^{-3})$$

is the best linear approximation of ρ that includes one bit of buffer b. To construct on LFSR that represents all $K^{(t)}$, we needs 63 $K^{(t)}$'s because of the size of buffer b. We can calculate the probability of this expression from the above linear approximation of ρ':

$$2^{63-1}(2^{-3})^{63} = 2^{-127}.$$

Thus, it takes $2^{2*127}*256$ (= 2^{262}) output sequences to break PANAMA.

3.2.4.Attacks on Reduced PANAMA

In PANAMA all values XORed from buffer *b* are different. This makes it difficult to find any correlation between the output sequence and the internal state. However, making all $K^{(t)}$'s independent makes nonsense of the essence of PRNG. We thus considered three reduced PANAMAs, TOY1, 2, and 3, and attack these weak PRNG-like modules. We found that the nonlinearlity of the function updating buffer *b* guarantees the security of PANAMA.

TOY1: Outputs all of state *a*

The updating transformation of TOY1 is described as follows:

$$a^{(t+1)} = \rho'(a^{(t)}) + K^{(t)}.$$

This shows $K^{(t)}$ in all round. That is, we can get a 512-bit state of buffer *b* in any round. Therefore all of buffer *b* come out from the output sequences of 21 rounds.

TOY2: Neglects the insertion of $K^{(t)}$

(1) We can get the relational expression

$$a_L^{(t+1)} = \rho'(a^{(t)})_L$$

and give a representation in Boolean polynomials:

$$c_{9,j} = a_{12,j+13} + (a_{13,j+13}+1)a_{14,j+13} + a_{2,j+23} + (a_{3,j+23}+1)a_{4,j+23} + a_{6,j+27} + (a_{7,j+27}+1)a_{8,j+27} + 1,$$
$$c_{10,j} = a_{2,j+23} + (a_{3,j+23}+1)a_{4,j+23} + a_{9,j+2} + (a_{10,j+2}+1)a_{11,j+2} + a_{13,j+9} + (a_{14,j+9}+1)a_{15,j+9} + 1,$$
$$c_{11,j} = a_{9,j+2} + (a_{10,j+2}+1)a_{11,j+2} + a_{16,j+14} + (a_{0,j+14}+1)a_{1,j+14} + a_{3,j+24} + (a_{4,j+24}+1)a_{5,j+24} + 1,$$
$$c_{12,j} = a_{16,j+14} + (a_{0,j+14}+1)a_{1,j+14} + a_{6,j+27} + (a_{7,j+27}+1)a_{8,j+27} + a_{10,j+8} + (a_{11,j+8}+1)a_{12,j+8} + 1,$$
$$c_{13,j} = a_{6,j+27} + (a_{7,j+27}+1)a_{8,j+27} + a_{13,j+9} + (a_{14,j+9}+1)a_{15,j+9} + a_{0,j} + (a_{1,j}+1)a_{2,j} + 1,$$
$$c_{14,j} = a_{13,j+9} + (a_{14,j+9}+1)a_{15,j+9} + a_{3,j+24} + (a_{4,j+24}+1)a_{5,j+24} + a_{7,j+1} + (a_{8,j+1}+1)a_{9,j+1} + 1,$$
$$c_{15,j} = a_{3,j+24} + (a_{4,j+24}+1)a_{5,j+24} + a_{10,j+8} + (a_{11,j+8}+1)a_{12,j+8} + a_{14,j+3} + (a_{15,j+3}+1)a_{16,j+3} + 1,$$
$$c_{16,j} = a_{10,j+8} + (a_{11,j+8}+1)a_{12,j+8} + a_{0,j} + (a_{1,j}+1)a_{2,j} + a_{4,j+6} + (a_{5,j+6}+1)a_{6,j+6} + 1,$$

where a_{ij} means the *i*-th word and *j*-th bit of $a^{(t)}$, and c_{ij} means the *i*-th word and *j*-th bit of $a^{(t+1)}$. Given the 11th and 15th words of the output of round *t*+1, we have the following relation:

$$c_{15,j} = c_{11,j} + (\text{the value from the } t\text{-th round's output sequence}) + (a_{0,j+14} + 1)a_{1,j+14}.$$

The value of $(a_{0,j+14} + 1)a_{1,j+14}$ is 1 with probability 1/4 if these components are independent of the other components. This shows that nonlinear transformation ρ of TOY2 has a nonlinear correlation with high probability and is comprised of known bits. As a result, we can predict $c_{15,j}$ with probability 3/4 if we get the output sequence of TOY2 to $c_{11,j}$.

(2) The discussion in 3.2.3 gives a linear correlation with deviation 1/8 that is comprised of known bits. Thus, we can predict a bit in the output sequence with probability 5/8 from two rounds of output, as in case (1).

TOY3: Fixes insert value *K*

We can get a bit of *K* in the same way as a linear attack on a block cipher by using the deviation described in the attack on TOY2.

4. Implementation

4.1. Software Implementation

Because of the internally used operations in finite field $\mathbf{F}2^{64}$, the MULTI-S01 cipher operates very efficiently, particularly on a 64-bit processor. Initially, we evaluated the performances on 64-bit 600-MHz processor (DEC Alpha). We implemented it using the C language. The details of the implementation environments are given in Table 7.

Table 7: Environment for MULTI-S01 Software Impl.(DEC Alpha)

Hardware	**CPU**	Alpha 21164A 600 MHz (w/ 4 MB 3rd cache)
	RAM	512 Mbyte
Software	**OS**	DIGITAL UNIX 4.0E
	Compiler	DEC cc
	Language	C
	Compiler option	`-tune-ev56 –arch –ev56 O6`

The next Table 8 shows a rough estimation of the cost of this implementation.

Table 8: Memory Consumption for Software Implementation

Code Size (step)		1167
Work Area (KB)	**Initialization**	2.4
	Encryption	3.6
	Decryption	3.7

The code size is the number of lines in C source code, excluding blank lines and lines for comments. For the work area, the memory size excludes storage for the whole message (or ciphertext).

In this environment, the MULTI-S01 cipher performs encryption at 270.7 Mbps and decryption at 267.3 Mbps. These figures converted to cycles/byte are shown in Table 9. For reference, we also show the performance of PANAMA (optimized C code by the authors of PANAMA) on the same environment.

These results were obtained in experiments in which sufficient repetitions of operations on plaintext, 4096 bytes long, took a couple of seconds.

Cipher initialization including key set-up, 2 KB table generation for multiplication, PANAMA initialization, and inverse calculation for decryption took 31,737 cycles in total.

Table 9: MUTLI-S01 Performance in Cycles/Byte (DEC Alpha)

Operation	Performance
Encryption	**17.7**
Decryption	**18.0**
PANAMA	**6.7**

In implementations on a 32-bit processor, the multiplication in $\mathbf{F}2^{64}$ gets more complicated, degrading performance significantly. For example, an implementation on a Pentium(R) Celeron 350-MHz processor running the C language performed encryption at 55 Mbps.

However, one can use MMX instructions on the processor so that 64-bit operations are efficiently executed, thereby improving performance.

We also report on the implementation on a Pentium(R) III processor. As is mentioned above MULTI-S01 uses 64-bit instructions very often. Therefore we coded MULTI-S01 with assembly language so that some MMX instructions are used. Consequently, we achieved the performances shown in Table 10. The devise on which we evaluated MULTI-S01 equips Intel Pentium(R) III 600 MHz processor with 256 KB cache memory. The results are of encryptions for a message in 32 KB length. In the same environment, the encryption achieved 21.75 clock/byte for a 1 MB message.

Table 10 : MUTLI-S01 Performance in Cycles/Byte (Pentium(R) III)

Operation	Performance
Encryption	**17.6**
Decryption	**18.5**

4.2. Hardware Implementation

Because most of MULTI-S01, except for PANAMA, consists of exclusive-or operations and multiplication on $\mathbf{F}2^{64}$, it can be implemented very efficiently in hardware. In our evaluation, an encryption circuit with 140K gates was estimated to achieve 9.1 Gbps, assuming a 0.35-μm CMOS process.

4.2.1. Design of logic circuits

In our evaluation, we designed two optimized circuits: a speed-optimized circuit and a gate-size optimized circuit. We used Hitachi's HG73C cell library (0.35-μm rule). The operating speed was estimated from the minimum gate latency, so the actual throughput could be slower than shown in this document.

Pseudorandom number generator (PANAMA)

Detail design of this circuit is described. Figure 1 shows the block diagram of PANAMA, and followings are its specifications.

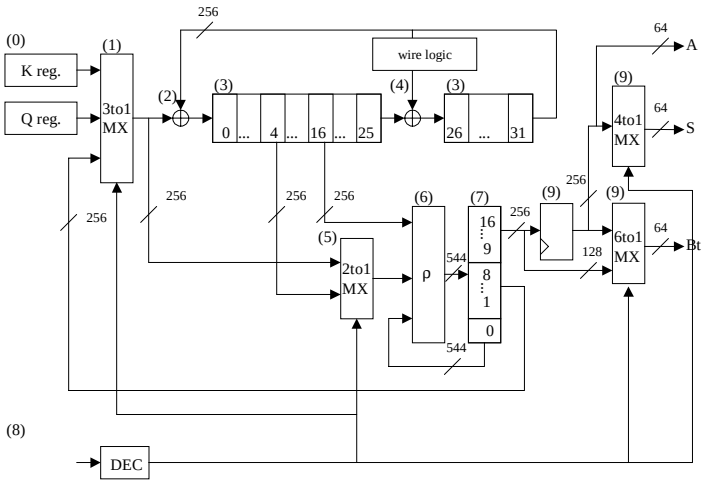

Figure 1: Block Diagram of PANAMA

Specifications:
Internal registers: *K* register (256 bits), *Q* register (256 bits)
Input: clock
Internal operation: Circuit operated by clock synchronization; *A* register, *B* register, λ circuit, and ρ circuit are given a clock by every four mother clocks.
Output: There are three basses. At the last stage, P/S operation is used so that a 64-bit pseudorandom number (the B_i sequence) is output by every clock using the B_t bass. *A* is output through the *A* bass at the same time as B_1 is generated and output. *S* is output through the *S* bass at the same time as B_{n-1} is output.

Details of the speed-optimized implementation are shown in Figure 4. The λ function of PANAMA is implemented as two independent circuits, λ_0 and λ_1. The core macros used in each segment, (0)…(9), are listed in Table 11.

Table 11. Core Macros Used for PANAMA

	Segment		Core Macro	No. of Units
(0)	*K* register, *Q* register		DFF	512
(1)	Switch input to b_0 at push/pull modes		3to1MX	256
(2)	λ_0	$b_{31} \oplus$ Ouput 1	2in-EOR	256
(3)	*B* register		DFF	8192
(4)	λ_1	Permutation of b_{31}	Wire Logic	-
		(Permuted b_{31}) $\oplus$ b_{25}	2inEOR	256
(5)	Switch input to ρ at push/pull modes		3to1MX	256
(6)	ρ	γ	2in-inv	544
			2in-OR	544
			2in-EOR	544
		π	Wire Logic	-
		θ	3in-EOR	544
		δ	2in-EOR	544
(7)	*A* register		DFF	544
(8)	Initialization sequence counter		DFF	6
	MX switching signal		+α	+α
(9)	Temporary register for last padding		DFF	256
	Selector of B_t		6to1MX	64
	Selector of *S*		4to1MX	64

Inverse Calculation

The inverse of arbitrary element $\alpha^{-1}=\alpha^{(2**64-2)}$ is calculated as follows:

Table 12: Inverse Operation of α

Value	Contents
S_0	α
S_1	$S_0^2 \otimes \alpha$
S_2	$S_1^2 \otimes \alpha$
S_3	$S_2^2 \otimes \alpha$
…	…
S_{62}	$S_{61}^2 \otimes \alpha$
S_{63}	$S_{62}^2 \otimes 1$

Since $(2^{64}-2)d$ = **FFFF FFFE**h,

$$S_n=(S_{n-1})^2 \; x \; \alpha \quad (1 \le n < 63),$$
$$S_n=(S_{n-1})^2 \; (n = 63).$$

Hence, $S_{63}=\alpha^{-1}$. The number of multiplications, including squaring and multiplication, is 125. Figure 2 shows the structure of the circuit in which the multiplication circuit is shared with the one in the encryption part.

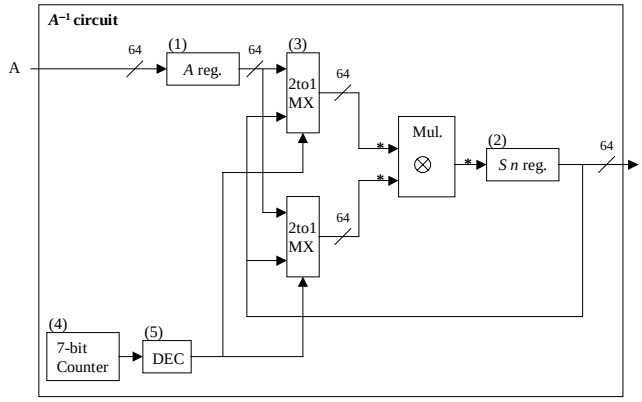

Figure 2: Block Diagram of Inverse Circuit

Specifications:
Internal registers: *A* register (64 bits), S_n temporary storage registers (64 bits)
Input: *A* (64 bits), clock
Internal operation: By path-switching signal sent from the operation sequencer, a stage of each operation is controlled.
Output: A^{-1} (64 bits).
Table 13 shows the core macros used in the operation circuits. The multiplier can be shared with that of the data randomizing part, so as to reduce the total gate count. Otherwise, the multiplier must be independently equipped and the selector (5) can be omitted.

Table 13: Core Macros Used for α^{-1}

	Segment			Core Macro	No. of Units
(1)	*A* register			DFF	64
(2)	S_n register			DFF	64
(3)	Switching operation path			2to1MX	128
(4)	Operation sequence counter			DFF	7
	MX switching signal			+α	+α
(5)	Path switch for multiplier sharing			2to1MX	256
	Multiplier	Speed optimized	Main circuit	-	(36k)
			+α	-	(1.3k)
		Gate count optimized	Main circuit	-	(0.9k)
			+α	-	(1.3k)

Data Randomizing Part
Specifications:
Internal registers: *A* register (64 bits), B_t register (64 bits), *R* register (64 bits), Plaintext/Ciphertext P_t / C_t data register (64 bits), Ciphertext/Plaintext C_t / P_t data register (64 bits),
Input: *A* (64 bits), B_t (64 bits), *S* (64 bits); directly connected to PANAMA circuit, *M* (64 bits); external input for external P_t / C_t, clock
Internal operation: Generate C_t / P_t out of P_t / C_t , *S*, *R*, *A*, and B_t.
Output: C_t / P_t (64 bits).
Figure 3 shows a block diagram of the data randomizing part. If the multiplier is shared with the inverse circuit, a data selector must be inserted in the data bass. The inserted data selector

(2to1MX) is shown (5) in Table 13, and the position to insert the selector is shown with an "*" in Figure 2, and Figure 3.

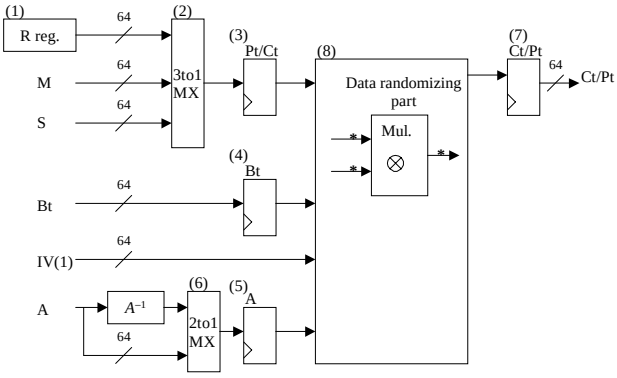

Figure 3: Block Diagram of Data Randomizing Part

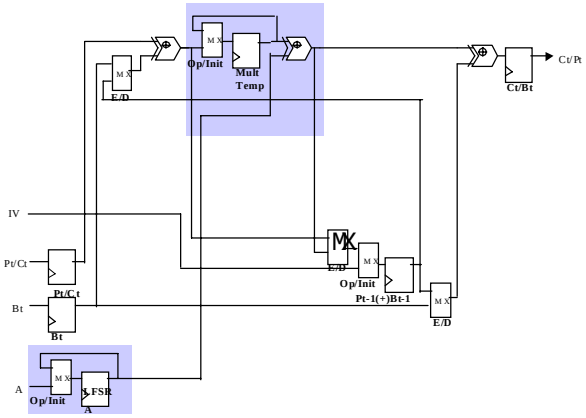

Figure 4: Block Diagram of Data Randomizing Circuit (Highest Speed)

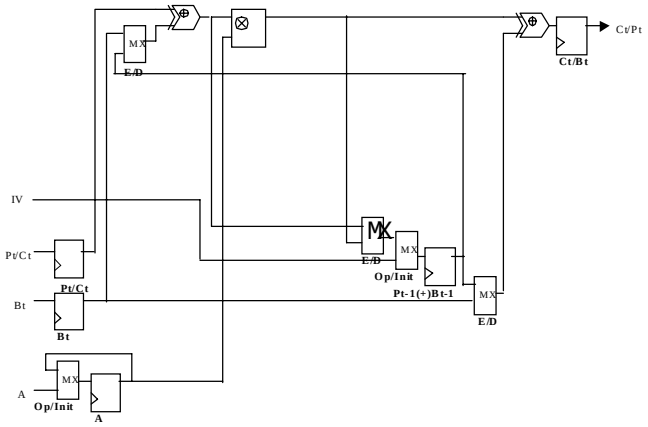

Figure 5: Block Diagram of Data Randomizing Circuit (Smallest Gate Count)

Table 14: Summary of Estimated Gate Count

Optimization	Highest Speed	Smallest Gate Count
Gate count	38.5k	3k
Operating speed (Hz)	150 M	800 M
Clocks for 64-bit operation	1	65
Throughput (bps)	9.6 G	800 M

Table 15: Core Macros Used for Data Randomizing Part (Highest Speed)

	Segment		Core Macro	No. of Units
(1)	R register		DFF	64
(2)	P_t / C_t switch		3to1MX	64
(3)	P_t / C_t register		DFF	64
(4)	B_t register		DFF	64
(5)	A register		DFF	64
(6)	A/A^{-1} switch		2to1MX	64
(7)	C_t / P_t register		DFF	64
(8)	Multiplier	Main Circuit	-	(36k)
		+α	-	(1.3k)

Table 16: Core Macros Used for Data Randomizing Part (Smallest Gate Count)

	Segment		Core Macro	#Units
(1)	R register		DFF	64
(2)	P_t / C_t switch		3to1MX	64
(3)	P_t / C_t register		DFF	64
(4)	B_t register		DFF	64
(5)	A register		DFF	64
(6)	A/A^{-1} switch		2to1MX	64
(7)	C_t / P_t register		DFF	64
(8)	Multiplier	Main circuit	-	(0.9k)
		+α	-	(1.3k)

4.2.2. Evaluation of Gate Size and Throughput

We evaluated the throughput of the above circuits. Using Hitachi's HG73C cell library (0.35-μm rule). Table 17 and Table 18, respectively, summarize the estimated gate count and throughput of each optimized circuit.

Table 17: Estimated Gate Count

Optimization	**Highest Speed**		**Smallest Gate Count**	
Multiplier	Shared	Two Circuits	Shared	Two Circuits
PANAMA	61.5k			
Inverse	1.8k	38.4k	1.8k	3.3k
Randomizing	39.6k	39.6k	4.5k	4.5k
Total	102.9k	139.5k	67.8k	69.3k

Table 18: Estimated Throughput

Optimization	**Maximum Latency**			
	Highest Speed		**Smallest Gate Count**	
	Shared	**Two Circuits**	**Shared**	**Two Circuits**
PANAMA	1.36ns			
Inverse	6.88 ns	6.16 ns	1.48 ns	1.12 ns
Randomizing	7.71 ns	6.99 ns	1.59 ns	1.23 ns
Clock frequency	130 MHz	140 MHz	620 MHz (200 MHz)	730 MHz (200 MHz)
A^{-1} duration	1 μs	0.8 μs	13 μs (40 μs)	11 μs (40 μs)
Throughput	8.3 Gbps	9.1 Gbps	620 Mbps (200 Mbps)	730 Mbps (200 Mbps)

Finally we show the details of critical paths for each circuit, which we used for estimated throughtputs.

PANAMA

Critical path: (0)-(1)-(5)-(6)-(7)

Segments on the path: 3to1MX, 2to1MX, EOR, DFF.

Latency: 1.36 ns

The maximum clock frequency for the circuit excluding (9) is 730 MHz.

This circuit operates faster than the data randomizing part, so it is not a bottle neck in encryption performance.

Inverse Circuit

(shared multiplier, highest-speed circuit)

Critical path: (1)-(3)-(2)

Segments on the path: 2to1MX×3, multiplier, DFF.

Latency: 6.88 ns

(shared multiplier, smallest-gatecount circuit)

Critical path: (1)-(3)-(2)
Segments on the path: 2to1MX×3, DFF.
Latency: 1.48 ns

(two independent multipliers, highest-speed circuit)
Critical path: (1)-(3)-(2)
Segments on the path: 2to1MX×1, multiplier, DFF.
Latency: 6.16 ns

(two independent multipliers, smallest-gatecount circuit)
Critical path: (1)-(3)-(2)
Segments on the path: 2to1MX×2, DFF.
Latency: 1.12 ns

Data Randomizing Part

(shared multiplier, highest-speed circuit)
Segments on the critical path: 2to1MX×5, EOR, multiplier, DFF.
Latency: 7.71 ns

(shared multiplier, smallest-gatecount circuit)
Segments on the critical path: 2to1MX×3, EOR, DFF.
Latency: 1.59 ns

(two independent multipliers, highest-speed circuit)
Segments on the critical path: 2to1MX×3, EOR, multiplier, DFF.
Latency: 6.99 ns

(two independent multipliers, smallest-gatecount circuit)
Segments on the critical path: 2to1MX×2, EOR, DFF.
Latency: 1.23 ns

Performance in Hardware Implementation
We estimated that the speed-optimized circuit of MULTI-S01 will encrypt at 9.1 Gbps with 140-MHz clock input. The circuit needs 140k gates. For the smallest implementation, the MUTLI-S01 circuit needs 68k gates. This smallest circuit encrypts at 620(200) Mbps with a clock input of 620(200) MHz.

5. References

[1] ANSI X9.9, ``American National Standard for Financial Institution Message Authentication (Wholesale)," *American Bankers Association*, 1981. Revised 1986.

[2] Anderson, R., Biham, E., ``Two Practical and Provably Secure Block Ciphers: BEAR and LION," *Fast Software Encryption, Third International Workshop, Cambridge, UK, February, LNCS Vol. 1039, Springer-Verlag*, 1996.

[3] Aoki, K., Ohta, K., ``Operations in F2n by Software Implementation," *The Symposium on Cryptography and Information Security, SCIS'97-14A*, 1997 (in Japanese).

[4] Bellare, M., Kilian, J., Rogaway, P., ``On the security of cipher block chaining," *Advances in Cryptology, -CRYPTO'94, LNCS Vol. 839, Springer-Verlag*, 1994.

[5] Biryukov, A. and Kushilevitz, E., ``From Differencial Cryptanalysis to Ciphertext-Only Attacks," *Advances in Cryptology, -CRYPTO'98, Proceedings, LNCS Vol. 1462, Springer-Verlag*, 1998.

[6] Black, J., Halevi, S., Krawczyk, H., Krovets, T., Rogaway, P., ``UMAC: Fast and Secure Message Authentication," *Advances in Cryptology, -CRYPTO'99, LNCS Vol. 1666, Springer-Verlag*, 1999.

[7] Daemen, J., ``Cipher and hash function design strategies based on linear and differential cryptanalysis," *Doctoral Dissertation*, March 1995, K. U. Leuven.

[8] Daemen, J., Clapp, C., ``Fast Hashing and Stream Encryption with PANAMA," *Fast Software Encryption, 5th International Workshop, FSE'98, Proceedings, LNCS Vol. 1372, Springer-Verlag*, 1998.

[9] FIPS 140-1, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standard (FIPS), Publication 140-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., January 1994, **`http://www.itl.nist.gov/fipspubs/index.htm`**.

[10] Furuya, S., Satoh, H., Takahashi, M., Miyazaki, K., Takaragi, K., Sasaki, R., ``Integrity Aware Mode of Stream Cipher," *The Symposium on Cryptography and Information Security, SCIS2000-A17*, 2000 (in Japanese).

[11] Gligor, V., Donescu, P., ``Integrity-Aware PCBC Encryption Schemes," *The 1999 Security Protocols Workshop Pre-proceedings*, Cambridge, 1999.

[12] Halevi, S., Krawczyk, H., ``MMH: Software message authentication in the Gbit/second rates," *Fast Software Encryption, 4th International Workshop, FSE'97, LNCS Vol. 1267, Springer-Verlag*, 1997.

[13] *Hitachi CMOS Cellbase IC HG73C/M Series Library*.

[14] Katz, J., Yung, M., ``Unforgeable Encryption and Chosen Cipher Secure Modes of Operation," *Fast Software Encryption, 7th International Workshop, FSE2000, Pre-proceedings*, 2000.

[15] Knuth, Donald E., ``*The Art of Computer Programming Volume II (2nd ed.)*," Addison-Wesley, 1981.

[16] Jakubowski, M. H., Venkatesan, R., ``The Chain & Sum Primitive and Its Applications to MACs and Stream Ciphers," *Advances in Cryptology, -EUROCRYPT'98, LNCS Vol. 1403, Springer-Verlag*, 1998.

[17] Menezes, Alfred J., van Oorschot, Paul C., Vanstone, Scott A., ``*HANDBOOK of APPLIED CRYPTOGRAPHY*," CRC Press, 1997.

[18] Rijmen, V., B. Rompay, B. V., Preneel, B., Vandewalle, J., ``Producing Collisions for PANAMA," Preproceedings of FSE2001, 8th Fast Software Encryption Worksop, Yokohama Japan, 2001.

[19] Roe, M., ``Cryptography and Evidence,'' *Doctoral Dissertation at the University of Cambridge*, 1997, available at **http://www.ccsr.cam. ac.uk/techreports/index.html**.
[20] Schneier, B., ``Block and Stream Ciphers,'' Crypto-Gram, January 15, 2000, available at **http://www.counterpane.com/crypto-gram-0001.html**.
[21] Shannon, C. E., ``A Mathematical Theory of Communication,'' *Bell System Technical Journal, Vol. 27, No. 4*, 1948.
[22] Shoup, V., ``On Fast and Provably Secure Message Authentication Based on Universal Hashing,'' preliminary version in *Advances in Cryptology, -CRYPTO'96, LNCS Vol. 1109, Springer-Verlag*, 1996.
[23] Taylor, R., ``An Integrity Check Value Algorithm for Stream Ciphers,'' *Advances in Cryptology, - CRYPTO'93, LNCS Vol. 773, Springer-Verlag*, 1993.
[24] Wegman, R., Carter, L., ``New hash functions and their use in authentication and set equality,'' *Journal of Computer and System Sciences, 22:265-279*, 1981.