A Word-Oriented Stream Cipher
Using Clock Control

Shinsaku Kiyomoto¹, Toshiaki Tanaka¹, and Kouichi Sakurai²

¹ KDDI R & D Laboratories Inc.
2-1-15 Ohara, Fujimino-shi, Saitama 356-8502, Japan
{kiyomoto, tl-tanaka}@kddilabs.jp
² Dept. of CSCE, Kyushu University
744 Motooka, Nishi-ku, Fukuoka-shi, Fukuoka 819-0395, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. Various clock-controlled stream ciphers and attacks have been described in a number of papers. However, few word-oriented algorithms with an irregular clocking mechanism have been proposed. This paper proposes a design for word-oriented stream ciphers using dynamic feedback control and show analysis results of its security and performance. The stream cipher K2 is a secure and high-performance clock controlled stream cipher. We believe that the dynamic feedback control mechanism is potentially effective against several attacks, not only existing attacks but also a novel attack.

Keywords: Stream Cipher, Irregular Clock, Dynamic Feedback Control

1 Introduction

Stream ciphers are used extensively to provide a reliable and efficient method for communicating securely. A basic stream cipher uses several independent linear feedback shift registers (LFSRs) together with nonlinear functions in order to produce a keystream. The keystream is then XORed with a plaintext to produce a ciphertext. Some stream ciphers use a general nonlinear function to clock one or more LFSR(s) irregularly. Various clock-controlled stream ciphers and attacks on them have been proposed. J. D. Golic proposed linear models for clock controlled shift registers and discussed correlation attacks on clock controlled shift register based keystream generators[11]. Clock-controlled stream ciphers are classified into two main types of stream ciphers: the A5 family and the LILI family. A5[4] is a well-known stream cipher designed to ensure the confidentiality of mobile communications. Alpha-1 [14] structures similar to those in A5.
LILI-like stream ciphers, such as LILI-128 [16] and LILI-II [6], have two different LFSRs for providing bits for clocking and keystream bits. One LFSR clocks regularly, providing input for a clock controller, and another LFSR clocks irregularly, providing keystream.

Recently, word-oriented stream ciphers have been developed in order to improve the performance of software implementations. In the NESSIE project, many word-oriented stream ciphers were proposed, such as SNOW [9] and SOBER[15], and demonstrated good performance in software. However, few word-oriented algorithms with an irregular clocking mechanism have been proposed because of inefficiency of clock control mechanism for software implementation. This paper proposes a new word-oriented stream cipher using dynamic feedback control as irregular clocking. Proposed stream cipher has a dynamic feedback control mechanism for a byte-wise feedback function of a LFSR, and realizes fast encryption/decryption for software implementation. We presents a stream cipher algorithm and show analysis results of its security and performance.

2 Design of Clock Control for Word-Oriented Stream Ciphers

The clock control mechanism of a stream cipher generally either controls LFSR clocking or shrinks or thins output. A clock control that shrinks or thins output reduces the performance of the stream cipher, because some output bits are discarded. If one applies shrinking to a word-oriented stream cipher, the performance is markedly reduced. On the other hand, the bit-oriented clock control mechanism for updating an LFSR is also inefficient, when the mechanism controls the LFSR for each register.

A dynamic feedback control for a LFSR is effective method to improve security of stream ciphers. The stream cipher MICKEY-128[2] has a dynamic feedback control mechanism for a bit-wise LFSR. POMARANCH[13] uses a cascade jump controlled sequence generator to modify the feedback function.

We propose a stream cipher design that operates on words and has an efficient dynamic feedback control as irregular clocking. The basic idea of the design is to modify a mixing operation during the state update. Feedback polynomials for word-oriented LFSR are described with coefficients; multiplying an input word by a coefficient means mixing the words. A typical example is a LFSR of SNOW 2.0[10]. Generally, the coefficients are selected such that the feedback polynomial is a primitive polynomial. We apply irregular clocking for this mixing operation, and this modifica-
tion causes only a minimal decrease in the encryption/decryption speed. That is, at least one LFSR is irregularly clocked to dynamically modify the feedback function to the dynamic feedback controller that receives the outputs of the other LFSRs. For example, the feedback function is defined as \( s_{t+a} = \alpha_0^{0,1} s_{t+b} \oplus \alpha_1^{0,1} s_{t+c} \oplus \alpha_2^{0,1} s_{t+d} \), where \( \{0, 1\} \)s are selected by the dynamic feedback controller.

The dynamic feedback control mechanism improves the security of a stream cipher because it changes a deterministic linear recurrence of some registers into a probabilistic recurrence. This property effectively protects against several attacks. An attacker has to obtain a linear recurrence of the keystream derived from the linear recurrence of some registers. By irregular modification, the linear recurrence exists with a low probability. An attacker has to guess some inputs to the non-linear function for some attacks; however, irregular modification makes it impossible: the attacker has to guess the inputs to the dynamic feedback controller first. Thus, irregular modification of the feedback function improves the security of the stream cipher.

We think that a dynamic feedback control mechanism is potentially effective against several attacks, not only existing attacks but also a novel attack.

### 3 Stream Cipher K2

In this section, we describe an stream cipher algorithm K2 that has a clock control mechanism.

#### 3.1 Linear Feedback Shift Registers

The K2 stream cipher consists of two linear feedback shift registers (LFSRs), \texttt{LFSR}-\texttt{A} and \texttt{LFSR}-\texttt{B}, a non-linear function with two internal registers \texttt{M1} and \texttt{M2}, and a dynamic feedback controller as shown in Fig. 1. The size of each register is 32 bits. \texttt{LFSR}-\texttt{A} has five registers, and \texttt{LFSR}-\texttt{B} has 11 registers. Let \( \beta \) be the roots of the primitive polynomial:

\[
x^8 + x^7 + x^6 + x + 1 \in GF(2)[x]
\]

A byte string \( y \) denotes \((y_7, y_6, \ldots, y_1, y_0)\), where \( y_7 \) is the most significant bit and \( y_0 \) is the least significant bit. \( y \) is represented by

\[
y = y_7 \beta^7 + y_6 \beta^6 + \ldots + y_1 \beta + y_0
\]
In the same way, let $\gamma, \delta, \zeta$ be the roots of the primitive polynomials,

\[
x^8 + x^5 + x^3 + x^2 + 1 \in GF(2)[x]
\]
\[
x^8 + x^6 + x^3 + x^2 + 1 \in GF(2)[x]
\]
\[
x^8 + x^6 + x^5 + x^2 + 1 \in GF(2)[x]
\]

respectively.
Let $\alpha_0$ be the root of the irreducible polynomial of degree four

\[
x^4 + \beta^{24} x^3 + \beta^3 x^2 + \beta^{12} x + \beta^{71} \in GF(2^8)[x]
\]

A 32-bit string $Y$ denotes $(Y_3, Y_2, Y_1, Y_0)$, where $Y_i$ is a byte string and $Y_3$ is the most significant byte. $Y$ is represented by

\[
Y = Y_3 \alpha_0^3 + Y_2 \alpha_0^2 + Y_1 \alpha_0 + Y_0
\]

Let $\alpha_1, \alpha_2, \alpha_3$ be the roots of the irreducible polynomials of degree four

\[
x^4 + \gamma^{230} x^3 + \gamma^{156} x^2 + \gamma^{93} x + \gamma^{29} \in GF(2^8)[x]
\]
\[
x^4 + \delta^{34} x^3 + \delta^{16} x^2 + \delta^{99} x + \delta^{248} \in GF(2^8)[x]
\]
\[
x^4 + \zeta^{157} x^3 + \zeta^{253} x^2 + \zeta^{56} x + \zeta^{16} \in GF(2^8)[x]
\]

respectively.

The feedback polynomials $f_A(x)$, and $f_B(x)$ of $LFSR-A$ and $LFSR-B$, respectively, are as follows;

\[
f_A(x) = \alpha_0 x^5 + x^2 + 1 \in GF(2^{32})[x]
\]
\[
f_B(x) = (\alpha_1^{d_1} + \alpha_2^{d_1} \alpha_3^{d_1} - 1) x^{11} + x^{10} + x^5 + \alpha_3^{d_2} x^3 + 1 \in GF(2^{32})[x]
\]

Let $d_1$ and $d_2$ be the sequences describing the output of the dynamic feedback controller. The outputs at time $t$ are defined in terms of $LFSR-A$. Let $A(x)$ denote the output of $LFSR-A$ at time $x$, and $A(x)[y] = \{0, 1\}$ denote the $y$th bit of $A(x)$, where $A(x)[31]$ is the most significant bit of $A(x)$. Then $d_1$ and $d_2$ (we call clock control bits) are described as follows;

\[
c_{l+1} = A_{l+2}[30], \quad c_{l+2} = A_{l+2}[31]
\]

Both $c_{l+1}$ and $c_{l+2}$ are integer sequences; more precisely, $c_{l+1}, c_{l+2} = \{0, 1\}$. Stop-and-go clocking is effective in terms of computational cost, because no computation is required in the case of 0. However, the feedback
function has no transformation for feedback registers with the probability 1/4 where all clockings are stop-and-go clockings. Thus, we use two types of clocking for the feedback function. \emph{LFSR-B} is defined by a primitive polynomial, where \( d2_k = 0 \).

### 3.2 Nonlinear Function

The non-linear function of K2 is fed the values of the two registers of \emph{LFSR-A} and the four registers of \emph{LFSR-B} and that of internal memories \( R1, R2, L1, L2 \), and outputs 64 bits of keystream every cycle. Fig. 2 shows the non-linear function of K2. The nonlinear function includes four substitution steps that are indicated by \emph{Sub}.

The \emph{Sub} step divides the 32-bit input string into four 1-byte strings and applies a non-linear permutation to each byte using an 8-to-8 bit substitution, and then applies a 32-to-32 bit linear permutation. The 8-to-8 bit substitution is the same as s-boxes of AES [8], and the permutation is the same as AES \emph{Mix Column} operation. The 8-to-8 bit substitution
consists of two functions: \( g \) and \( f \). The \( g \) calculates the multiplicative inverse modulo the irreducible polynomial \( m(x) = x^8 + x^4 + x^3 + x + 1 \) without \( 0x00 \), and \( 0x00 \) is transformed to itself \( (0x00) \). \( f \) is an affine transformation defined by:

\[
\begin{bmatrix}
    b_7 \\
    b_6 \\
    b_5 \\
    b_4 \\
    b_3 \\
    b_2 \\
    b_1 \\
    b_0
\end{bmatrix} =
\begin{bmatrix}
    11110000 \\
    01111100 \\
    00111110 \\
    00011111 \\
    10001111 \\
    11000111 \\
    11100011 \\
    11110001
\end{bmatrix}
\times
\begin{bmatrix}
    a_7 \\
    a_6 \\
    a_5 \\
    a_4 \\
    a_3 \\
    a_2 \\
    a_1 \\
    a_0
\end{bmatrix} \oplus \begin{bmatrix}
    0 \\
    1 \\
    1 \\
    0 \\
    0 \\
    0 \\
    1 \\
    1
\end{bmatrix}
\]

where \( a = (a_7, ..., a_0) \) is the input and \( b = (b_7, ..., b_0) \) is an output, and \( a_0 \) and \( b_0 \) are the least significant bit (LSB).

Let \( C \) be \((c_3, c_2, c_1, c_0)\) and output \( D \) be \((d_3, d_2, d_1, d_0)\), where \( c_i \), \( d_i \) are 8-bit values. The linear permutation \( D = p(C) \) is described as follows:

\[
\begin{bmatrix}
    d_0 \\
    d_1 \\
    d_2 \\
    d_3
\end{bmatrix} =
\begin{bmatrix}
    02 & 03 & 01 & 01 \\
    01 & 02 & 03 & 01 \\
    01 & 01 & 02 & 03 \\
    03 & 01 & 01 & 02
\end{bmatrix}
\begin{bmatrix}
    c_0 \\
    c_1 \\
    c_2 \\
    c_3
\end{bmatrix}
\]

in \(GF(2^8)\) of the irreducible polynomial \( m(x) = x^8 + x^4 + x^3 + x + 1\).

### 3.3 Keystream Output

Let keystream at time \( t \) be \( Z_t = (z_t^L, z_t^H) \) (each \( z_t^L \) is a 32-bit value, and \( z_t^H \) is a higher string). The keystream \( z_t^L \), \( z_t^H \) is calculated as follows:

\[
\begin{align*}
    z_t^L &= B_t \oplus R2_t \oplus R1_t \oplus A_{t+4} \\
    z_t^H &= B_{t+10} \oplus L2_t \oplus L1_t \oplus A_{t}
\end{align*}
\]

where \( A_x \) and \( B_x \) denote outputs of \( LFSR-A \) and \( LFSR-B \) at time \( x \), and \( R1_x, R2_x, L1_x, \) and \( L2_x \) denote the internal memory registers at time \( x \), and symbols \( \oplus \), \( \boxplus \) denote bitwise exclusive-or operation and 32-bit addition respectively. Finally, the internal memory registers are updated as follows:

\[
\begin{align*}
    R1_{t+1} &= Sub(L2_t \boxplus B_{t+9}), \quad R2_{t+1} = Sub(R1_t) \\
    L1_{t+1} &= Sub(R2_t \boxplus B_{t+4}), \quad L2_{t+1} = Sub(L1_t)
\end{align*}
\]
where \( \text{Sub}(X) \) is a output of the \( \text{Sub} \) step for \( X \). The set of \( \{B_t, B_{t+3}, B_{t+8}, B_{t+10}\} \) is a Full Positive Difference Set (FPDS).

### 3.4 Initialization Process

K2's initialization process consists of two steps, a key loading step and an internal state initialization step. First, an initial internal state is generated from a 128-bit initial key or a 256-bit initial key, and a 256-bit initial vector (IV) by using the key scheduling algorithm. The key scheduling algorithm is the same as the round key generation function of AES and the algorithm extends the 128-bit initial key or the 256-bit initial key to 512 bits. The key scheduling algorithm for a 128-bit key is described as:

\[
K_i = \begin{cases} 
    IK_i & (0 \leq i \leq 3) \\
    K_{i-4} \oplus s((K_{i-1} \ll 8) \oplus (K_{i-1} \gg 24)) \oplus Rcon[i/4 - 1] & (i = 4n) \\
    K_{i-4} \oplus K_{i-1} & (i \neq 4n)
\end{cases}
\]
where $IK = (IK_0, IK_1, IK_2, IK_3)$ is the initial key, $x^{i-1}$ is an element of $GF(2^8)$ generated by the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$, $i$ is a positive integer $0 \leq i \leq 15$, and $n$ is a positive integer. The function $s(X)$ is a 32-bit substitution that consists of four 8-to-8 substitutions for each byte. $Rcon[i]$ denotes $(x^i \mod x^8 + x^4 + x^3 + x + 1, 0x00, 0x00, 0x00)$ and $x$ is $0x02$. The $x^{i-1}$ are pre-computed and the results are implemented as a table. The internal state is initialized with $K_i$ and $IV = (IV_0, IV_1, ..., IV_7)$ as follows:

**step1**: $A_m = K_m$ \((m = 0, ..., 4)\), \(B_m = K_{5+m}\) \((m = 0, ..., 10)\)

**step2**: \(R10 = IV_0, L10 = IV_1, R20 = IV_2, L20 = IV_3,\)

\[ B_5 = B_5 \oplus IV_4, B_6 = B_6 \oplus IV_5, B_7 = B_7 \oplus IV_6, B_8 = B_8 \oplus IV_7 \]

After the above processes, the cipher clocks 24 times \((j = 1, ..., 24)\), updating the internal states. The internal states $A_{j+4}$ $B_{j+10}$ are also updated as follows:

\[ A_{j+4} = a_0 A_{j-1} \oplus A_{j+2} \oplus z_{j-1}^L \]

\[ B_{j+10} = (a_1 c_{j-1}^{B_{j-1}} + a_2 c_{j-1}^{B_{j-1}} - 1) B_{j-1} \oplus B_j \oplus B_{j+5} \oplus a_3 c_{j-1}^{B_{j+1}} B_{j+7} \oplus z_{j-1}^H \]

The recommended maximum number of cycles for K2 without re-initializing is $2^{58}$ cycles \((2^{64}\) keystream bits).

### 4 Analysis of K2 Stream Cipher

#### 4.1 Analysis of Periods

The cipher has two LFSRs. LFSR-A is defined by a primitive polynomial. Thus, the sequence of 32-bit outputs generated by LFSR-A has a maximum period of $2^{60} - 1$. The output is XORed with the output of LFSR-B to produce a keystream. Thus, the keystream of the cipher has at least this period. In an experimental analysis using some sequences of the keystream produced by the cipher, we did not find any short periods.

#### 4.2 Analysis of Statistical Tests

The statistical properties of the cipher also depend on the properties of the output sequences of LFSR-A, and LFSR-B; thus, we expect the keystream of the cipher to have good statistical properties. We evaluated the statistical properties for keystream of the cipher and output sequences of LFSR-A and LFSR-B, and confirmed that these properties were good.
4.3 Security Analysis

We briefly discuss security of the cipher against existing attacks.

**Time-Memory Trade-Offs.** We chose the size of the secret key and IV taking into consideration the discussion of general time-memory trade-offs by Hong and Sarker [12]. This attack generally requires $O\left(2^{\frac{3(k+w)}{4}}\right)$ pre-computation, $O(2^{\frac{k+w}{2}})$ memory, and $O(2^{\frac{k+w}{4}})$ available data, enabling an online attack with time complexity of $O\left(2^{\frac{k+w}{4}}\right)$, where the lengths of the secret key and IV are $k$ and $w$, respectively.

The IV is as long as the secret key, and the internal state is sufficiently larger than the secret key. Thus, we think the cipher is not vulnerable to time-memory trade-off attacks.

**Correlation Attacks.** The feasibility of correlation attacks depends on the number of inputs to the non-linear function and on the tap positions for the function. The use of a full positive difference set for the taps and the non-linear function has sufficient non-linearity to make the attacks infeasible. We evaluate the security using an asymptotic analysis described in [5]. An required length $N$ of keystream for an correlation attack is $N \approx 1/4 \cdot (2k \cdot t! \cdot ln2)^{1/t} \cdot e^{-2} \cdot 2^{\frac{l-k}{t}}$, where $l$, $k$, and $t$ denote a target LFSR length, and algorithm parameters, and a correlation probability of the target stream cipher is $1/2 + \epsilon$. The computational complexity of this pre-computation phase in the attack is approximately $N^{[(t-1)/2]}$ and $N^{[(t-1)/2]}$ is required. Furthermore, decoding algorithm stores $(N^t \cdot 2^{k-t})/t!$ parity checks and its computational complexity is $2^k$ times the number of parity checks. In case of attacking the regular clocked LFSR-B in K2, the lowest correlation probability for the attack is approximately $1/2 + 2^{-13}$, where $t = 9$, $k = 26$, and computational complexity and required memories are roughly $O(2^{256})$. However, no correlation between input and output sequences of the non-linear function with the probability larger than $1/2 + 2^{-13}$ is found. Furthermore, the irregular clocking improves the security against correlation attacks, because the linear relations of bits in LFSR-B are more complicated using the irregular clock.

**Chosen/Related IV Attacks.** The initial key is expanded using the AES key scheduling algorithm, and the IV and expanded keys are thoroughly mixed by the 24 cycles comprising the initialization process. After 13 cycles of the run-up, all values of an internal state of the cipher
includes all IVs. Furthermore, initialization process additionally runs 11 cycles and the IVs and an initial key are well mixed into the internal state. Thus, we think that the cipher is not vulnerable to chosen/related IV attacks.

**Guess-and-Determine Attacks.** A simple guess-and-determine attack is where the attacker guesses all values of LFSR-A and all internal memories and determines all values of LFSR-B. However, this attack is impossible because the computational complexity of the attack is at least $O(2^{288})$. Now, we consider a guess-and-determine attack against a simplified K2 that is performed without multiplying $a_i$ ($i = 0, 1, 2, 3$) and additions are replaced by exclusive-or operations. First, we consider an attack designed to remove $A_{t+4}$ from the equation of keystream and to attack focusing on LFSR-B as follows:

$$z^L_t \oplus z^H_{t+4} = (B_t \oplus R2_t) \oplus R1_t \oplus A_{t+4} \oplus (B_{t+14} \oplus L2_{t+4}) \oplus L1_{t+4} \oplus A_{t+4}$$

$$= (B_t \oplus Sub(R1_{t-1})) \oplus R1_t \oplus (B_{t+14} \oplus Sub(L1_{t-1})) \oplus L1_{t+4}$$

If an attacker guesses five elements of the above equation, then the attacker can determine the other element such as $B_{t+14}$ and the attacker also determine $A_{t+4}$. To determine all values of LFSRs, the attacker have to guess at least 10 elements; thus, this attack will be impossible. Next, we consider the other attack where the relations of four internal memories $R1, R2, L1, L2$ are used. The relations of the memories are described as follows:

$$R2_{t+1} = Sub(R1_t), \quad L1_{t+2} = Sub(R2_{t+1} \oplus B_t)$$

$$L2_{t+3} = Sub(L1_{t+2}), \quad R1_{t+4} = Sub(L2_{t+3} \oplus B_t)$$

That is, if an attacker guesses $R1_t, B_t, B_{t+12}$, then the attacker determines $R2_{t+1}, L1_{t+2}, L2_{t+3}, R1_{t+4}$ using the above equations. Now we consider a more simplified algorithm without LFSR-A, that is the attacker obtain values of $z^H_t \oplus A_t$ and $z^L_t \oplus A_{t+4}$ in each cycle $t$. In this situation, if the attacker guesses six elements $R1_{t+1}, R1_{t+2}, L1_t, L1_{t+1}, B_{t+6},$ and $B_{t+7}$, the attacker can determine all values of LFSR-B. The complexity of the second attacks is $O(2^{282})$. However, more than two values of LFSR-A have to be guesses for obtaining all values internal state. Furthermore, the attacker needs to guess clock control bits for the full version algorithm. Thus, we think the full version of the algorithm is
secure against guess-and-determine attacks.

**Distinguishing Attacks.** In distinguishing attacks, a probabilistic linear relation of keystream bits is needed as a distinguisher. We try to construct a linear recurrence from output keystream bits with fixed clock control bits $d1_t = d2_t = 0$ for each cycle. A two-round linear masking of K2 is shown Fig. 4 (see Appendix A). An equation from some cycle output keystreams is as follows:

$$
\begin{align*}
\Gamma \alpha & \cdot \Sigma z^u(t) \oplus \Gamma \alpha \cdot \Sigma z^u(t + 3, t + 5) \oplus \Gamma \alpha \cdot \Sigma z^u(t + 1, t + 6, t + 8, t + 11) \\
\oplus \Gamma & \cdot \Sigma z^u(t + 4, t + 6, t + 9, t + 13, t + 14, t + 16) \oplus \Gamma \alpha \cdot \Sigma z^u(t + 1, t + 6, t + 8, t + 11) \\
\oplus \alpha & \cdot \Sigma z^u(t + 4, t + 6, t + 9, t + 13, t + 14, t + 16) \oplus \Gamma \alpha \cdot \Sigma z^u(t + 1) \\
\oplus \Phi & \cdot \Sigma z^u(t + 4, t + 6) \oplus \Psi \alpha \cdot \Sigma z^u(t + 2, t + 7, t + 9, t + 12) \\
\oplus \Phi \cdot & \cdot \Sigma z^u(t + 4, t + 6) \oplus \Psi \alpha \cdot \Sigma z^u(t + 2, t + 7, t + 9, t + 12) \\
\oplus \Psi & \cdot \Sigma z^u(t + 5, t + 7, t + 10, t + 14, t + 15, t + 17) = 0
\end{align*}
$$

where $\Gamma$, $\lambda$, $\Phi$, and $\Psi$ are linear masks, and $\Sigma z^u(y_1, y_2, ..., y_n)$ denotes $\Sigma z^u \oplus z^u \oplus ... \oplus z^u$. If a bias for a combination of linear masks has high probability, an attacker constructs a distinguisher from the above equation. However, we have not found a combination of linear masks with a bias value higher than $2^{-128}$. Furthermore, to obtain the above equation, all clock control bits for 15 times feedback operations of LFSR-B are $d1_t = d2_t = 0$; the probability for this condition of clock control bits is about $2^{-30}$. That is, computational complexity of a distinguishing attack against the cipher increase $2^{30}$ times by using the dynamic feedback control mechanism. Additionally, the cipher is assumed to be re-initialized after $2^{58}$ cycles. Thus, distinguishing attacks against K2 is impossible.

**Algebraic Attacks.** The non-linear function has ideal algebraic properties; the non-linear function consists of AES S-boxes and an effective permutation function. Furthermore, the dynamic feedback control increases the cost of solving a system of internal values. Coutoos presented an evaluation method for the complexity of general algebraic attacks [7]. A general evaluation suggests that K2 is secure against algebraic attacks; computational complexity of the attack is roughly $O(2^{64})$.

We investigated the possibility of algebraic attacks, when we assumed that LFSR-B has regular clocking and the addition modulo $2^{32}$ operation
is replaced by the XOR operation. We tried to apply an algebraic attack against SNOW 2.0 as proposed by O. Billet and H. Gilbert. We obtained the following equation:

\[ R_{t+1} = (B_t \oplus \text{Sub}(R_{t-1})) \oplus A_{t+4} \oplus z_t^I \]

\[ L_{2t-1} = (B_{t+9} \oplus \text{Sub}^{-1}(L_{2t})) \oplus A_{t-1} \oplus z_{t-1}^H \]

The attacker cannot obtain linear equation of fixed values of internal memories and registers, \( R_{t0}, L_{20}, B_0, B_1, \ldots, B_{10}, A_0, \ldots, A_4 \) for \( R_t \) and \( L_{2t-1} \). This attack is impossible even for the regular clocking algorithm. Furthermore, the attacker has to guess the clocks of each cycle to determine the equations for full version of the cipher. Let \( M \) be the total number of non-constant monomials appearing in the overdefined system of equations, and \( N \) be the number of equations that the attacker obtains per output of one cycle. The computational complexity of the algebraic attack increases \( 2^{2 \cdot (\lceil M/N \rceil - 1)} \leq 2^{360} \) times by using the dynamic feedback control mechanism. Thus, we think the full version is secure against the algebraic attack.

**Clock Control Guessing Attack.** This attack [17] is effective against bit-oriented clock controlled stream ciphers. The cipher is a word-oriented stream cipher having a large internal state, and its non-linear part is more complicated than the existing stream ciphers broken by the attacks. An extended attack based on algebraic approach have been discussed by S. Al-Hinai \( et. \ al. \) [1]. However, it is difficult to apply the attack when a sufficiently secure non-linear function is used to generate the keystream. Thus, we expect that the cipher will be secure against such attacks.

**Divide-and-Conquer Approach.** The output sequences of LFSR-A and LFSR-B have good statistical properties. Thus, we expect that divide-and-conquer attacks for the LFSRs are infeasible.

### 4.4 Performance Analysis

We implemented the algorithm on a PC (Pentium 4 3.2 GHz) using Intel C++ Compiler Ver.9 (for Windows), and evaluated its performance. The evaluation results are shown in Table 1. “Key. Gen.” indicates the required cycles for a 1 byte keystream generation and “Init.” indicates the required cycles for one initialization including an initial key and IV setup. The optimal version is optimized in assembly language and produces 128-byte keystream for each 16 cycles. The performance of K2 is much
faster than existing clock controlled stream ciphers and AES in software implementation, and is competitive against word-oriented stream ciphers such as SNOW 2.0. The inner state efficiency \cite{[18]} of the cipher, 0.4, is enough efficient.

5 Conclusion

This paper proposed a new design for stream cipher that is a word-oriented stream cipher using dynamic feedback control. The stream cipher K2 is a secure against several attacks and it offers high-performance encryption /decryption for software implementations.

Acknowledgments. The authors thank K. Fukushima and J. Kurihara for their help and the anonymous reviewers of SASC 2007 for helpful comments.

References

A Two-Round Linear Distinguisher for K2

A two-round linear distinguisher for SNOW is shown in Fig. 3.

![Diagram](image)

**Fig. 3.** Linear Masking of K2 for Two-Round Outputs

B Test Vector

Initial Key:

\[ IK_0 = 0x00000000, IK_1 = 0x00000000, IK_2 = 0x00000000, IK_3 = 0x00000000 \]

Initial Vector:

\[ IV_0 : 0x00000000, IV_1 = 0x00000000, IV_2 = 0x00000000, IV_3 = 0x00000000 \]
\[ IV_4 : 0x00000000, IV_5 = 0x00000000, IV_6 = 0x00000000, IV_7 = 0x00000000 \]

Keystream:

0xA8B6E59E9518C566, 0x9EA8F1874085F28F, 0xFF3BB37BDD15DB2A
0xEFC69F9473A12FC, 0xD95A151E66AE124E, 0xD0241C1CDD0CFA1C
0x36D2D56AA69C141D, 0x26429C1959B6CE8A