THE SECURITY ANALYSIS ON THE RABBIT STREAM CIPHER

*Kai Chain**

Department of Intelligent Network Technology, I-SHOU University
Taiwan, R.O.C.

* Corresponding Author, e-mail: kaichain@isu.edu.tw

Abstract: In recent years, stream cipher systems that have been traditionally designed using linear feedback shift register have been almost entirely compromised by algebraic attack methods. Thus, identifying a method to establish concepts for new-generation ciphers, prevent existing security problems, and design new stream cipher systems that consider both security and performance has become a crucial concern in the field of cryptography. In 2004, the European Union initiated the eSTREAM project to emulate the Advanced Encryption Standard used in the United States. The project consisted of 48 participating stream cipher candidates. Through open selection, review, and runoff voting, the results were announced in May 2008. This research investigated one of the finalists of the eSTREAM competition: The Rabbit stream cipher. Additionally, stream cipher attack methods have been extensively studied in recent years, especially those for distinguishing attacks. Thus, contributions of this article is explored the design concepts of the core algorithms in the new-generation stream cipher systems for determining the corresponding mathematical principles and practical approaches to contribute to the study of stream cipher systems.

Key words: Security analysis, eSTREAM, Rabbit stream cipher, Pseudo-linear function modulo, Distinguishing attack.

1. INTRODUCTION

The design of stream ciphers, including the Rabbit stream cipher, is focused on producing security key streams [1]. Since Rabbit was proposed in 2003, its algorithm during the entire selection process of the eSTREAM project has remained almost unchanged. Ultimately, because it garnered a high score, the Rabbit was selected as one of the new-generation stream ciphers, which reflects the success of its entire framework design. The major attribute that distinguishes the Rabbit algorithm from other stream encryption algorithms is their corresponding input sources. Most stream ciphers consist of two input sources (i.e., the key and initialisation vector), whereas in the Rabbit algorithm, the initialisation vector is considered an additional optional support. Additionally, the security and performance analysis in Boesgaard et al. [1, 16] all used

the key as a single input. Over the past few years, several papers have been exploring the implementation of the Rabbit stream cipher using hardware [18, 19].

A 128-bit key is initially input in the Rabbit algorithm as an initial value. After completing each iteration of the Rabbit algorithm, a 128-bit-long key stream is generated as the output. During encryption (decryption), the iteration of the Rabbit algorithm is determined according to the key stream lengths required for plaintext (ciphertext). The output key stream and plaintext (ciphertext) are then computed using the exclusive-or (XOR) logic to complete the encryption (decryption) process.

This paper is organized as follows. Section 2 reviews the Rabbit stream cipher. Section 3 reviews the fast probability distribution calculations. Section 4 analyzes the security of the Rabbit stream cipher. Finally, Section 5 concludes the paper.

Stream ciphers are intriguing, formed by a combination of numerous mathematical and logical operations. The main contribution of this article lies in conducting a detailed analysis of the security of the Rabbit stream cipher, and providing examples for reference.

2. THE RABBIT STREAM CIPHER

2.1. Structure and specification

The Rabbit stream cipher uses 128-bit keys and consists of 513-bit internal states. 512 bits are divided between eight 32-bit state variables $x_{j,t}$ and eight 32-bit counter variables $c_{j,t}$, where $x_{j,t}$ is the state variable of subsystem j at iteration t, and $c_{j,t}$ denotes the corresponding counter variable. There is one counter carry bit is initialized to zero. The eight state variables and eight counters are derived from the key at initialization. Assuming that the internal state at time t ($t \geq 1$) is expressed using a combination of 16 32-bit words (i.e., $x_{0,t},\ldots,x_{7,t}$, $c_{0,t},\ldots,c_{7,t}$) and a 1-bit $\phi_{7,t}$, a sum of 16 × 32 + 1 = 513 bits is obtained. The term $c_{j,t}$ can be obtained through Equation (1), which expresses that $c_{j,t}$ in the t^{th} iteration is obtained using $c_{j,t-1}$ from the $(t\text{-}1)^{\text{th}}$ iteration.

$$c_{j,t} = \begin{cases} c_{0,t-1} + a_0 + \phi_{7,t-1} \bmod 2^{32}, if\ j = 0 \\ c_{j,t-1} + a_j + \phi_{j-1,t} \bmod 2^{32}, if\ j > 0 \end{cases} \quad (1)$$

In the relation,

$$\phi_{j,t+1} \begin{cases} 1, if\ j = 0\ and\ c_{0,t} + a_0 + \phi_{7,t} \geq 2^{32} \\ 1, if\ j > 0\ and\ c_{j,t} + a_j + \phi_{j-1,t+1} \geq 2^{32} \\ 0, otherwise \end{cases} \quad (2)$$

a_i is a constant; expressed using hexadecimals: $a_0 = a_3 = a_6$ = 0x4D34D34D, $a_1 = a_4 = a_7$ = 0xD34D34D3, $a_2 = a_5$ = 0x34D34D34.

$x_{j,t+1}$ is calculated as follows:

$x_{j,t}$ at iteration t is obtained using $x_{j,t-1}$ at iteration $(t\text{-}1)$ and $c_{j,t}$ at iteration t; thus, $x_{j,t}$ can only be obtained after calculating $c_{j,t}$ at iteration t.

$$x_{0,t+1} = g_{0,t} + (g_{7,t} <<< 16) + (g_{6,t} <<< 16), \quad (3)$$

$$x_{1,t+1} = g_{1,t} + (g_{0,t} <<< 8) + g_{7,t}, \quad (4)$$

$$x_{2,t+1} = g_{2,t} + (g_{1,t} <<<16) + (g_{0,t} <<<16), \quad (5)$$

$$x_{3,t+1} = g_{3,t} + (g_{2,t} <<<8) + g_{1,t}, \quad (6)$$

$$x_{4,t+1} = g_{4,t} + (g_{3,t} <<<16) + (g_{2,t} <<<16), \quad (7)$$

$$x_{5,t+1} = g_{5,t} + (g_{4,t} <<<8) + g_{3,t}, \quad (8)$$

$$x_{6,t+1} = g_{6,t} + (g_{5,t} <<<16) + (g_{4,t} <<<16), \quad (9)$$

$$x_{7,t+1} = g_{7,t} + (g_{6,t} <<<8) + g_{5,t}, \quad (10)$$

The symbol <<< in Equations (3) – (10) represents rotating three bits to the left (<<< 16 for j even, <<< 8 for j odd). □ denotes logical XOR, ‖ denotes concatenation of two sequences. All additions (expressed using the symbol +) are modulo 2^{23}. In addition, $g_{j,t}$ is defined as follows:

$$g_{j,t} = (x_{j,t} + c_{j,t+1})^2 \oplus ((x_{j,t} + c_{j,t+1})^2 >> 32), \; 0 \le j \le 7. \quad (11)$$

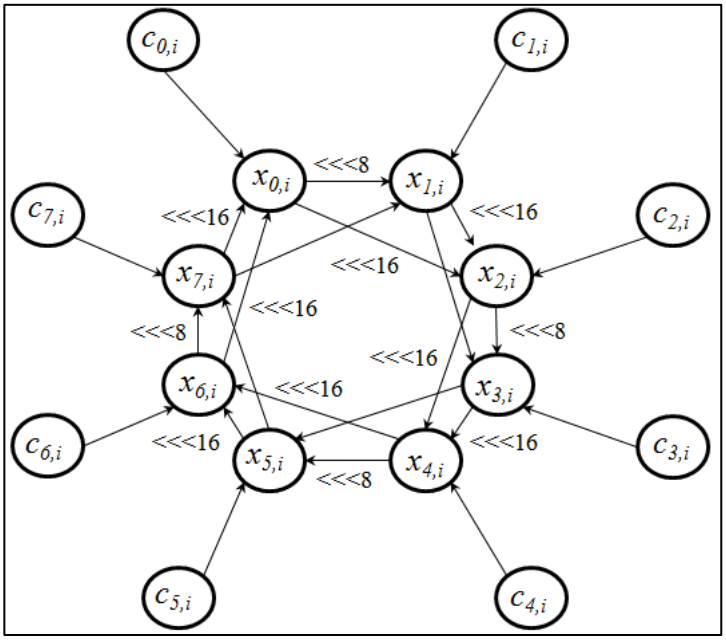

Figure 1. A conceptual graph of the internal state update in the Rabbit algorithm
Source: Graphical illustration of the rabbit algorithm [17]

Figure 1 illustrates the concept of the aforementioned computations. The internal state update process is the core of the Rabbit algorithm. Key streams are obtained through these internal states. The Rabbit algorithms can be simplified into two steps, from internal state initialisation to key stream generation [2, 3], and the corresponding processes are described in the following two sections.

2.2. Internal state initialization

During initialisation, the 128-bit key K is set to $k_7k_6...k_0$; each k_i is a 16-bit value; then, the internal state at t = - 4 is expressed in the following relations:

$$x_{j,-4} = \begin{cases} k_{(j+1 \bmod 8)} \| k_j, \text{ for even } j \\ k_{(j+5 \bmod 8)} \| k_{(j+4 \bmod 8)}, \text{for odd } j \end{cases}, \quad (12)$$

$$c_{j,-4} = \begin{cases} k_{(j+4 \bmod 8)} \| k_{(j+5 \bmod 8)}, \text{ for even } j \\ k_j \| k_{(j+1 \bmod 8)}, \text{for odd } j \end{cases}, \quad (13)$$

$$\phi_{7,-4} = 0. \quad (14)$$

These internal states yield the final initial value after four updates, when each $c_{j,0}$ is replaced using Equation (15).

$$c_{j,0} = c_{j,0} \oplus x_{(j+4 \bmod 8),0}, \quad (15)$$

after completing internal state initialisation, the next iteration ($t = 1$) begins to output the first key stream.

2.3. Key stream generation

When $t \geq 1$, 128-bit key streams $s_t$are computed using the following relations:

$$s_t^{[15...0]} = x_{0,t}^{[15...0]} \oplus x_{5,t}^{[31...16]}, \quad (16)$$

$$s_t^{[31...16]} = x_{0,t}^{[31...16]} \oplus x_{3,t}^{[15...0]}, \quad (17)$$

$$s_t^{[47...32]} = x_{2,t}^{[15...0]} \oplus x_{7,t}^{[31...16]}, \quad (18)$$

$$s_t^{[63...48]} = x_{2,t}^{[31...16]} \oplus x_{5,t}^{[15...0]}, \quad (19)$$

$$s_t^{[79...64]} = x_{4,t}^{[15...0]} \oplus x_{1,t}^{[31...16]}, \quad (20)$$

$$s_t^{[95...80]} = x_{4,t}^{[31...16]} \oplus x_{7,t}^{[15...0]}, \quad (21)$$

$$s_t^{[111...96]} = x_{6,t}^{[15...0]} \oplus x_{3,t}^{[31...16]}, \quad (22)$$

$$s_t^{[127...112]} = x_{6,t}^{[31...16]} \oplus x_{1,t}^{[15...0]}, \quad (23)$$

$s_t^{[a...b]}$ denotes the bit block of s_t at a^{th} to the b^{th} bit; this sums to (a–b + 1) bits with the assumption that $a \geq b$.

2.4. Encryption and decryption

Encryption and decryption are computed using XOR logic in which $c_t = p_t \oplus s_t$ and $p_t = c_t \oplus s_t$; and c_t and p_t, respectively, represent the t^{th} ciphertext and plaintext block [4].

3. FAST PROBABILITY DISTRIBUTION CALCULATIONS

Maximov and Johansson [5] published their paper in Asiacrypt 2005, proposing an algorithm for fast pseudo-linear function modulo 2^n (PLFM) distributions, which aimed at increasing the efficiency of cryptanalysis, especially for linear attacks (linear cryptanalysis). According to a previous study, a linear attack is one of the most threatening attacks toward stream ciphers. Because the content in Maximov and Johansson is difficult to comprehend, examples are organised in this paper to facilitate

the understanding of the concepts and meanings of the algorithms. Please refer to the original paper for detailed descriptions [5].

3.1. The pseudo-linear function modulo 2^n and its distribution

Prior to discussing Maximov and Johansson's algorithms, examples were designed to facilitate interpreting the PLFM and the corresponding distribution and meaning of the function.

Example 1: Assuming that two 2-bit random variables are X_1 and X_2 and that their sum through modular addition is γ, then a PLFM can be defined as $F(X_1, X_2) = X_1+X_2$, subsequently, the distribution of this PLFM can be obtained by calculating the probabilities of all $P_r\{F(X_1, X_2) = X_1+X_2 = \gamma$. The computation of this modular addition is expressed as follows:

$$\begin{array}{rcl} & X_1 = X_{1,1} & X_{1,0} \\ + & X_2 = X_{2,1} & X_{2,0} \\ \hline & \gamma = \gamma_1 & \gamma_0 \end{array}, \tag{24}$$

to calculate the probability of $F(X_1, X_2) = X_1+X_2$, the intuitive method is to observe through the computation process in Figure 2 in which the respective probabilities of γ = 0, 1, 2, and 3 are 1/4, 1/4, 1/4, and 1/4:

$$\begin{aligned} P_r\{F(X_1, X_2) = 0_{10} = 00\} = 4/16 = 1/4 \\ P_r\{F(X_1, X_2) = 1_{10} = 01\} = 4/16 = 1/4 \\ P_r\{F(X_1, X_2) = 2_{10} = 10\} = 4/16 = 1/4 \\ P_r\{F(X_1, X_2) = 3_{10} = 11\} = 4/16 = 1/4 \end{aligned}. \tag{25}$$

When the probabilities of the γ values can be determined, the distribution of the PLFM can be expressed. From the aforementioned relations, the calculation of $F(X_1, X_2) = X_1 + X_2$ is similar to the exhaustive method in which various X_1 and X_2 are input to obtain γ values. Moreover, X_1 and X_2 are both 2-bit random variables and each consists of four types of values; thus, $2^2 \times 2^2 = 16$ combinations of X_1 and X_2 are required for calculating $F(X_1, X_2) = X_1 + X_2$.

Therefore, the complexity of this method is expressed as follows, where n represents the number of bits in each random variable and k denotes the number of random variables in $F(X_1, X_2, \ldots, X_k)$.

$$\underbrace{2^n \times 2^n \times \ldots \times 2^n}_{k} = 2^{kn}. \tag{26}$$

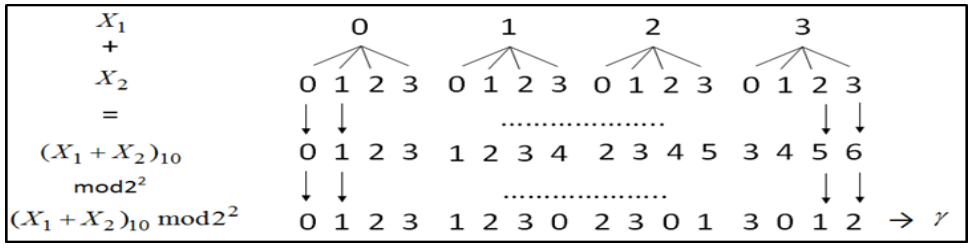

Figure 2. Intuitive calculations of $F(X_1, X_2) = X_1+X_2 = \gamma$

Noticeably, the complexity of using intuitive methods for obtaining PLFM distributions is high. Subsequent analysis shows that the aforementioned calculations are viewed from a variable-wise perspective. However, Maximov and Johansson provided another viewpoint in their paper and changed to a bit-wise perspective in the entire calculation. Through analysing the carry between each variable bits and using the concept of Markov chain, all transfer matrices $M_{r_t|t}$ were constructed. The probabilities of various γ values are obtained statistically through the entire process for which a calculation complexity of only $O(n\theta_{max}2^n)$ is required. Moreover, the complexity is only related to the length of n and unrelated to k, which can be regarded as being removed from the original index location, thereby substantially reducing the complexity. To explain the method of Maximov and Johansson, the aforementioned example is used and presented from a bit-wise perspective.

$$\begin{array}{rl} \sigma = \sigma_1, \ \sigma_0 & \leftarrow \text{Carry value} \\ +X_1 = x_{1,1} \ x_{1,0} & \\ +X_2 = x_{2,1} \ x_{2,0} & \\ \hline \gamma = \gamma_1 \ \gamma_0 & \end{array}$$

Figure 3. Calculation of $F(X_1, X_2) = X_1+X_2 = \gamma$ and variable σ

Figure 3 shows an additional variable σ because the modular addition in the $F(X_1, X_2) = X_1+X_2$ function results in a carry between bits during calculations. Additionally, because of modular computations, a carry from the most significant bit can be viewed as the carry on the least significant bit. Thus, the existence of a carry variable σ must be considered for each bit. This example consists of two bits, thus yielding the carry variables σ_0 and σ_1. Figure 4 shows the addition and carry conditions of the second bit $x_{n,1}$, which are identical to those of the first bit; therefore, additional explanation is not provided.

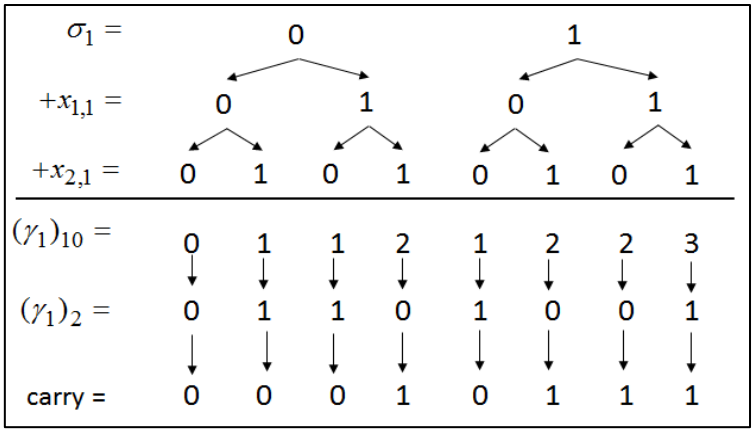

Figure 4. The calculation of $F(X_1, X_2) = X_1+X_2 = \gamma$ from a bit-wise perspective

The statistical data obtained in Figure 4 can be used to calculate all transfer matrices $M_{r_i|t}$. In this example, $M_{r_i|t}$ is expressed as follows:

$$M_{r_i|t} = \begin{bmatrix} NN & YN \\ NY & YY \end{bmatrix}. \tag{27}$$

This relation refers to the following conditions:
NN: No carry enters during this stage and no carry is provided for the next stage.
NY: No carry enters during this stage but carry is provided for the next stage.
YN: Carry enters during this stage but no carry is provided for the next stage.
YY: Carry enters during this stage and carry is also provided for the next stage.

Theorem1: Assigning a PLFM function F(X_1,…,X_k) and a fixed value $\gamma \in Z_{2^b}$ yields the following relation:

$$P_r\{F(X_1, X_2, ..., X_k) = \gamma\} = \frac{1}{2^{k \cdot n}}(11...1) \times \left(\sum_{t=n-1}^{0} M_{r_i|t} \right) \times (10...0)^T, \tag{28}$$

where $M_{r_i|t}$ is a ($\theta_{max} \times \theta_{max}$) matrix; thus, the following results can be obtained through calculations by using Equation (28):

$$\begin{aligned}
P_r\{F(X_1, X_2) = 00_2\} &= \frac{1}{2^{2 \cdot 2}}(11) \times M_{0|1}\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \times M_{0|0}\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{4} \\
P_r\{F(X_1, X_2) = 01_2\} &= \frac{1}{2^{2 \cdot 2}}(11) \times M_{0|1}\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \times M_{1|0}\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{4} \\
P_r\{F(X_1, X_2) = 10_2\} &= \frac{1}{2^{2 \cdot 2}}(11) \times M_{1|1}\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \times M_{0|0}\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{4} \\
P_r\{F(X_1, X_2) = 11_2\} &= \frac{1}{2^{2 \cdot 2}}(11) \times M_{1|1}\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \times M_{1|0}\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{4}
\end{aligned} \tag{29}$$

From an in-depth perspective, the presence of these previously obtained statistical information enables a substantial reduction of the complexity in calculating distributions of PLFMF(X_1, X_2) = X_1+X_2. In exchange, additional memory or hard drive space is used to store these transfer matrices. Based on the amount of calculations in Equation (28), the complexity is only O(n$\theta_{max}2^n$).

3.2. The distribution of specific PLFM outputs

The aforementioned descriptions show that transfer matrices are calculated using statistical methods even when the input is assumed to be a uniform distribution. For example, the probabilities of σ_1, $X_{1,1}$, and $X_{2,1}$ being 0 or 1 are all 1/2 (Figure 4). To counter problems when the input is not a uniform distribution, Maximov and Johansson proposed using the fast Fourier and fast Harley transforms (FFT and FHT) to obtain PLFM distributions. However, this method is currently only suitable for two special PLFMs that consist of either ⊞ or □ computations. Whether this method can be used to calculate other types of computations remains an open problem.

Similarly, this study provided two examples to explain the entire process of calculating PLFM distributions.

Example 2: Assuming that two 1-bit random variables X and Y exist in which the probabilities of $X = 0$ and 1 and $Y = 0$ and 1 are respectively 0.6 and 0.4 and 0.3 and 0.7, this is expressed as $[D_X] = \{p_X(0), p_X(1)\} = \{0.6, 0.4\}$ and $[D_Y] = \{p_Y(0), p_Y(1)\} = \{0.3, 0.7\}$. Intuitively, the probabilities of $[D_Z] = [D_X \boxplus D_Y]$ are shown as follows:

$0 + 0 = 0 \bmod 2 \rightarrow p_{00} = 0.6 \times 0.3 = 0.18$

$0 + 1 = 1 \bmod 2 \rightarrow p_{01} = 0.6 \times 0.7 = 0.42$

$1 + 0 = 1 \bmod 2 \rightarrow p_{10} = 0.4 \times 0.3 = 0.12$

$1 + 1 = 0 \bmod 2 \rightarrow p_{11} = 0.4 \times 0.7 = 0.28$

where p_{00} and p_{11} represent the probability of $Z = 0$ when $X = 0$ and $Y = 0$ and $X = 1$ and $Y = 1$, respectively. Thus, the probability of $p_{00} + p_{11} = 0.46$ when $Z = 0$ was ultimately obtained. Similarly, the probability of $Z = 1$ was $p_{01} + p_{10} = 0.54$; therefore, the distribution of Z can be expressed as follows:

$$[D_Z] = \{p_Z(0), p_Z(1)\} = \{0.46, 0.54\}. \quad (30)$$

Using the method proposed by Maximov and Johansson results in the following expression:

$$[D_Z] = [D_X \boxplus D_Y] = FFT_n^{-1}(FFT_n([D_X]) \cdot FFT_n([D_Y])). \quad (31)$$

Fast multiplication of two polynomials can be done via *Fast Fourier Transform* (*FFT*). To calculate Z distributions, the following expressions are used:

$FFT_1([D_X]) = FFT_1(\{0.6, 0.4\}) = \{1, 0.2\}$

$FFT_1([D_Y]) = FFT_1(\{0.3, 0.7\}) = \{1, -0.4\}$

$FFT_1([D_X]) \cdot FFT_1([D_Y]) = \{1, -0.08\}$

$[D_Z] = FFT_1^{-1}(\{1, -0.08\}) = \{0.46, 0.54\}$

where $FFT_1([D_X])$ represents the FFT computation results obtained for $[D_X]$. The subscript of FFT_1 denotes that the random variable X is a 1-bit variable. Thus, the aforementioned results proved that D_Z distributions can be obtained through FFT.

FFT computations often require float numbers and thus are comparatively less suitable in practice because complex and float numbers are difficult to process. Instead, computation using *Fast Hadamard Transform* (*FHT*) is preferred. FHT is similar to FFT because both algorithms are capable of converting signals into frequency domains. These processes can be regarded as signal processing mechanisms in the current scenario. The two processes are identical in complexity, which are both $O(q \cdot \log q)$ where q represents the FFT (or FHT) input space. FHT is comparatively faster in practice because it does not require complex number i during conversion and float numbers; moreover, FHT is simple to program. The following shows the FHT method proposed by Maximov and Johansson:

$$[D_Z] = [D_X \square D_Y] = 1/2^n \cdot FHT_n(FHT_n([D_X]) \cdot FHT_n([D_Y])). \quad (32)$$

FHT is comparatively advantageous in calculations. Thus, to substitute Equation (32) for calculating the distribution of the $\boxplus$, Equation (33) is often used to approximate the conversion between $\boxplus$ and $\square$, in which N can be regarded as noise. This conversion was also previously adopted by Englund and Maximov [6].

$$X_1 \boxplus X_2 \rightarrow X_1 \square X_2 \square N. \quad (33)$$

Lastly, the following equation can be used to obtain D_Z when calculating the distribution of $Z = C_1X_1 \square C_2X_2 \square \ldots \square C_kX_k$ using FHT:

$$D_{(Z = C1X1 \square C2X2 \square \ldots \square CkXk)} = 1/2^n \cdot FHT_n(FHT_n([D_{C1X1}]) \cdot \ldots \cdot FHT_n([D_{CkXk}])). \quad (34)$$

Maximov and Johansson defined PLFM functions commonly used in various types of linear attack methods and mentioned that the functions used in the linear attack studies were merely sets of the PLFM function. To calculate the distribution of the PLFM function outputs, Maximov and Johansson proposed two methods for calculating the distribution of PLFM function outputs when the PLFM function input is uniformly or arbitrarily distributed. These two methods substantially increased the efficiency of cryptanalysis. This study also found that some studies have adopted techniques proposed by Maximov and Johansson for cryptanalysis [6, 7, 11].

4. SECURITY ANALYSIS OF THE RABBIT STREAM CIPHER

4.1. Distinguishing attacks on the Rabbit

Distinguishing attack refers to the determination of whether the output probability distribution is random variables or a type of stream cipher-generated special distribution pattern when the attacker collects a series of unknown stream data [10, 12, 13, 14, 15]. In 2007, Aumasson [8] found that key streams generated by the Rabbit exhibited bias in which some bits consisted bias values greater than $2^{-123.5}$. Consequently, the distinguisher can be used to differentiate random keys and Rabbit streams through approximately 2^{247} samples of key stream. Although the complexity of this substantially exceeds the 2^{128} values required to compromise in brute-force attacks, Aumasson was the first to adopt distinguishing attack on Rabbit. In 2008, Lu et al. [7] also proposed distinguishing attack for the Rabbit. In the attack, FFT was used to calculate the probability distributions of variables such as $x_{j,t}^i$ and $g_{j,t}^i$ obtain the bias for every bit of the key stream. The complexity of this algorithm was 2^{158}, which are approximately 2^{89} less than the calculation bias of the method proposed by Aumasson. Currently, Lu's method is the simplest method for distinguishing attack.

Additionally, Lu et al. were the first to propose an extended multi frame attack capable of compromising Rabbit keys. This attack also uses the key stream bias information. The overall compromised complexity requires $O(2^{32})$ of pre computation resource, $O(2^{32})$ of memory space, and $O(2^{97.5})$ of compromise process time. However, this method requires strict assumptions, $2^{51.5}$ frames, and prior knowledge of differential information such as $2^{51.5}$numbers of $d_{j,1}^i$ and $d_{j,2}^i$ to obtain the keys; the frame s^i is composed of n 128-bit key streams: $s^i = \{s_1^i, s_2^i, ..., s_n^i\}$. The variables $d_{j,1}^i$ and $d_{j,2}^i$ are calculated using differential cryptanalysis. The greatest challenge in current research is to obtain the keys without knowing $d_{j,1}^i$ and $d_{j,2}^i$.

4.2. The distinguishing attack method proposed by Lu et al.

Lu et al. first proposed Equation (11) in the following expression through observing the overall architecture of the Rabbit algorithm:

$$g_{j,t} = y_{j,t}^2 \oplus (y_{j,t}^2 >> 32), \tag{35}$$

when $j = 0,\ldots,7$, the following relation is established:

$$y_{j,t} = x_{j,t} + c_{j,t+1} \bmod 2^{32}. \tag{36}$$

Assuming $x_{j,t}$ and $c_{j,t+1}$ are uniform distributions and mutually independent, $y_{j,t}$ must also be a uniform distribution. Thus, through the complexity of $O(2^{32})$, $g_{j,t}$ distribution can be obtained and expressed using $D(g)$.

Equations (3) to (10) show that when js are even numbers, $x_{j,t+1}$ distributions are identical and are expressed using $D(x_0)$. When j is odd number, $x_{j,t+1}$ distributions are also identical and are expressed using $D(x_1)$.

Using the technique presented in Maximov and Johansson for calculating $D(x_0)$, the following expression can be obtained:

$$\begin{aligned} D(x_0) &= D(g) \otimes D(g <<< 16) \otimes D(g <<< 16) \\ &= FFT^{-1}(FFT(D(g)) \cdot FFT(D(g <<< 16)) \cdot FFT(D(g <<< 16)) \end{aligned} . \quad (37)$$

Similarly:

$$\begin{aligned} D(x_1) &= D(g) \otimes D(g <<< 8) \otimes D(g) \\ &= FFT^{-1}(FFT(D(g)) \cdot FFT(D(g)) \cdot FFT(D(g <<< 8)) \end{aligned} . \quad (38)$$

After obtaining $D(x_0)$ and $D(x_1)$, the bias of each x_i (for $i = 0,\ldots,7$) can be calculated. For example, $\varepsilon(x_5^{[16]}) \approx 2^{-49.56}$ in Maximov and Johansson represents that the bias of the 16th bit of x_5 is $2^{-49.56}$, which is the current optimal result in the field. When x_0 and x_5 are assumed to be mutually independent, $\varepsilon(s_t^{[0]}) = \varepsilon(x_0^{[0]}) \cdot \varepsilon(x_5^{[16]}) \approx 2^{-91.41}$ can be obtained through pilling-up lemma; this indicates that a complexity of $O(2^{192.82})$ is necessary for executing the distinguishing attack. This reduced the complexity by approximately $O(2^{52.34})$ from that presented in Aumasson [8].

Baigneres, Junod, and Vaudenay [9] proposed the concept of optimised distinguishers in their paper; through the simultaneous analysis of bias for all bits, attack complexity was reduced to approximately $O(1/\Delta(D))$. In the complexity expression, $\Delta(D)$ is expressed as follows:

$$\Delta(D) = 2^r \sum_a (D(a) - 2^{-r})^2 . \quad (39)$$

Relevant calculation results are listed in Tables 1 and Table 2. Based on the results, the following complexities can be calculated:

$$\Delta(D(s^{[15...0]})) = \Delta(D(s^{[47...32]})) = \Delta(D(s^{[79...64]})) = \Delta(D(s^{[111...96]})) \approx 2^{-158}, \quad (40)$$

$$\Delta(D(s^{[31...16]})) = \Delta(D(s^{[63...48]})) = \Delta(D(s^{[95...80]})) = \Delta(D(s^{[127...112]})) \approx 2^{-160}. \quad (41)$$

Equations (40) and (41) show that when the 16 bits of each key stream s_t can be simultaneously analysed, the distinguishing attack requires a complexity of approximately $O(2^{158})$, exhibiting a reduction of approximately $O(2^{35})$ compared with the previous complexity of $O(2^{192.82})$.

Table 1. Calculation results

Distribution D	$D(g)$	$D(x_1^{[31...16]})$	$D(x_1^{[15...0]})$	$D(x_1^{[31...0]})$	$D(x_0^{[15...0]})$	$D(s^{[15...0]})$
SEI $\Delta(D)$	1.01	2^{-76}	2^{-100}	2^{-45}	2^{-100}	2^{-158}

Table 2. Calculation results (continued)

Distribution D	$D(x_0^{[31\cdots16]})$	$D(x_0^{[15\cdots0]})$	$D(x_0^{[31\cdots0]})$	$D(x_1^{[15\cdots0]})$	$D(s^{[31\cdots16]})$
SEI $\Delta(D)$	2^{-76}	2^{-100}	2^{-45}	2^{-100}	2^{-160}

5. CONCLUSION

This research investigated Rabbit stream cipher-related attacks and fast probability distribution algorithms. Based on current literature, the method proposed by Lu et al. remains the optimal distinguishing attack, which requires a complexity of merely $O(2^{158})$. However, this is still greater than the complexity of $O(2^{128})$ needed in the brute-force method. Although Lu et al. also proposed an extended multi frame attack method for compromising keys; the assumptions in this attack are not easily attained. Thus, the security of the Rabbit remains strong. Furthermore, the Rabbit and Salsa20 stream ciphers tied for the highest score in the eSTREAM project; therefore, highly effective attack methods for compromising the Rabbit are unlikely to be developed within a short time frame.

ACKNOWLEDGEMENT

This research was supported by MOE Teaching Practice Research Program.

REFERENCES

[1] Boesgaard, M., Vesterager, M., Pedersen, T., Christiansen, J., and Scavenius, O. Rabbit: A new high-performance stream cipher. In T. Johansson, editor. *Fast Software Encryption 2003* (*FSE 2003*), 2887, pp. 307-329. DOI: 10.1007/978-3-540-39887-5_23.

[2] Cryptico A/S. *Algebraic analysis of Rabbit*, white paper, 2003.

[3] Cryptico A/S. *Differential properties of the g-funcion*, white paper, 2003.

[4] Cryptico A/S. *Security analysis of the IV-setup for Rabbit*, white paper, 2003.

[5] Maximov, A. and Johansson, T. Fast computation of large distributions and its cryptographic applications. *ASIACRYPT 2005*, *Lecture Notes in Computer Science*, 2005, 3788, pp. 313-332. DOI: 10.1007/11593447_17.

[6] Englund, H. and Maximov, A. Attack the Dragon. Progress in Cryptology - *INDOCRYPT 2005*. *Lecture Notes in Computer Science*, vol. 3797. Springer, Berlin, Heidelberg. DOI: 10.1007/11596219_11.

[7] Lu, Y., Wang, H. and Ling, S. Cryptanalysis of Rabbit. *Information Security: 11th International Conference* (ISC 2008), pp. 204-214. DIO: /10.1007/978-3-540-85886-7_14.

[8] Aumasson, J. P. On a bias of Rabbit. *SASC 2007*. http://www.ecrypt.eu.org/stvl/sasc2007/.

[9] Baigneres, T., Junod, P. and Vaudenay, S. How far can we go beyond linear cryptanalysis. *ASIACRYPT'04*, *Lecture Notes in Computer Science*, 2004, 3329, pp. 432-450.

[10] Daemen, J. *Cipher and hash function design strategies based on linear and differential cryptanalysis*. PHD thesis, KU Leuven, March 1995.

[11] Chain, K. The Study and Security Analysis of HC Stream Cipher, *Journal of Convergence Information Technology*, vol. 6, no. 6, 2011, pp. 439-454. DOI: 10.4156/jcit.vol6.issue6.44

[12] Scavenius, O., Boesgaard, M., Pedersen, T., Christiansen, J. and Rijmen, V. Periodic properties of counter assisted stream cipher. In T. Okamoto, editor. *CT-RSA 2004*, 2964, pp. 39-53.

[13] Shamir, A. and Tsaban, B. Guaranteeing the diversity of number generators. *Information and Computation*, vol. 171, no. 2, 2001, pp. 350-363. DOI: 10.1006/inco.2001.3045

[14] Li, S., Hu, Y., Zhao, Y. and Wang, Y., Algebraic Cube Attack on Sfinks Stream Cipher, *International Journal on Information*, vol. 15, no. 10, 2012, pp. 4295-4302.

[15] Jeong, K., Sung, J., Hong, S. and Lee, C., A New Approach of Differential Fault Analysis on Block Ciphers with S-box, *International Journal on Information*, vol. 16(3A), 2013, pp. 1915-1928.

[16] Boesgaard M., Vesterager M., Zenner E., The Rabbit Stream Cipher. In: Robshaw M., Billet O. (eds) New Stream Cipher Designs. *Lecture Notes in Computer Science*, 2008, Vol. 4986. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-68351-3_7

[17] Jenog, K.; Lee, Y.; Sung, J.; and Hong, S. (2010). A note on improved fast correlation attacks on stream ciphers. *IACR Cryptology ePrint Archive*, 201021.

[18] Shweta S Tadkal and Mahalinga V Mandi, Secure Trasmission of Data using Rabbit Algorithm, *International Research Journal of Engineering and Technology*, vol. 04, no. 05, pp. 2395-0072, 2017.

[19] Prathima N., Chetan S. and Syed M Rehman., ASIC Implementation of Rabbit Stream Cipher Encryption for Data, *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, 15-16 November 2019. DOI: 10.1109/WIECON-ECE48653.2019.9019903

Information about the authors:

Kai Chain received the M.S. degree in Electrical Engineering from National Taiwan University in 2003. He is an associate professor in the Department of Intelligent Network Technology at the I-SHOU University. He received the Ph.D. degree from the Institute of Computer Science and Communication Engineering at National Cheng Kung University under Profs. Chi-Sung Laih and Jar-Ferr Yang in 2015. His research interests include Network and Information Security, with a concentration on applied Cryptography.

Manuscript received on 06 February 2024