Final Report

Evaluation of the
Word-Oriented Stream Cipher: K2

by
Bimal Roy
General Secretary
Cryptology Research Society of India
India
1 Introduction

In this report we investigate upon the security aspects of the word oriented stream cipher, K2. The stream cipher has been developed by the Information Security Laboratory of the KDDI R&D Laboratories Inc., Japan. More details on the specification of the stream cipher is given in Section 2 of this report.

Scope of the Project: The scope of the project and hence that of this report is limited to the investigation of the security of the algorithm K2 against attacks. Specifically, we investigate the following aspects:

- Design of a non-linear function of the algorithm,
- Design of linear feedback shift registers of the algorithm.

The report will further describe if any other vulnerabilities can be identified in the algorithm.

Deliverables: The deliverable of the project is a report addressing any vulnerabilities that have been identified.

2 Specification

We analyzed the stream cipher K2 as described in the document


We assume the above document [8] as the final specification of the algorithm.

2.1 Implementation

We have two implementations in ‘C’ programming language of the algorithm – one as supplied by the KDDI Lab and the other initially developed by us and further modified by the KDDI Lab. These two implementation agree in their output and we assume both to be correct instantiation of the algorithm.
3 Structural Analysis of K2

3.1 Structure

K2 is a word oriented stream cipher; in each round it outputs two words each of size 32 bits. The cipher uses irregular clocking as dynamic feedback control. Designer(s) of K2 envisages the dynamic feedback control as a potentially effective tool against different kinds of attacks – existing as well as novel.

The stream cipher K2 consists of two parts:

- Feedback shift register part,
- Non-linear part.

The feedback shift register part consists of two feedback shift registers, \( LFSR-A \) and \( LFSR-B \). \( LFSR-A \) consists of five registers and \( LFSR-B \) consists of eleven registers, where each register is of 32 bits. The \( LFSR-A \) is defined by a degree five primitive polynomial over \( GF(2^{32}) \), where the finite field \( GF(2^{32}) \) is built using a towering field implementation – a degree four extension field of the base field \( GF(2^8) \) defined by a certain irreducible polynomial of degree eight over \( GF(2) \). The connection polynomial for \( LFSR-B \) has four possibilities chosen by the dynamic feedback controller. Amongst these four possibilities, two are primitive polynomials of degree eleven over \( GF(2^{32}) \). The other two possibilities do not have definite algebraic structure per se.

The non-linear part has four memory registers \( R_1, R_2, L_1, L_2 \) each of 32 bits and the non-linear function consists of four substitution steps denoted as \( Sub \). The \( Sub \) function first divides the 32 bits input into 4 bytes and applies the 8-to 8 bit substitution of the AES [4] S-box. This is followed by a 32-to-32 bit linear permutation which is same as the AES \( Mix \ Column \) operation.

For more details on the structure of the K2 we refer the readers to the original specification [8].

3.2 Key Generation

3.2.1 Initialization

The key initialization consists of two steps - key loading and internal state initialization. The process can be described both for an 128-bit initial key
\((IK0, IK1, IK2, IK3)\) and a 256-bit initial key \((IK_0, IK_1, \ldots, IK_7)\) together with a 256-bit initial vector \(IV = (IV_0, IV_1, \ldots, IV_7)\). First the initial key (128-bits or 256-bits) is expanded to 512-bits using a key scheduling algorithm. This key scheduling is similar to the round key generation function of AES [4]. Once the key loading is done, the internal states are initialized as follows:

\[
A_i = K_i \text{ for } i = 0, \ldots, 4; \quad B_i = K_{5+i} \text{ for } i = 0, \ldots, 10; \quad R_{0} = IV_0;
\]

\[
L_0 = IV_1; \quad R_{2} = IV_2; \quad L_2 = IV_3;
\]

\[
B_5 = B_5 \oplus IV_4; \quad B_6 = B_6 \oplus IV_5; \quad B_7 = B_7 \oplus IV_6;
\]

\[
B_8 = B_8 \oplus IV_7;
\]

Once the internal states are initialized the cipher is clocked 24 times without taking any output. During this process, the states \(A_{j+4}\) and \(B_{j+10}\) of LFSR-A and LFSR-B respectively, are also updated according to certain rules. Once the cipher is initialized, the recommended maximum number of keystream words is \(2^{58}\) which amounts to \(2^{64}\) bits.

### 3.3 Keystream Generation

The key stream is generated as per the following rule:

\[
\begin{align*}
Z_t^L & = B_t + R_{2t} \oplus R_{1t} \oplus A_{t+4} \\
Z_t^H & = B_{t+10} + L_{2t} \oplus L_{1t} \oplus A_t
\end{align*}
\]

where the output at time \(t\) \(Z_t = (Z_t^H, Z_t^L)\) is divided into two sub-strings of 32 bits. The operations +, \(\oplus\) denote bit wise exclusive-or and addition modulo 32 respectively.

The memory registers are also updated as follows:

\[
\begin{align*}
R_{1t+1} & = Sub(L_{2t} + B_{t+9}) \\
R_{2t+1} & = Sub(R_{1t}) \\
L_{1t+1} & = Sub(R_{2t} + B_{t+4}) \\
L_{t+1} & = Sub(L_{1t})
\end{align*}
\]
4 Theoretical Analysis

In our analysis of the algorithm K2, we assume that the only secret part is the initial key ($IK$). Everything else, including the initial vector ($IV$), are publicly known.

4.1 On Primitivity

The algorithm K2 uses three primitive polynomials – one for the connection polynomial of $LFSR - A$ and the other two for the connection polynomial of $LFSR - B$ depending upon the output of the dynamic feedback control. The former is a degree five polynomial over $GF(2^{32})$ and the later two are degree eleven polynomials over $GF(2^{32})$. We have tested this polynomials for primitivity and found that they are indeed primitive polynomials over $GF(2^{32})$.

4.2 Low Degree $t$-nomial Multiple

A polynomial with $t$ non zero terms, one of them being the constant term is called $t$-nomial, or in other words a polynomial of weight $t$. As example, $x^s + x^t + 1$ is 3-nomial (trinomial), and $x^s + x^t + x^r + 1$ is a 4-nomial or a polynomial of weight 4. It is important to note that towards resisting cryptanalytic attacks, the LFSRs should be designed keeping the following points in mind [10, 7].

1. The connection polynomial should be primitive.
2. The weight of the connection polynomial should be high.
3. There should not be any sparse multiple of moderate degree for the connection polynomial.

For the case of connection polynomials of LFSR-A and LFSR-B of K2, low degree $t$-nomial multiples do not seem to be a threat.

4.3 Period of Output Sequence

The period of the key stream generated by the algorithm K2 will depend upon the period of the output sequence of $LFSR - A$, $LFSR - B$ as well as on the
Non-Linear Function. Since the connection polynomial of $LFSR - A$ is a degree five primitive polynomial over $GF(2^{32})$; the output sequence generated by $LFSR - A$ is an $m$-sequence with period $2^{160} - 1$. $LFSR - B$ has period $\geq 2^{318}$. It seems the period of the key stream is very high. The recommended maximum length of the key stream for $K2$ before reinitializing is $2^{64}$. This is well within the safe limit.

4.4 Exhaustive Key Search

The length of the initial key of $K2$ is either 128-bits or 256-bits. So the exhaustive key search will have a time complexity of order $2^{128}$ or $2^{256}$ as the case may be. This is not really a threat. Any method that reduces the time complexity from that of the exhaustive key search can be regarded as a potential attack on $K2$.

4.5 SNOW 2.0 Like Structure

The non-linear part of $K2$, together with $LFSR - B$ has strong resemblance with the structure of SNOW 2.0. In fact non-linear part of $K2$ may be interpreted as two FSMs of SNOW 2.0 cascaded in between. SNOW 2.0 is a well studied word oriented stream cipher. It can be expected that the use of SNOW 2.0 like FSMs in the non-linear part in case of $K2$ will lead to property inheritance.

4.6 Use of AES S Box and Mix Column Operation

The substitution steps in the non-linear part of $K2$ use 8-to-8 bits non-linear permutations which are same as the S-boxes of AES; followed by 32-to-32 bits linear permutation which is again same as the Mix Column operation of AES. The non-linearity of these operations of AES is known and the individual Sub steps of $K2$ retain this non-linearity.

4.7 Walsh Transform

The Walsh transform (WT) of an $m$-variable Boolean function $g$ is an integer valued function $W_g : \{0, 1\}^m \rightarrow [-2^m, 2^m]$ defined by (see [9, page 414])

$$W_g(u) = \sum_{w \in \mathbb{F}_2^m} (-1)^{g(w) + \langle u, w \rangle}.$$

(1)
The WT is called the *spectrum* of $g$.

Given a truth table we can compute WT by using the fast WT algorithm [9] which has complexity $O(m2^m)$, where $m$ is the number of input variables.

### 4.8 Non-linearity

A parameter of fundamental importance in cryptography is the *nonlinearity* of a Boolean function. This quantity measures the distance of a Boolean function from the set of all affine functions. Let $A_m$ be the set of all $m$-variable affine functions. The nonlinearity $nl(f)$ of an $m$-variable Boolean function is defined as $nl(f) = \min_{l \in A_m} d(f, l)$, where $d()$ stands for Hamming distance.

It is sometime convenient to express nonlinearity in terms of the spectrum of a Boolean function. The nonlinearity $nl(f)$ of an $m$-variable Boolean function $f$, can be written as

$$nl(f) = 2^{m-1} - \frac{1}{2} \max_{u \in \mathbb{F}_2^m} |W_f(u)|.$$

Let $f$ be an $(n, m)$ S-box. The nonlinearity of $f$ is defined to be

$$nl(f) = \min\{nl(l \circ f) : l \text{ is a non-constant } m\text{-variable linear function}\}.$$

For an $n$ variable Boolean function complexity of computing nonlinearity is $O(n2^n)$, and for an $(n, m)$ S-box complexity of computing nonlinearity is $O(n2^n \times 2^m)$.

Linear cryptanalysis is a very powerful cryptanalytic method for symmetric ciphers. The study of correlation between linear combinations of input and output of an S-box is therefore very important. A function with low nonlinearity is prone to linear approximation attack. Linear approximation means approximating the combining function by a linear function. Thus for symmetric cipher applications we need functions (both Boolean functions and S-boxes) with high nonlinearity.

We can think of the Non-Linear Function of K2 as an S-box. To do so we are considering the Non-Linear Function with four 32 bit inputs (we are ignoring two 32 bit inputs from LFSR-A, as the operation on it is simple Xor) and two 32 bit outputs. So we have $(128, 64)$ S-box. Complexity of computing nonlinearity is $O(128 \times 2^{128} \times 2^{64})$ which is infeasible. We explore
the partial nonlinearity of the Non-Linear Function (here an S-box denoted by $f$) with some experiments. To do so we apply two methods, Method 1 and Method 2.

**Method 1.** We define Walsh distance (WD) of two binary strings of equal length (say $l$) to be the number of places they are equal minus the number of places they are unequal. Note, WD does not has metric property. For two random strings WD follows binomial distribution. Our goal is to check randomness property for the Non-Linear Function in this direction. Let $(x, y)$ be a random pair, we compute $f(x)$ and $f(y)$. Now we compute WD between $f(x \oplus y)$ and $f(x) \oplus f(y)$. In ideal case WD should be zero.

We randomly select $2^k$ pair (here $k$ is around 30) of 128 bit strings say $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{2^k}, y_{2^k})\}$. We compute $\{(f(x_1), f(y_1)), (f(x_2), f(y_2)), \ldots, (f(x_{2^k}), f(y_{2^k}))\}$. Now we compute frequency distribution of WD between $f(x_i \oplus y_i)$ and $f(x_i) \oplus f(y_i)$ for $1 \leq i \leq 2^k$. We repeat this experiment for different values of $k$ and for different random pairs once $k$ is fixed. Our experimental values closely match with theoretical values. We provide one such experimental result as an example in Table 1.

**Method 2.** We randomly select $2^k$ number (here $k$ is of the order of 10) of 128 bit strings say $\{x_1, x_2, \ldots, x_{2^k}\}$. We compute $\{f(x_1), f(x_2), \ldots, f(x_{2^k})\}$ which are 64 bit strings as the S-box here is (128, 64). We take random linear combination of inputs (say $I_s$) and outputs (say $O_s$). Now we compute frequency distribution of WD between $I_s$ and $O_s$ for $1 \leq i \leq 2^l$ (here $l$ is around 20). We repeat this experiment for different values of $k$ and $l$ and for different random inputs once $k$ and $l$ are fixed. We provide one sample experimental result in Table 2.

### 4.9 Barlekamp-Massey Attack

The Barlekamp-Massey algorithm states that if an LFSR has linear complexity $L$, then from an output sequence of length $2L$ generated by the LFSR, it is possible to uniquely generate the connection polynomial of that LFSR. This is the reason why LFSRs by themselves are never directly used to generate the key streams. Another interpretation of the Barlekamp-Massey algorithm is, for (pseudo)random bit sequence of length $n$, the linear complexity should be of the order $n/2$. 
The algorithm K2 would be vulnerable to Barlekamp-Massey attack if the non-linear transformation was absent. The use of dynamic feedback control is effective to increase the linear complexity of the system. But that by itself is not sufficient enough to resist the Barlekamp-Massey attack.

We have run the Barlekamp-Massey algorithm on the key stream generated by K2 and on its random subsequences. The results of the experiments indicate that the linear complexity of the key bit sequences is like that of a (pseudo) random sequence.

<table>
<thead>
<tr>
<th>WD</th>
<th>Experimental Value</th>
<th>Theoretical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20</td>
<td>1226610</td>
<td>1169209</td>
</tr>
<tr>
<td>-18</td>
<td>2132264</td>
<td>2135078</td>
</tr>
<tr>
<td>-16</td>
<td>3519338</td>
<td>3647425</td>
</tr>
<tr>
<td>-14</td>
<td>5579856</td>
<td>5835879</td>
</tr>
<tr>
<td>-12</td>
<td>8774900</td>
<td>8753819</td>
</tr>
<tr>
<td>-10</td>
<td>12178384</td>
<td>12320190</td>
</tr>
<tr>
<td>-8</td>
<td>16108004</td>
<td>16280250</td>
</tr>
<tr>
<td>-6</td>
<td>20237172</td>
<td>20209966</td>
</tr>
<tr>
<td>-4</td>
<td>23339004</td>
<td>23578294</td>
</tr>
<tr>
<td>-2</td>
<td>26505590</td>
<td>25860064</td>
</tr>
<tr>
<td>0</td>
<td>26692612</td>
<td>26668191</td>
</tr>
<tr>
<td>2</td>
<td>25905169</td>
<td>25860064</td>
</tr>
<tr>
<td>4</td>
<td>23753302</td>
<td>23578294</td>
</tr>
<tr>
<td>6</td>
<td>20514948</td>
<td>20209966</td>
</tr>
<tr>
<td>8</td>
<td>16420362</td>
<td>16280250</td>
</tr>
<tr>
<td>10</td>
<td>12050350</td>
<td>12320190</td>
</tr>
<tr>
<td>12</td>
<td>8703893</td>
<td>8753819</td>
</tr>
<tr>
<td>14</td>
<td>5675940</td>
<td>5835879</td>
</tr>
<tr>
<td>16</td>
<td>3621760</td>
<td>3647425</td>
</tr>
<tr>
<td>18</td>
<td>2120094</td>
<td>2135078</td>
</tr>
<tr>
<td>20</td>
<td>1174076</td>
<td>1169209</td>
</tr>
</tbody>
</table>

Table 1: Partial Nonlinearity Values for $k = 28$ from Method 1.
<table>
<thead>
<tr>
<th>WD</th>
<th>Experimental Value</th>
<th>Theoretical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-20</td>
<td>302836</td>
<td>248984</td>
</tr>
<tr>
<td>-18</td>
<td>371272</td>
<td>334997</td>
</tr>
<tr>
<td>-16</td>
<td>457632</td>
<td>436693</td>
</tr>
<tr>
<td>-14</td>
<td>556358</td>
<td>551612</td>
</tr>
<tr>
<td>-12</td>
<td>655300</td>
<td>675249</td>
</tr>
<tr>
<td>-10</td>
<td>760880</td>
<td>801143</td>
</tr>
<tr>
<td>-8</td>
<td>860254</td>
<td>921314</td>
</tr>
<tr>
<td>-6</td>
<td>1002430</td>
<td>1027039</td>
</tr>
<tr>
<td>-4</td>
<td>1046078</td>
<td>1109865</td>
</tr>
<tr>
<td>-2</td>
<td>1074844</td>
<td>1162715</td>
</tr>
<tr>
<td>0</td>
<td>1077852</td>
<td>1180883</td>
</tr>
<tr>
<td>2</td>
<td>1057839</td>
<td>1162715</td>
</tr>
<tr>
<td>4</td>
<td>1018280</td>
<td>1109865</td>
</tr>
<tr>
<td>6</td>
<td>951989</td>
<td>1027039</td>
</tr>
<tr>
<td>8</td>
<td>858487</td>
<td>921314</td>
</tr>
<tr>
<td>10</td>
<td>758644</td>
<td>801143</td>
</tr>
<tr>
<td>12</td>
<td>619208</td>
<td>675249</td>
</tr>
<tr>
<td>14</td>
<td>523356</td>
<td>551612</td>
</tr>
<tr>
<td>16</td>
<td>445402</td>
<td>436693</td>
</tr>
<tr>
<td>18</td>
<td>367775</td>
<td>334997</td>
</tr>
<tr>
<td>20</td>
<td>311857</td>
<td>248984</td>
</tr>
</tbody>
</table>

Table 2: Partial Nonlinearity Values for $k = 7$ and $l = 24$ from Method 2.

### 4.10 Algebraic Attack

The idea behind the algebraic attacks is to express the cipher as a system of multivariate equations whose solution gives the secret key. The complexity of the attack depends on the number of such equations, their type and their algebraic degree. The first algebraic attack on a block cipher was discussed in [11]. In [2] Courtois and Pieprzyk showed that AES [4] may be vulnerable by solving an overdefined system of algebraic equations. This is possible because the only nonlinear component in AES, i.e., the S-box, can be expressed as an overdefined system of algebraic equations. The authors presented an
algorithm called XSL to solve this system of multivariate equations and also introduced the notion of algebraic immunity, $\Gamma$, of S-boxes where $\Gamma$ is an important parameter in measuring the complexity of the XSL algorithm. For a $GF(2^n) \rightarrow GF(2^n)$ S-box $\Gamma$ is defined as $\Gamma = ((t - r)/n)\lceil(t-r)/n\rceil$, where $r$ is the number of equations and $t$ is the number of monomials in these equations. They showed that for AES S-box $\Gamma = 2^{22.9}$ and claimed that for secure ciphers $\Gamma$ should be greater than $2^{32}$.

In [3, Appendix A] Courtois et al. also used the polynomial representation of the algebraic S-boxes to prove that S-boxes based on inverse mapping in $GF(2^n)$ have $3n - 1$ bi-affine equations and $5n - 1$ quadratic equations.

We have not found any weakness in terms of algebraic attacks.

### 4.11 Time-Memory Trade-off Attack

Time-Memory trade-off attack is feasible if the size of the initial vector ($IV$) is less than the size of the initial key ($IK$), if one ignores the pre-computation [6]. Since the size of the initial key of K2 is either $IK = 128$ or $IK = 256$ and $IV = 256$ and the size of the internal state is 640 bits. So this kind of attack is not possible on K2.

### 5 Statistical Analysis

#### 5.1 NIST Tests

National Institute of Standards and Technology (NIST), Govt. of US has developed a statistical package called the NIST Test Suite [1]. The package consists of 16 tests designed to check the randomness of bit sequences of arbitrary lengths. These tests are:

- The Frequency Test
- Frequency Test within a Block
- The Runs Test
- Test for the Longest-Run-of-Ones in a Block
- The Binary Matrix Rank test
- The Discrete Fourier Transform (Spectral) Test
• The Non-overlapping Template Matching Test
• The Overlapping Template Matching Test
• Maurer’s “Universal Statistical” Test
• The Lempel-Ziv Compression Test
• The Linear Complexity Test
• The Serial Test
• The Approximate Entropy Test
• The Cumulative Sums Test
• The Random Excursions Test
• The Random Excursions Variant Tests

We have run all these tests on the output of the key streams generated by K2. The length of the bit sequences was varied upto $2^{32}$. The key streams passed through all the tests.

5.1.1 Tests on Random Subsequences

Any random subsequence of a random sequence is random. We have generated many random subsequences by different methods and tested it. No bias was found by any of the 16 tests from the NIST suite.

6 Conclusion

In this report, we have summarised the results of our investigation on the security aspects of the stream cipher K2. We found no security flaws in the cipher. The results are encouraging as far as the security of the cipher is concerned.
References


