Improving security of Vector Stream Cipher

Atsushi Iwasaki$^{1a)}$ and Ken Umeno$^1$

$^1$ Graduate School of Informatics, Kyoto University
Yoshidahonmachi, Sakyo-ku, Kyoto City, Japan

$a)$ iwasaki.atsushi.47e@st.kyoto-u.ac.jp

Received May 1, 2015; Revised August 27, 2015; Published January 1, 2016

Abstract: Vector Stream Cipher (VSC) is a stream cipher based on the chaos theory. The algorithm for generating stream keys is very simple and the encryption speed is very fast. Some theoretical attacks for VSC have been reported so far since the invention of VSC in 2004. In this paper, we improve the security of VSC and design a new cipher system “Vector Stream Cipher 2.0” so that the theoretical attacks cannot work. We show that the encryption speed of VSC 2.0 keeps more than 85% of that of VSC, and key-stream of VSC 2.0 has good randomness. The main result of this paper is that our proposed VSC 2.0 is shown to have provable security for attacks with linear masking. Because there is few cryptography based on the chaos theory which has proven security, VSC 2.0 is a rare example.

Key Words: stream cipher, chaos, pseudorandom

1. Introduction

Recently, the Internet is more used and more and more data which must be communicated securely is generated. A cipher system is needed to be secure and being performed very fast. One of such cipher systems is Vector Stream Cipher (VSC) [1]. VSC is a stream cipher based on the chaos theory. One of the authors at Communication Research Laboratory which is now called the National Institute of Information and Communications Technology developed VSC in 2004. Encryption speed of VSC is very fast. By hardware implementation, it has recorded 25Gbps. In addition, the algorithm of VSC is very simple, so it can be programmed very easily whether it is software or hardware.

The security of VSC has been investigated by several researchers [2–5]. Although an attack breaking VSC practically is not reported, some theoretical attacks including distinguishing attack with linear masking are realized. In this paper, we improve the security of VSC in order to avoid their attacks and design a new cipher system, which we call “Vector Stream Cipher 2.0”. Although there are some cryptographies based on the chaos theory, except few examples [6], the security is not evaluated. VSC 2.0, therefore, is important for information security.

This paper is constructed as follows. In section 2, we introduce the algorithm of the original VSC. In section 3, some attacks for VSC are introduced. In section 4, we propose methods of improving security of VSC. In section 5, we perform some experiments to investigate the security of the methods. Finally, we conclude this paper.
2. Vector Stream Cipher

In this section, we introduce VSC128, which is VSC with 128-bit secret key and 128-bit initial vector. The algorithm of VSC128 has been published [1]. It is given as follows:

1. Assume $A$, $B$, $C$, $D$, $X$, $Y$, $Z$ and $W$ 32-bit integer variables. Assign a secret key to $A$, $B$, $C$ and $D$, and an initial vector to $X$, $Y$, $Z$ and $W$.

2. Repeat the following operation 8 times. (In this paper, we call the operation “round”.)

(a) Assume $a$, $b$, $c$, $d$, $x$, $y$, $z$ and $w$ 32-bit integer variables. Calculate the values of $a$, $b$, $c$, $d$, $x$, $y$, $z$ and $w$ as follows.

\[
\begin{align*}
a &= A - (A \mod 4) + 1, \\
b &= B - (B \mod 4) + 1, \\
c &= C - (C \mod 4) + 1, \\
d &= D - (D \mod 4) + 1, \\
x &= X - (X \mod 4) + 1, \\
y &= Y - (Y \mod 4) + 1, \\
z &= Z - (Z \mod 4) + 1, \\
w &= W - (W \mod 4) + 1.
\end{align*}
\]

In this paper, we regard “mod” as modulus operator.

(b) Assume $A'$, $B'$, $C'$, $D'$, $X'$, $Y'$, $Z'$ and $W'$ 32-bit integer variables. Calculate the values of $A'$, $B'$, $C'$, $D'$, $X'$, $Y'$, $Z'$ and $W'$ as follows.

\[
\begin{align*}
A' &= A(2A + y) \mod 2^{32}, \\
B' &= B(2B + x) \mod 2^{32}, \\
C' &= C(2C + z) \mod 2^{32}, \\
D' &= D(2D + w) \mod 2^{32}, \\
X' &= X(2X + c) \mod 2^{32}, \\
Y' &= Y(2Y + d) \mod 2^{32}, \\
Z' &= Z(2Z + a) \mod 2^{32}, \\
W' &= W(2W + b) \mod 2^{32}.
\end{align*}
\]

(c) Regard $(A', B', C', D', X', Y', Z', W')$ as a 256-bit sequence, and perform 5-bit left rotational shift. After that, copy the sequence to $(A, B, C, D, X, Y, Z, W)$. Writing mathematically,

\[
\begin{align*}
A &= (A' << 5) \oplus (B' >> 27) \mod 2^{32}, \\
B &= (B' << 5) \oplus (C' >> 27) \mod 2^{32}, \\
C &= (C' << 5) \oplus (D' >> 27) \mod 2^{32}, \\
D &= (D' << 5) \oplus (X' >> 27) \mod 2^{32}, \\
X &= (X' << 5) \oplus (Y' >> 27) \mod 2^{32}, \\
Y &= (Y' << 5) \oplus (Z' >> 27) \mod 2^{32}, \\
Z &= (Z' << 5) \oplus (W' >> 27) \mod 2^{32}, \\
W &= (W' << 5) \oplus (A' >> 27) \mod 2^{32}.
\end{align*}
\]

where “\(<\ll\)” and “\(\gg\)” mean simple bit-shift.

3. Assume $D1$, $D2$, $D3$ and $D4$ 32-bit plaintexts and $E1$, $E2$, $E3$ and $E4$ are the corresponding ciphertexts respectively. Then, calculate the values of $E1$, $E2$, $E3$ and $E4$ as follow.

\[
\begin{align*}
E1 &= D1 \oplus X, \\
E2 &= D2 \oplus Y, \\
E3 &= D3 \oplus Z, \\
E4 &= D4 \oplus W.
\end{align*}
\]

4. Repeat step 2 and 3 until all given plaintexts are encrypted.
3. Attacks for VSC128

In this section, we introduce some attacks for VSC128 or weakness points of VSC128.

3.1 Distinguishing attack with linear masking

Linear probability \((LP)\) of a function \(f_K\) is defined as

\[
LP(\Gamma X, \Gamma Y) := \left( 2^{-\#\{(\bar{X}, K) | \bar{X} \cdot \Gamma \bar{X} = f_K(\bar{X}) \cdot \Gamma Y\}} - 1 \right)^2,
\]

where \(\Gamma X, \Gamma Y, \bar{X}\) and \(K\) are integers which are expressed as \(n\)-bit integers. The \(K\) corresponds to a key. The \(\Gamma X\) and \(\Gamma Y\) are called “Linear-Mask”. The operation “\(\cdot\)" means the inner product over \(\mathbb{GF}(2)\).

Assume \(g_{K_1,K_2} := f_{K_1} \circ f_{K_2}\). Linear characteristic probability with considering multi pass \((LCPM)\) of \(g_{K_1,K_2}\) is defined as

\[
LCPM(\Gamma X, \Gamma Z) := \sum_{\Gamma Y} LP(\Gamma X, \Gamma Y) LP(\Gamma Y, \Gamma Z).
\]

The fundamental function of VSC128 with a key \(K\) is described as

\[
f_K(\bar{X}) = \bar{X}(2 \bar{X} + K \mod 4 + 1) \mod 2^{32},
\]

where \(\bar{X}\) and \(K\) are expressed as 32-bit integers. The value of \(LP\) of the fundamental function is studied [4]. The value is

\[
LP(\Gamma X, \Gamma Y) = \begin{cases} 
1 & \text{for } (\Gamma X, \Gamma Y) = (1,1), (2,3) \text{ or } (3,2) \\
2^{-2\text{maxbit}(\Gamma X)+4} & \text{for } \text{maxbit}(\Gamma X) = \text{maxbit}(\Gamma Y) > 3 \text{ and } [\Gamma X]_{\text{maxbit}(\Gamma X)} = [\Gamma Y]_{\text{maxbit}(\Gamma Y)} \\
0 & \text{otherwise}
\end{cases}
\]

where \([\alpha]_i\) means the \(i\)-th least significant bit of \(\alpha\) and \(\text{maxbit}(\alpha) := \max\{i | [\alpha]_i = 1\}\). It was also studied that the maximum value of \(LCPM\) with 8 rounds is \(2^{-115}\) [5]. Because a secret key of VSC128 is 128-bit, a distinguishing attack is realized.

3.2 Chosen initial vector attack

It is reported that the output sequence (key-stream) of VSC128 have a statistical deviation if the initial vector is chosen among specific vectors, and so the distinguishing attack is realized if an attacker can chose an initial vector intentionally [3]. More concretely, the distinguishing attack is practical if there are \(2^{32}\) initial vectors whose 96 bits from the least significant bit are 0.

3.3 Collision of key stream

The round of VSC128 is not a bijection. For example, assume \(A, B, C, D, X, Y, Z\) and \(W\) odd numbers and \(\bar{A} := A \oplus 0x80000000, \cdots, \bar{W} := W \oplus 0x80000000\). The values of \((A, \cdots, W)\) and \((\bar{A}, \cdots, \bar{W})\) after the round are equal. The fact means that effective key length of VSC128 is smaller than 128-bit.

4. Improving security of VSC128

In this section, we propose improvement of VSC128 to avoid such attacks. We call the this improved VSC128 “Vector Stream Cipher 2.0” (VSC 2.0).
4.1 Increasing the repetition number of rounds

Since the maximum LCPM becomes smaller when repetition number of rounds increases, we try to calculate the maximum LCPM with more rounds than 8.

Assume \( LP_{\text{round}} \) means \( LP \) of the round, and \( GW_i = 1 << (i - 1) \) and \( GZ_j = 1 << (j - 1) \) are Linear-Masks which mask \( W \) and \( Z \) respectively on the round, where \( i \) and \( j \) are natural numbers which are smaller than 32. \( LP_{\text{round}} \) is calculated as follows:

\[
LP_{\text{round}}(\Gamma W_1, \Gamma) = \begin{cases} 
1 & \text{for } \Gamma = \Gamma W_6 \\
0 & \text{otherwise} 
\end{cases} 
\]

\[
LP_{\text{round}}(\Gamma W_2, \Gamma) = \begin{cases} 
1 & \text{for } \Gamma = \Gamma W_7 \oplus \Gamma W_6 \\
0 & \text{otherwise} 
\end{cases} 
\]

For \( 3 \leq i \leq 27 \),

\[
LP_{\text{round}}(\Gamma W_i \oplus \alpha_{i-2}, \Gamma) = \begin{cases} 
2^{-2i+4} & \text{for } \Gamma = \Gamma W_i+5 \oplus (\beta_{i-2} << 5) \\
0 & \text{otherwise} 
\end{cases} 
\]

where \( \alpha_{i-2} \) and \( \beta_{i-2} \) are arbitrary \((i - 2)\)bit integers. For \( i = 28 \) and \( 29 \),

\[
LP_{\text{round}}(\Gamma W_i \oplus \alpha_{i-2}, \Gamma) = \begin{cases} 
2^{-2i+4} & \text{for } \Gamma = \Gamma Z_{i-27} \\
0 & \text{otherwise} 
\end{cases} 
\]

For \( i = 30, 31 \) and \( 32 \),

\[
LP_{\text{round}}(\Gamma W_i \oplus \alpha_{i-2}, \Gamma) = \begin{cases} 
1 & \text{for } \Gamma = \Gamma Z_{i-27} \oplus \beta_{i-29} \\
0 & \text{otherwise} 
\end{cases} 
\]

\( LP_{\text{round}}(\Gamma Z_{i-1}, \cdot) \) is the same as the above.

Assume \( LCPM_k \) means \( LCPM \) with \( k \) rounds.

\[
LCPM_9(\Gamma W_3, \Gamma Z_{16}) = \sum_{\alpha_1, \alpha_4, \alpha_6, \alpha_{11}, \alpha_{16}, \alpha_{21}} [LP_{\text{round}}(\Gamma W_3, \Gamma W_8 \oplus (\alpha_1 << 5)) \\
LP_{\text{round}}(\Gamma W_8 \oplus (\alpha_1 << 5), \Gamma W_{13} \oplus (\alpha_6 << 5)) \\
LP_{\text{round}}(\Gamma W_{13} \oplus (\alpha_6 << 5), \Gamma W_{18} \oplus (\alpha_{11} << 5)) \\
LP_{\text{round}}(\Gamma W_{18} \oplus (\alpha_{11} << 5), \Gamma W_{23} \oplus (\alpha_{16} << 5)) \\
LP_{\text{round}}(\Gamma W_{23} \oplus (\alpha_{16} << 5), \Gamma W_{28} \oplus (\alpha_{21} << 5)) \\
LP_{\text{round}}(\Gamma W_{28} \oplus (\alpha_{21} << 5), \Gamma Z_1) \\
LP_{\text{round}}(\Gamma Z_1, \Gamma Z_6) \\
LP_{\text{round}}(\Gamma Z_6, \Gamma Z_{11} \oplus (\alpha_4 << 5)) \\
LP_{\text{round}}(\Gamma Z_{11} \oplus (\alpha_4 << 5), \Gamma Z_{16})] \\
= 2^{-129}.
\]

For more detail, see the Table I.

We repeat similar calculation, and get that \( 2^{-129} \) is the largest value of \( LCPM_9 \). It means that the linear attack is not practical, so we change the repetition number of rounds at step 2 of VSC128’s algorithm from 8 to 9 in the specification of VSC 2.0.

4.2 Preprocessing

If we replace an initial vector with its hash value when the initial vector is given, we can avoid chosen initial vector attack because an attacker cannot calculate an inverse of a hash value effectively. Because the attack [3] needs \( 2^{32} \) initial vectors whose 96 bits from the least significant bit of initial vector are 0 lead to the statistical deviation, an attacker who will perform the attack is needed to calculate the hash function \( 2^{96} \times 2^{32} = 2^{128} \) times.
Although there are many hash functions in the world, we design a new hash function with the round because we should keep the program small. The algorithm of the new hash function is as follows.

1. Set \( A = 0x\text{fedcba}098, B = 0x01234567, C = 0x89abcdef \) and \( D = 0x76543210. \)
2. Assign a given initial vector to \( X, Y, Z \) and \( W. \)
3. Perform the round 30 times.
4. Output \( X, Y, Z \) and \( W. \)

Because any person who does not know the given initial vector cannot know the values of \( A, B, C \) and \( D \) at step 4 of the above algorithm, it is difficult for him to calculate the value of the given initial vector even if he know the output. Then, we expect that the algorithm can be regarded as a hash function. We, therefore, add the operation replacing a given initial vector with its hash value with the algorithm to VSC128.

If the values of \( A, B, C \) and \( D \) at step 1 are all “\( 0 \)”, the preprocessing is weak for a preimage attack. The value of \( D \) must not be odd number and the reason is shown in the next section. Then, we choose \( A = 0x\text{fedcba}098, B = 0x01234567, C = 0x89abcdef \) and \( D = 0x76543210. \) The reputation number at step 3 is chosen to be safe enough. We confirm experimentally it at section 5.

### 4.3 Avoiding collision

First, we prove the following theorem.

**Theorem** Consider a map \( g: (\mathbb{Z}/2^n\mathbb{Z})^m \to (\mathbb{Z}/2^n\mathbb{Z})^m \), which is described as

\[
g(A_0, A_1, \ldots, A_{m-1}) = (A'_0, A'_1, \ldots, A'_{m-1}),
\]

\[
A'_i = A_i + a(A_{i+1} \mod m) \mod 2^n \text{ } (\forall i \in \mathbb{Z}/2^m\mathbb{Z}),
\]

where \( A_1, \ldots, A_m \) and \( A'_1, \ldots, A'_m \) are elements of \( \mathbb{Z}/2^n\mathbb{Z} \) and \( a(A_i) = A_i - (A_i \mod 4) + 1 \) (\( \forall i \in \mathbb{Z}/2^m\mathbb{Z} \)). Assume \( O_n \) a subset of \( \mathbb{Z}/2^n\mathbb{Z} \), which is constructed of odd numbers in \( \mathbb{Z}/2^n\mathbb{Z} \). Then, if we restrict the domain of \( g \) to \( (\mathbb{Z}/2^n\mathbb{Z})^m \) except \( (O_n)^m \), \( g \) becomes a bijective map on \( (\mathbb{Z}/2^n\mathbb{Z})^m \backslash (O_n)^m \).

**Proof** Since \( g(X) \in (\mathbb{Z}/2^n\mathbb{Z})^m \backslash (O_n)^m \) for all \( X \in (\mathbb{Z}/2^n\mathbb{Z})^m \backslash (O_n)^m \) obviously, \( g \) can be regarded as a map from \( (\mathbb{Z}/2^n\mathbb{Z})^m \backslash (O_n)^m \) to \( (\mathbb{Z}/2^m\mathbb{Z})^m \backslash (O_n)^m \). Then, we prove injectivity.

Assume \( (A_0, A_1, \ldots, A_{m-1}) \) and \( (\tilde{A}_0, \tilde{A}_1, \ldots, \tilde{A}_{m-1}) \) elements of \( (\mathbb{Z}/2^m\mathbb{Z})^m \backslash (O_n)^m \) satisfying \( (A_0, A_1, \ldots, A_{m-1}) \neq (\tilde{A}_0, \tilde{A}_1, \ldots, \tilde{A}_{m-1}) \). There are integers \( 0 \leq s_i \leq n \) and \( 2 \leq t_i \leq n \) (\( i \in \mathbb{Z}/2^m\mathbb{Z} \)) such that

\[
(s_0, s_1, \ldots, s_{m-1}, t_0, t_1, \ldots, t_{m-1}) \neq (n, n, \ldots, n),
\]

\[
s_i \leq t_i \text{ } (\forall i \in \mathbb{Z}/2^m\mathbb{Z}),
\]

\[
\tilde{A}_i = A_i + (2^{q_i} - 1)2^{s_i} \mod 2^n \text{ } (\forall i \in \mathbb{Z}/2^m\mathbb{Z}),
\]

\[
a(\tilde{A}_i) = a(A_i) + (2^{q_i} - 1)2^{s_i} \mod 2^n \text{ } (\forall i \in \mathbb{Z}/2^m\mathbb{Z}),
\]
where \( p_i \) and \( q_i \) are natural numbers.

Assume \((A_0', A_1', \ldots, A_{m-1}') = g(A_0, A_1, \ldots, A_{-1})\) and \((\tilde{A}_0', \tilde{A}_1', \ldots, \tilde{A}_{m-1}') = g(\tilde{A}_0, \tilde{A}_1, \ldots, \tilde{A}_{m-1})\). To simplify notation, we define new notations \(A_m, \tilde{A}_m, t_m\) and \(q_m\) as \(A_0, \tilde{A}_0, t_0\) and \(q_0\) respectively. Then,

\[
\tilde{A}_{i}' = \tilde{A}_i \{ 2\tilde{A}_i + a(A_{i+1}) \} \mod 2^n
\]

\[
= \{ A_i + (2p_i - 1)2^{s_i} \} \{ 2A_i + (2p_i - 1)2^{s_i} + 1 + a(A_{i+1}) + (2q_{i+1} - 1)2^{t_{i+1}} \} \mod 2^n
\]

Case 1: There are \( i, j \in \mathbb{Z}/2^m\mathbb{Z} \) such that \( s_i \neq t_j \). In this case, there is \( k \in \mathbb{Z}/2^m\mathbb{Z} \) such that \( s_k < t_{k+1} \leq n \), because \( s_i \leq t_i \) \((\forall i \in \mathbb{Z}/2^m\mathbb{Z})\). Then,

\[
(2r_k - 1)2^{s_k} + A_k(2q_{k+1} - 1)2^{t_{k+1}} \mod 2^n
\]

\[
\text{and}
\]

\[
(2r_k - 1)2^{s_k} + A_k(2q_{k+1} - 1)2^{t_{k+1}} \mod 2^n
\]

Then, the \((s_k + 1)\)-th least bit of \(A_k'\) is different from that of \(A_k'\).

Case 2: \( s_0 = s_1 = \cdots = s_{m-1} \neq t_0 = t_1 = \cdots = t_{m-1} < n \). In this case, there are \( k \in \mathbb{Z}/2^m\mathbb{Z} \) and \( B \in \mathbb{Z}/2^{n-1}\mathbb{Z} \) such that \( A_k = 2B \). Then,

\[
(2r_k - 1)2^{s_k} + A_k(2q_{k+1} - 1)2^{t_{k+1}} \mod 2^n
\]

\[
= (2r_k - 1)2^{s_k} + 2B(2q_{k+1} - 1)2^{s_k} \mod 2^n
\]

\[
= [2(r_k + B(2q_{k+1} - 1)) - 1]2^{s_k} \mod 2^n.
\]

Since \( s_k < n \), \(A_k' \neq A_k'\).

Therefore, \((\tilde{A}_0', \tilde{A}_1', \ldots, \tilde{A}_{m-1}') \neq (A_0', A_1', \ldots, A_{m-1}')\). \(\square\)

Using the theorem, the round becomes a bijection if we restrict the domain to the case that at least one of \(A, B, C, D, X, Y, Z\) or \(W\) is even. Then, we introduce a new rule, “Keep the value of \(D\) is even”. To keep the new rule, we add to the algorithm two modifications as follows.

- Change the length of secret key from 128-bit to 127-bit, and assign secret key to \(A, B, C\) and significant 31-bit of \(D\) at step 1 of the VSC128’s algorithm. Set the least bit of \(D\) ‘0’ simultaneously.

- At step 2(b) of the algorithm, if \(D\) is even, \(D'\) is also even. Then, change the step 2(c) as follows, and we can keep \(D\) even after performing the round.

\[
A = (A' << 5) \oplus (B' >> 27) \mod 2^{32}, \quad B = (B' << 5) \oplus (C' >> 27) \mod 2^{32},
\]

\[
C = (C' << 5) \oplus (D' >> 27) \mod 2^{32}, \quad D = (D' << 5) \oplus ((X') >> 27) << 1 \mod 2^{32},
\]

\[
X = (X' << 5) \oplus (Y' >> 27) \mod 2^{32}, \quad Y = (Y' << 5) \oplus (Z' >> 27) \mod 2^{32},
\]

\[
Z = (Z' << 5) \oplus (W' >> 27) \mod 2^{32}, \quad W = (W' << 5) \oplus (A' >> 27) \mod 2^{32}.
\]

5. Experiments

In this section, we perform some experiments for VSC 2.0.
5.1 Speed
We measure the speeds of performing VSC128, VSC 2.0 and AES-128. The environment in which we measure is shown in Table II. The detail of measuring is follow.

1. Select a key (and an initial vector) randomly.
2. Reserve 1MB arrays on main memory, and write plaintext which is generated randomly in the arrays.
3. Encrypt the plaintext and write the ciphertext in the arrays. Measure a time performing this step.
4. Repeat step 1-3 1000 times, and calculate the average of times which are measured at step 3.

As results, we got Table III. The speed of performing VSC 2.0 is more than 85% of the speed of performing VSC128, and it is 280% of the speed of performing AES-128.

| Algorithm     | Speed (Mbps) |
|---------------|--------------|
| VSC 2.0       | 1039.222464  |
| VSC128        | 1202.254889  |
| AES-128 ECB   | 366.901621   |

5.2 Property of the preprocessing
We investigate an properties of the preprocessing defined in 4.2.

1. Select an input randomly. (We call the input $I_1$.)
2. Select a bit of $I_1$ and revers the bit. (We call the value $I_2$.)
3. Apply the preprocessing to $I_1$ and $I_2$. (We call the outputs $I'_1$ and $I'_2$ respectively.)
4. Measure Haming distance between $I'_1$ and $I'_2$.
5. Repeat step 1-4 1000000 times. Calculate the average of Haming distance which are measure at step 3.

As a result, the value of the average Haming distance was 64.000107. Since an output length of the preprocessing is 128bit, the result shows that the preprocessing has a good property.

5.3 Randomness of key stream
We performed randomness test described by NIST SP800-22 [7] for key streams generated by VSC 2.0. The test was performed for 11 sets. Each set is constructed of 1000 sequences. (Exceptionally, set 10 and 11 are constructed of 255 sequences respectively. A sequence of set 10 is generated by VSC 2.0 with an initial condition (key and initial vector) whose one bit is “1” and the others are “0”. VSC 2.0 requires the least bit of $D$ is “0”. Then, set 10 is constructed of only 255 sequences. A sequence of set 11 is generated by VSC 2.0 with an initial condition whose two bits are “0” and the others are “1”. One of the two is the least bit of $D$. Then, set 11 is also constructed of only 255 sequences.) Each sequence is constructed of 1000000bits, which are generated by VSC 2.0 with a secret key and
an initial vector. The secret key and the initial vector are chosen randomly, but random pattern is dependent on a set. For reference, we perform the same experiments for VSC128. Table IV shows the results. The randomness test is constructed of 188 test items. Even if sequences are exactly random, there are cases that the sequence does not pass all test items. Therefore, the result shows that there is no problem about randomness of the key stream of VSC 2.0. By the same reason, the results do not mean that the randomness of key streams generated by VSC 2.0 is worse than those of key streams generated by VSC128.

| Set | Numbers of the passed test items |
|-----|---------------------------------|
|     | VSC 2.0 | VSC128 |
| 1   | 188      | 188    |
| 2   | 187 *    | 188    |
| 3   | 186 *    | 188    |
| 4   | 188      | 187 *  |
| 5   | 188      | 187 *  |
| 6   | 187 *    | 188    |
| 7   | 187 *    | 188    |
| 8   | 188      | 188    |
| 9   | 188      | 188    |
| 10  | 188      | 188    |
| 11  | 188      | 188    |

6. Conclusion
The original VSC was developed based on the chaos theory. Some theoretical attacks for VSC have been reported so far. Here, we proposed VSC 2.0 which is based on VSC, and it dramatically improves the security of VSC. As a main result, the security of VSC 2.0 for the linear attack is provable. The encryption speed of VSC 2.0 is very fast because of the simplicity of the algorithm, although it is little bit slower than that of the original VSC. VSC 2.0 can be also used as a pseudorandom numbers generator. We think that chaos encryption algorithm based on the chaos theory is of important class of encryption algorithm. VSC 2.0 is a rare chaotic encryption algorithm with proven security. We should proceed further in investigating proven security properties such as the distribution of periods of key stream for the chaotic encryption algorithm VSC 2.0.

References
[1] K. Umeno, S. Kim, and A. Hasegawa, “128bit VSC Specification,” http://www.chaosware.com/vsc128.pdf, 2004
[2] H. Tanaka, K. Nemoto, T. Miki, and M. Sato, “Security evaluation of 128bit VSC,” Technical Report of IEICE, ISEC2003-144, pp. 179–184, 2004.
[3] Y. Tuno, T. Saito, K. Myao, T. Suzaki, and T. Kawabata, “Distinguishing attack with chosen initial vector against VSC128,” SASC Workshop, pp. 212–219, 2004.
[4] M. Nakamura, T. Nosaka, and T. Kaneko, “A study on the linear property of basic function in VSC128,” Proc. ESS Conf. IEICE, p. 126, 2004.
[5] T. Nosaka, M. Nakamura, and T. Kaneko, “A study on linear cryptanalysis of VSC128,” Proc. ESS Conf. IEICE, p. 127, 2004.
[6] H. Okutomi and K. Nakamura, “On initial value estimating method of the preudo random bit sequence generated by tent-type map with finite precision,” IEICE Trans. Fundamentals (Japanese Edition), vol. J93-A, no. 6, pp. 414–422, 2010.
[7] NIST, “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” Special Publication 800-22 Revision 1a.