Cryptanalysis of RAKAPOSHI Stream Cipher[†]

Lin Ding, Jie Guan

Zhengzhou Information Science and Technology Institute, China

E-mail: dinglin_cipher@163.com; guanjie007@163.com

Abstract. RAKAPOSHI is a hardware oriented stream cipher designed by Carlos Cid et al. in 2009. The stream cipher is based on Dynamic Linear Feedback Shift Registers, with a simple and potentially scalable design, and is particularly suitable for hardware applications with restricted resources. The RAKAPOSHI stream cipher offers 128-bit security. In this paper, we point out some weaknesses in the cipher. Firstly, it shows that there are 2^{192} weak (key, IV) pairs in RAKAPOSHI stream cipher. Secondly, for weak (key, IV) pairs of RAKAPOSHI, they are vulnerable to linear distinguishing attack and algebraic attack. Finally, we propose a real time related key chosen IV attack on RAKAPOSHI. The attack on RAKAPOSHI recovers the 128-bit secret key of with a computational complexity of 2^{37}, requiring 47 related keys, 2^8 chosen *IV*s and $2^{14.555}$ keystream bits. The success probability of this attack is 0.999, which is quite close to 1. The experimental results corroborate our assertion.

Keywords: Cryptanalysis; linear distinguishing attack; algebraic attack; related key chosen IV attack; RAKAPOSHI; stream cipher.

1 Introduction

Stream ciphers are symmetric encryption algorithms based on the concept of pseudorandom keystream generator. Although it seems rather difficult to construct a very fast and secure stream cipher, some efforts to achieve this have recently been deployed. The NESSIE project [1] launched in 1999 by the European Union did not succeed in selecting a secure enough stream cipher. In 2005, the European project ECRYPT decided to launch a competition to identify new stream ciphers that might be suitable for widespread adoption. This project is called eSTREAM [2] and received 35 submissions. Those candidates are divided into software oriented and hardware oriented stream ciphers. Hardware oriented stream ciphers should be suitable for deployment on passive RFID tags or low-cost devices such as might be used in sensor networks. Such devices are exceptionally constrained in computing potential because of the number of logic gates available or the amount of power that might realistically be available.

In 2009, Carlos Cid et al. [3] proposed a new hardware-oriented stream cipher called RAKAPOSHI, which aims to complement the current eSTREAM portfolio of hardware-oriented stream ciphers. The RAKAPOSHI stream cipher offers 128-bit security. The designers claimed that the cipher design and

[†] The paper had been submitted and is under review now. The paper was done independently of two similar works, i.e., [8] and [9].

security evaluation incorporates lessons learned during the several years of extensive analysis in the eSTREAM process, and thus RAKAPOSHI is less likely to be susceptible to more recent attacks, such as initialization attacks.

RAKAPOSHI is based on Dynamic Linear Feedback Shift Registers (DLFSR), with a simple and potentially scalable design, and is particularly suitable for hardware applications with restricted resources. A Dynamic Linear Feedback Shift Register scheme is a general construction consisting usually of two registers: the first subregister A, is clocked regularly and updated using a fixed mapping. Subregister B, is updated using a linear mapping, which varies with time and depends of the state in register A. The design can be seem as a generalization of constructions found in early designs, including the stop-and-go generator [4], LILI [5], dynamic feedback polynomial switch [6], and K2 [7]. In fact, the RAKAPOSHI stream cipher is in fact a successor of the K2 stream cipher, but aiming at low-cost hardware implementations.

In this paper, we point out some weaknesses in the cipher. Firstly, the result shows that there are 2^{192} weak (key, IV) pairs in RAKAPOSHI stream cipher. Secondly, for weak (key, IV) pairs of RAKAPOSHI, they are vulnerable to linear distinguishing attack and algebraic attack. Finally, we proposed a real time related key chosen IV attack on RAKAPOSHI. The time complexity of our related key chosen IV attack on RAKAPOSHI is about 2^{34}, requiring 39 related keys on average. The results show that there exist some weaknesses in the cipher. See also [8] and [9] for two works that were done independently of our results.

This paper is organized as follows. In Section 2, we briefly describe RAKAPOSHI stream cipher. We discuss the existence of weak (key, IV) pairs, and present linear distinguishing attack and algebraic attack against the weak (key, IV) pairs of RAKAPOSHI in Section 3. Section 4 proposes a real time related key chosen IV attack on RAKAPOSHI. The Section 5 concludes this paper.

2 Brief Description of RAKAPOSHI

The RAKAPOSHI stream cipher consists of three main building blocks, namely a 128-bit Non-Linear Feedback Shift Register (denoted as register A), a 192-bit Linear Feedback Shift Register (denoted as register B), and a non-linear filter function over $GF(2^8)$. It uses two bits from the state of the NLFSR to select and dynamically modify the linear feedback function of the LFSR. The keystream is produced by combining the output of both registers with the output of the non-linear filter function. An overview of the different blocks used in the stream cipher can be found in Fig. 1.

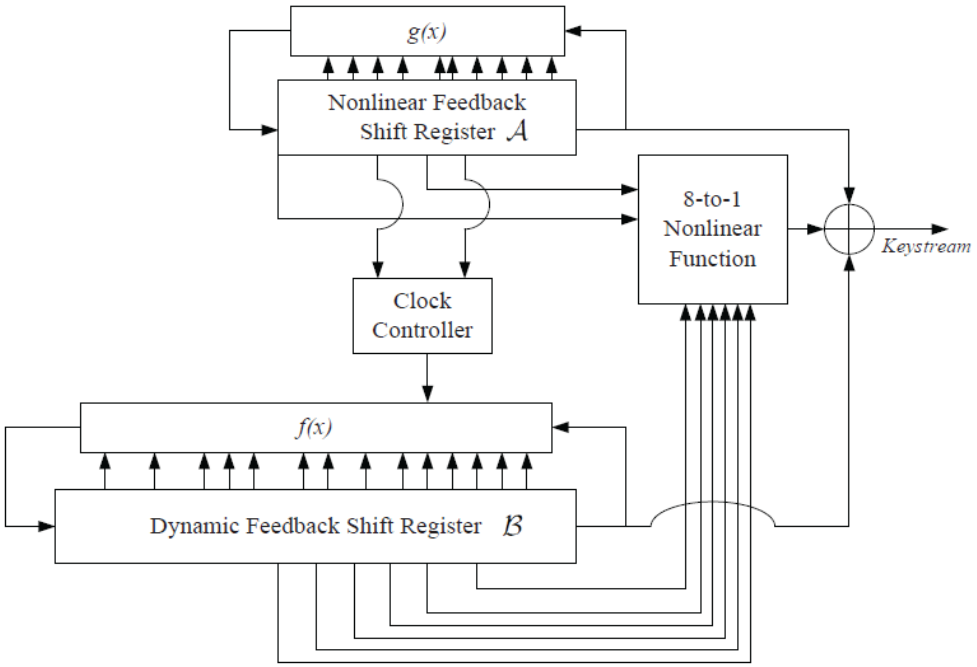

Figure 1. The structure of RAKAPOSHI stream cipher

We denote by $A^t = (a_t, a_{t+1}, \cdots, a_{t+127})$ the contents of the LFSR at time *t*. Similarly, the content of the NFSR is denoted by $B^t = (b_t, b_{t+1}, \cdots, b_{t+127})$ at time *t*.

Non-Linear Feedback Shift Register A. The register A is a 128-bit NLSFR, defined using the following recurrence relation.

$$\begin{aligned} a_{t+128} &= g(a_t, a_{t+6}, a_{t+7}, a_{t+11}, a_{t+16}, a_{t+28}, a_{t+36}, a_{t+45}, a_{t+55}, a_{t+62}) \\ &= 1 \oplus a_t \oplus a_{t+6} \oplus a_{t+7} \oplus a_{t+11} \oplus a_{t+16} \oplus a_{t+28} \oplus a_{t+36} \oplus a_{t+45} \oplus a_{t+55} \oplus a_{t+62} \oplus a_{t+7}a_{t+45} \oplus \\ &\quad a_{t+11}a_{t+55} \oplus a_{t+7}a_{t+28} \oplus a_{t+28}a_{t+55} \oplus a_{t+6}a_{t+45}a_{t+62} \oplus a_{t+6}a_{t+11}a_{t+62} \end{aligned}$$

Linear Feedback Shift Register B. The register B is a 192-bit dynamic LFSR. Register A is used to select and dynamically modify the feedback function of LFSR B using two bits from the state of register A, and as a result, Register B, which can use four different linear recursive functions, presents an irregular updating mechanism. Let c_0 and c_1 be the 42nd and 90th bits of register A at time *t*, respectively (that is, $c_0 = a_{t+41}$ and $c_1 = a_{t+89}$). Then LFSR B at time *t* is defined by the following recurrence relation.

$$\begin{aligned} b_{t+192} &= f(c_0, c_1, b_t, b_{t+14}, b_{t+37}, b_{t+41}, b_{t+49}, b_{t+51}, b_{t+93}, b_{t+107}, b_{t+120}, b_{t+134}, b_{t+136}, b_{t+155}, b_{t+158}, b_{t+176}) \\ &= b_t \oplus b_{t+14} \oplus b_{t+37} \oplus b_{t+41} \oplus b_{t+49} \oplus b_{t+51} \oplus b_{t+93} \oplus \overline{c_0} \cdot \overline{c_1} \cdot b_{t+107} \oplus \overline{c_0} \cdot c_1 \cdot b_{t+120} \oplus c_0 \cdot \overline{c_1} \cdot b_{t+134} \oplus \\ &\quad c_0 \cdot c_1 \cdot b_{t+136} \oplus \overline{c_0} \cdot b_{t+155} \oplus c_0 \cdot b_{t+158} \oplus b_{t+176} \end{aligned}$$

Where $\overline{c_i} = c_i \oplus 1$ represents the negation of c_i.

Non-Linear Filter. The 8-to-1 non-linear filter function is the same function used as the non-affine component of the AES S-Box. This function is a balanced Boolean function, with polynomial representation (ANF) of degree 7. In the RAKAPOSHI stream cipher, the input bits for this function are extracted from both registers A and B, as

$$s_t = v(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128})$$

The explicit polynomial expression of the function $v(\bullet)$ is given in the Appendix A.

Initialization Process. The RAKAPOSHI stream cipher supports key size of 128 bits and IV size of 192 bits. Before the generation of the cipher keystream, the cipher is initialized with the secret key and a selected IV. The initialization process of RAKAPOSHI is done as follows.

Firstly, the secret key $K = (k_0, \cdots, k_{127})$ and $IV = (iv_0, \cdots, iv_{191})$ are loaded into the NLFSR and DLFSR, respectively, as follows.

$$(a_0, \cdots, a_{127}) \leftarrow (k_0, \cdots, k_{127})$$
$$(b_0, \cdots, b_{191}) \leftarrow (iv_0, \cdots, iv_{191})$$

Secondly, the cipher then clocks 448 times with the output of the filter function $v(\bullet)$ (i.e., s_t) being fed back into the cipher state. This process is divided into two stages:

- In the first stage of the initialization, the cipher runs for 320 cycles, with the output of the non-linear filter function $v(\bullet)$ being fed back into the register B.
- In the second stage of the initialization, the cipher runs for further 128 cycles, with the output of the non-linear filter function $v(\bullet)$ being fed back into the register A.

At the end of the initialization, the cipher internal state is $S^{448} = (A^{448}, B^{448})$, and it is ready to produce the first keystream bit z_0.

Keystream Generation. The cipher outputs one keystream bit at each cycle. Given the cipher state $S_t = (A_t, B_t)$ at time *t*, the cipher operates as follows:

- The keystream bit z_t is computed as $z_t = a_t \oplus b_t \oplus s_t$ and is output.
- $c_0, c_1 \in A^t$ are used to update the register B.
- Registers A and B are updated to obtain A^{t+1} and B^{t+1}, respectively.

3 Weak (key, IV) Pairs of RAKAPOSHI

3.1 The Existence of the Weak (key, IV) Pairs

RAKAPOSHI uses a dynamic linear feedback shift register to ensure good cryptographic properties. However, there is an “lock-up” state for the register A That is, if the initialization process leaves A in the all ones state, then it will remain permanently in that state, and register B will become a simple linear feedback shift register, which is the only active block of the cipher. As well known, keystream sequences generated by a single LFSR and a non-linear filter function are vulnerable to distinguishing attack and algebraic attack. Consequently, if the key and IV pair results in this “lock-up” state for the register A, we define the key and IV pair as a **weak (key, IV) pair**.

Now we reveal the existence of the weak (key, IV) pairs in RAKAPOSHI stream cipher. For a weak (key, IV) pair, the state A^{448} is the all one state $(1,1,\cdots,1)$. Then our following purpose is to obtain an available S^0 from known S^{448} in reverse. According to the structure of RAKAPOSHI, we present an algorithm to compute the weak (key, IV) pairs.

Finding Weak (key, IV) pairs Algorithm for RAKAPOSHI

---

1. Set A^{448} be the all one state $(1,1,\cdots,1)$, and select B^{448} randomly.

2. From t = 447 to t = 320 do

(2.a) Compute $c_0 = a_{t+41}$ and $c_1 = a_{t+89}$.

(2.b) Compute $s_t = v(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128})$.

(2.c) Compute

$$b_t = b_{t+192} \oplus b_{t+14} \oplus b_{t+37} \oplus b_{t+41} \oplus b_{t+49} \oplus b_{t+51} \oplus b_{t+93} \oplus \overline{c_0} \cdot \overline{c_1} \cdot b_{t+107} \oplus \overline{c_0} \cdot c_1 \cdot b_{t+120} \oplus c_0 \cdot \overline{c_1} \cdot b_{t+134} \oplus c_0 \cdot c_1 \cdot b_{t+136} \oplus \overline{c_0} \cdot b_{t+155} \oplus c_0 \cdot b_{t+158} \oplus b_{t+176}$$

(2.d) Compute

$$a_t = a_{t+128} \oplus 1 \oplus a_{t+6} \oplus a_{t+7} \oplus a_{t+11} \oplus a_{t+16} \oplus a_{t+28} \oplus a_{t+36} \oplus a_{t+45} \oplus a_{t+55} \oplus a_{t+62} \oplus a_{t+7}a_{t+45} \oplus a_{t+11}a_{t+55} \oplus a_{t+7}a_{t+28} \oplus a_{t+28}a_{t+55} \oplus a_{t+6}a_{t+45}a_{t+62} \oplus a_{t+6}a_{t+11}a_{t+62} \oplus s_t$$

3. From t = 319 to t = 0 do

(3.a) Compute $c_0 = a_{t+41}$ and $c_1 = a_{t+89}$.

(3.b) Compute $s_t = v(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128})$.

(3.c) Compute

$$b_t = b_{t+192} \oplus b_{t+14} \oplus b_{t+37} \oplus b_{t+41} \oplus b_{t+49} \oplus b_{t+51} \oplus b_{t+93} \oplus \overline{c_0} \cdot \overline{c_1} \cdot b_{t+107} \oplus \overline{c_0} \cdot c_1 \cdot b_{t+120} \oplus c_0 \cdot \overline{c_1} \cdot b_{t+134} \oplus c_0 \cdot c_1 \cdot b_{t+136} \oplus \overline{c_0} \cdot b_{t+155} \oplus c_0 \cdot b_{t+158} \oplus b_{t+176} \oplus s_t$$

(3.d) Compute

$$a_t = a_{t+128} \oplus 1 \oplus a_{t+6} \oplus a_{t+7} \oplus a_{t+11} \oplus a_{t+16} \oplus a_{t+28} \oplus a_{t+36} \oplus a_{t+45} \oplus a_{t+55} \oplus a_{t+62} \oplus a_{t+7}a_{t+45} \oplus a_{t+11}a_{t+55} \oplus a_{t+7}a_{t+28} \oplus a_{t+28}a_{t+55} \oplus a_{t+6}a_{t+45}a_{t+62} \oplus a_{t+6}a_{t+11}a_{t+62}$$

4. Output the initial state $S^0 = (A^0, B^0)$.

According the algorithm above, it is easy to see that for each value of B^{448}, there exists a unique weak (key, IV) pair. Since B^{448} can be randomly selected in this algorithm, there are certainly 2^{192} weak (key, IV) pairs in RAKAPOSHI stream cipher. One example is illustrated in Table 1.

Table 1. Weak (key, IV) pair of RAKAPOSHI

Key	0x81094ef5d124b47934d15b5228e5b8fa
IV	0x4f9fe37b7c5a87f592ed1e89175e0f6d6634eada6411eaf3
A^{448}	0xffffffffffffffffffffffffffffffffffffffffffffffff
B^{448}	0x7a72bd70c98392497034e2ba72c56fb629adfc8265f4aada

3.2 Linear Distinguishing Attack and Algebraic Attack on Weak (key, IV) Pairs

In distinguishing attacks, one attempts to distinguish the keystream sequence from a purely random sequence. For weak (key, IV) pairs of RAKAPOSHI, we tried to construct linear distinguishers, by considering linear approximations of functions used in the stream cipher. For each weak (key, IV) pair of RAKAPOSHI, the internal states of the register A are all ones after the initialization process, the recurrence relation of LFSR B will remain permanently as follows.

$$b_{t+192} = b_t \oplus b_{t+14} \oplus b_{t+37} \oplus b_{t+41} \oplus b_{t+49} \oplus b_{t+51} \oplus b_{t+93} \oplus b_{t+136} \oplus b_{t+158} \oplus b_{t+176}$$

Now, the number of terms in the recurrence function of LFSR B is 11. We can construct a linear distinguisher using 11 keystream bits as follows.

$$D: z_t \oplus z_{t+14} \oplus z_{t+37} \oplus z_{t+41} \oplus z_{t+49} \oplus z_{t+51} \oplus z_{t+93} \oplus z_{t+136} \oplus z_{t+158} \oplus z_{t+176} \oplus z_{t+192} = 0$$

Simultaneity, the 8-to-1 non-linear filter function $v(\bullet)$ can be simplified as follows since the first two input bits are set to ones.

$$\begin{aligned} s_t &= v(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128}) \\ &= v(1,1, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128}) \\ &\underline{\Lambda} v'(\bullet) \end{aligned}$$

The best affine approximation of the nonlinear filter $v(\bullet)$ has bias $\varepsilon = 2^{-6}$ as claimed in sec.5.3 by the designers [3]. However, the best affine approximation of the nonlinear filter $v'(\bullet)$ has bias $\varepsilon = 2^{-2.42}$. Using the Pilling-up Lemma [10], the bias of the linear distinguisher is estimated as

$$2^{10} \times (2^{-2.42})^{11} = 2^{-16.62}$$

So, if the keystream sequence is generated by RAKAPOSHI with a weak (key, IV) pair, the linear distinguisher *D* holds with the probability bias $2^{-16.62}$. As a result, to distinguish the RAKAPOSHI with a weak (key, IV) pair, it requires $\left(1/2^{-16.62}\right)^2 = 2^{33.24}$ keystream bits. As claimed in sec.5.3 by the designers, RAKAPOSHI stream cipher is secure against distinguishing attacks. However, for weak (key, IV) pairs of RAKAPOSHI, it is vulnerable to linear distinguishing attacks.

A weak Key-IV can be distinguished means that the register A initial state of the weak Key-IV is the all ones state. Consequently, the next task is to recover the register B initial state. Algebraic attacks against stream ciphers were originally proposed in 2003 by Courtois and Meier [11]. The attack is a powerful cryptanalytic technique against some LFSR-based stream ciphers.

For the RAKAPOSHI stream cipher with a weak (key, IV) pair, the keystream output z_t is computed by means of a Boolean function over $GF(2^7)$ given by

$$\begin{aligned} z_t &= a_t \oplus b_t \oplus s_t = a_t \oplus b_t \oplus v(a_{t+67}, a_{t+127}, b_{t+23}, b_{t+53}, b_{t+77}, b_{t+81}, b_{t+103}, b_{t+128}) \\ &= 1 \oplus b_t \oplus v'(\bullet) \end{aligned}$$

This function has degree 5 and algebraic immunity 3 (which is the same of the function $v'(\bullet)$).

For each weak (key, IV) pair of RAKAPOSHI, the internal states of the register A are all ones after the initialization process. Here we use the XL method [12] to estimate the time complexity of solving the above algebraic equation system, and obtain that its time complexity is about

$$\left(\sum_{i=0}^{3} \binom{192}{i} \right)^3 \approx 2^{60.51}$$

The algebraic attack requires approximately $2^{20.17}$ keystream bits. The time complexity of this attack is much smaller than the time complexity of a brute force attack. Therefore for weak (key, IV) pairs of RAKAPOSHI, it is weak against algebraic attack.

4 Related Key Chosen IV Attack on RAKAPOSHI

In this section, we will propose a related key chosen IV attack on RAKAPOSHI stream cipher. The attack on RAKAPOSHI is done by assuming four conditions as shown in Table 2. The relation of (K, IV) and

(K', IV') is as follows.

$$\begin{aligned} K = (k_0, \cdots, k_{127}) &\Rightarrow K' = (k_1, \cdots, k_{127}, k_0 \oplus \varepsilon) \\ IV = (iv_0, \cdots, iv_{191}) &\Rightarrow IV' = (iv_1, \cdots, iv_{191}, 0) \end{aligned}.$$

Where $\varepsilon \in \{0,1\}$, is a binary constant.

Table 2. Conditions used in the attack on RAKAPOSHI

$K = (k_0, \cdots, k_{127}), IV = (iv_0, \cdots, iv_{191})$			$K' = (k_1, \cdots, k_{127}, k_0 \oplus d), IV' = (iv_1, \cdots, iv_{191}, 0)$			Condition
t	a_{t+127}	b_{t+191}	t	a'_{t+127}	b'_{t+191}	
0	k_{127}	iv_{191}				
1	$g(A^0)$	$f(S^0) \oplus s_0$	0	$k_0 \oplus \varepsilon$	0	$g(A^0) = k_0 \oplus \varepsilon$ $f(S^0) \oplus s_0 = 0$
2	$g(A^1)$	$f(S^1) \oplus s_1$	1	$g(A'^0)$	$f(S'^0) \oplus s'_0$	
⋮	⋮	⋮	⋮	⋮	⋮	
320	$g(A^{319})$	$f(S^{319}) \oplus s_{319}$	319	$g(A'^{318})$	$f(S'^{318}) \oplus s'_{318}$	
321	$g(A^{320}) \oplus s_{320}$	$f(S^{320})$	320	$g(A'^{319})$	$f(S'^{319}) \oplus s'_{319}$	$s_{320} = 0$ $(s_{320} = s'_{319})$
322	$g(A^{321}) \oplus s_{321}$	$f(S^{321})$	321	$g(A'^{320}) \oplus s'_{320}$	$f(S'^{320})$	
⋮	⋮	⋮	⋮	⋮	⋮	
448	$g(A^{447}) \oplus s_{447}$	$f(S^{447})$	447	$g(A'^{446}) \oplus s'_{446}$	$f(S'^{446})$	
449	$g(A^{448})$	$f(S^{448})$	448	$g(A'^{447}) \oplus s'_{447}$	$f(S'^{447})$	$s_{448} = 0$ $(s_{448} = s'_{447})$
⋮	⋮	⋮	⋮	⋮	⋮	

Let Z and Z' be keystream sequences generated from (K, IV) and (K', IV'). According to table 2, we can get Property 1 as follows.

Property 1. The system $S^1 = S'^0$ ***holds when the following two conditions are simultaneously satisfied.***

a) $g(A^0) = k_0 \oplus \varepsilon$;

b) $f(S^0) \oplus s_0 = 0$.

Recall the initialization of RAKAPOSHI stream cipher. It consists of two stages, which differ from each other in the feedback of s_t. According to table 2 and Property 1, we can get Property 2 as follows.

Property 2. If the system $S^1 = S'^0$ ***holds, the system*** $S^{i+1} = S'^i$ ***holds for*** $i \geq 1$ ***when the following two conditions are simultaneously satisfied.***

c) $s_{320} = 0$;

d) $s_{448} = 0$.

It is easy to see that if the system $S^{449} = S'^{448}$ holds, there exists a clear relation between Z and Z', i.e., $z_{i+1} = z'_i$ for $i \geq 0$.

Let $Z[m] = (z_1, \cdots, z_m)$ be the m-bit keystream sequence starting from the second keysteam bit generated by (K, IV) and $Z'[m] = (z'_0, \cdots, z'_{m-1})$ be m-bit keystream sequence starting from the first keysteam bit generated

by related (K', IV') pair. We define the event $Z[m] = Z'[m]$ (i.e., $z_{i+1} = z'_i$ for $0 \le i \le m-1$) by Ω and its complement by Ω^c. We denote the event that the conditions a) and b) are simultaneously satisfied by L and its complement by L^c. Since the conditions c) and d) are obtained in different clocks, and then are assumed to be independent to facilitate calculation of probability. Hence, when the event L occurs, the event Ω occurs with probability

$$\Pr(\Omega \mid L) = \Pr(c \bigcap d) = \Pr(c) \cdot \Pr(d) = 2^{-2}.$$

However, when the event L^c occurs, after 447 clocks of iteration, the differentials of internal states will be fairly even and unpredictable, the event Ω occurs with probability 2^{-m}. That is,

$$\Pr(\Omega \mid L^c) = 2^{-m}$$

In order to recover the some key bits effectively, we should fix some IV bits. Here, we choose $iv_{107} = iv_{120} = iv_{134} = iv_{136} = iv_{155} = 0$ and $iv_{158} = 1$. Thus, the equation $f(S^0) \oplus s_0 = 0$ can be simplified as follows.

$$\begin{aligned} f(S^0) \oplus s_0 &= iv_0 \oplus iv_{14} \oplus iv_{37} \oplus iv_{41} \oplus iv_{49} \oplus iv_{51} \oplus iv_{93} \oplus c_0 \cdot iv_{158} \oplus iv_{176} \oplus s_0 \\ &= iv_0 \oplus iv_{14} \oplus iv_{37} \oplus iv_{41} \oplus iv_{49} \oplus iv_{51} \oplus iv_{93} \oplus k_{41} \cdot iv_{158} \oplus iv_{176} \oplus s_0 \\ &= iv_0 \oplus iv_{14} \oplus iv_{37} \oplus iv_{41} \oplus iv_{49} \oplus iv_{51} \oplus iv_{93} \oplus k_{41} \oplus iv_{176} \oplus s_0 \\ &= 0 \end{aligned} \quad (1)$$

Where $s_0 = v(a_{67}, a_{127}, b_{23}, b_{53}, b_{77}, b_{81}, b_{103}, b_{128}) = v(k_{67}, k_{127}, iv_{23}, iv_{53}, iv_{77}, iv_{81}, iv_{103}, iv_{128})$.

Recall the non-linear filter function $v(\bullet)$. Three functions can be obtained as follows.

When $iv_{23} = iv_{53} = iv_{77} = iv_{81} = iv_{128} = 0$ and $iv_{103} = 1$,

$$s_0 = v(k_{67}, k_{127}, iv_{23}, iv_{53}, iv_{77}, iv_{81}, iv_{103}, iv_{128}) = 1 \quad (2)$$

When $iv_{23} = iv_{53} = iv_{77} = iv_{103} = iv_{128} = 0$ and $iv_{81} = 1$,

$$s_0 = v(k_{67}, k_{127}, iv_{23}, iv_{53}, iv_{77}, iv_{81}, iv_{103}, iv_{128}) = k_{67} \oplus 1 \quad (3)$$

When $iv_{77} = iv_{81} = iv_{103} = iv_{128} = 0$ and $iv_{23} = iv_{53} = 1$,

$$s_0 = v(k_{67}, k_{127}, iv_{23}, iv_{53}, iv_{77}, iv_{81}, iv_{103}, iv_{128}) = k_{127} \quad (4)$$

Now, we will introduce a method to recover one key bit with 2^h chosen IVs, named Algorithm 1.

Algorithm 1

1. Under fixed and unknown K and K', choose 2^h IVs where $iv_{107} = iv_{120} = iv_{134} = iv_{136} = iv_{155} = 0$, $iv_{158} = 1$, $(iv_0, \cdots, iv_{h-1})$ are all 2^h-bit values and the remaining bits are fixed to e $(\in \{0,1\})$.
2. Until no more chosen IV remain, repeat the followings.
   - Generate the keystream sequences $Z[m] = (z_1, \cdots, z_m)$ using (K, IV);
   - Generate the keystream sequences $Z'[m] = (z'_0, \cdots, z'_{m-1})$ using (K', IV');
   - Check $Z[m] = Z'[m]$. If the two keystream sequences pass the check, output $g(A^0) = k'_0 \oplus \varepsilon$. Otherwise try other chosen IVs.
3. If no chosen IVs passes the check, then output $g(A^0) = k_0 \oplus \varepsilon \oplus 1$.

With a decision rule of Algorithm 1, two types of errors denoted as type I and type II exist, see Figure.2. Let α and β denote the probabilities of type I and II errors, respectively. Thus, $\Pr(\alpha) = (1-2^{-2})^{2^h}$ and $\Pr(\beta) = 1-(1-2^{-m})^{2^h}$. Hence, we can obtain the equation $g(A^0) = k'_0 \oplus \varepsilon$ with a success probability

of $1-\Pr(\alpha)$ if the equation $g(A^0)=k_0'\oplus\varepsilon$ holds, and obtain the equation $g(A^0)=k_0\oplus\varepsilon\oplus 1$ with a success probability of $1-\Pr(\beta)$ if the equation $g(A^0)=k_0'\oplus\varepsilon$ does not hold. Thus, the success probability of this algorithm is bounded below by $1-\Pr(\alpha)-\Pr(\beta)$. In our paper, we use $p_0=1-\Pr(\alpha)-\Pr(\beta)$ as the success probability of our attack. The actual value of success probability is in fact more, making our attack marginally stronger.

		Decision	
		Accept L	Accept L^c
Truth	$L: g(A^0)=k_0'\oplus\varepsilon$	Correct retention	Type I error
	$L^c: g(A^0)=k_0'\oplus\varepsilon\oplus 1$	Type II error	Correct rejection

Figure 2. Two types of errors in the decision rule of Algorithm 1

The number of chosen *IV*s used In Algorithm 1 is 2^h. For each chosen *IV*, the algorithm needs to execute the encryption process of RAKAPOSHI two times. Thus, the computational complexity of Algorithm 1 is $2^h\cdot 2=2^{h+1}$. It requires $2^h\cdot(m+1+m)=2^{h+1}(2m+1)$ keystream bits.

In our related key attacks on RAKAPOSHI, we use $2m+1$ related keys, which are shows as follows.

For $0\le j\le m-1$,

$$K_j^0=(k_0,\cdots,k_{127})\Rightarrow K_{j+1}^0=(k_1,\cdots,k_{127},k_0), K_{j+1}^1=(k_1,\cdots,k_{127},k_0\oplus 1)$$
$$IV_j=(iv_0,\cdots,iv_{191})\Rightarrow IV_{j+1}=(iv_1,\cdots,iv_{191},0)$$

Note that $K_0^0=K$.

Now, we will introduce a method to recover more key bits using these related keys, named Algorithm 2.

Algorithm 2

For $0\le j\le m-1$, do the followings.

1. Under the related keys K_j^0 and K_{j+1}^0, run Algorithm 1. If Algorithm 1 outputs $g(A^0)=k_0'$, goto step 2. Otherwise, goto step 3.

2. Under the related keys K_j^0 and K_{j+1}^0, do the followings.

- Set $iv_{23}=iv_{53}=iv_{77}=iv_{81}=iv_{128}=0$ and $iv_{103}=1$, and then run Algorithm 1. Recover the key bit k_{41} of K_j^0 using the equation (1) and (2).
- Set $iv_{23}=iv_{53}=iv_{77}=iv_{103}=iv_{128}=0$ and $iv_{81}=1$, and then run Algorithm 1. Recover the key bit k_{67} of K_j^0 using the equation (1) and (3).
- Set $iv_{77}=iv_{81}=iv_{103}=iv_{128}=0$ and $iv_{23}=iv_{53}=1$, and then run Algorithm 1. Recover the key bit k_{127} of K_j^0 using the equation (1) and (4).

3. Under the related keys K_j^0 and K_{j+1}^1, do the followings.

- Set $iv_{23}=iv_{53}=iv_{77}=iv_{81}=iv_{128}=0$ and $iv_{103}=1$, and then run Algorithm 1. Recover the key bit k_{41} of K_j^0 using the equation (1) and (2).
- Set $iv_{23}=iv_{53}=iv_{77}=iv_{103}=iv_{128}=0$ and $iv_{81}=1$, and then run Algorithm 1. Recover the key bit k_{67} of K_j^0

using the equation (1) and (3).

- Set $iv_{77} = iv_{81} = iv_{103} = iv_{128} = 0$ and $iv_{23} = iv_{53} = 1$, and then run Algorithm 1. Recover the key bit k_{127} of K_j^0 using the equation (1) and (4).

Using the Algorithm 2, we can recover the following key bits.

$$k_{41 \bmod 128}, \cdots, k_{41+m-1 \bmod 128}$$
$$k_{67 \bmod 128}, \cdots, k_{67+m-1 \bmod 128}$$
$$k_{127 \bmod 128}, \cdots, k_{127+m-1 \bmod 128}$$

Furthermore, we also can obtain m non-linear functions with some key bits. In Algorithm 2, for $0 \le j \le m-1$, we should run Algorithm 1 four times under related keys. Hence, Algorithm 2 requires $2^h \cdot 4 = 2^{h+2}$ chosen *IV*s and $2^{h+1}(2m+1) \cdot 4 = 2^{h+3}(2m+1)$ keystream bits. The computational complexity of Algorithm 2 is $2^{h+1} \cdot 4 = 2^{h+3}$. The success probability of Algorithm 2 is bounded below by p_0^{3m}, since the success probability of Algorithm 1 is bounded below by p_0.

In order to get a high success probability for recovering the 128-bit secret key, m and h should be properly selected. When we set $m = 23$, then the attacker can recover 69 key bits $(k_0, \cdots, k_{21}, k_{41}, \cdots, k_{63}, k_{67}, \cdots, k_{89}, k_{127})$. We also can obtain 23 non-linear functions with 22 key bits $(k_{22}, \cdots, k_{40}, k_{64}, k_{65}, k_{66})$. In fact, we can simplify these non-linear functions using the recovered 69 key bits. Thus, the 22 key bits $(k_{22}, \cdots, k_{40}, k_{64}, k_{65}, k_{66})$ can be recovered easily, with a computational complexity of 2^{22} at most. Finally, the remaining 37 key bits $(k_{90}, \cdots, k_{126})$ should be exhaustive searched. Thus, considering Algorithm 2, the computational complexity of our related key chosen IV attack on RAKAPOSHI is about $2^{h+3} + 2^{22} + 2^{37}$, requiring 2^{h+2} chosen *IV*s and $2^{h+3}(2m+1)$ keystream bits. Table 3 shows the complexities of our attack on RAKAPOSHI for parameters *h*. Note that in our related key attacks on RAKAPOSHI, we use $2m+1 = 47$ related keys in Algorithm 2.

Table 3. Our results on RAKAPOSHI for parameters *h*

h	Computational complexity	Chosen *IV*s	Keystream bits required	Success probability
4	2^{37}	2^6	$2^{12.555}$	0.514
5	2^{37}	2^7	$2^{13.555}$	0.993
6	2^{37}	2^8	$2^{14.555}$	0.999

Here, we choose $h = 6$. Thus, we can recover the 128-bit secret key of RAKAPOSHI with a computational complexity of 2^{37}, requiring 47 related keys, 2^8 chosen *IV*s and $2^{14.555}$ keystream bits. The success probability of our attack on RAKAPOSHI is 0.999, which is quite close to 1.

We also validate our result by simulating Algorithm 2 and solving the 23 non-linear functions. The result shows that we can recover the 91 (=69+22) key bits within 10.8 minutes on average. By simulating RAKAPOSHI stream cipher, the remaining 37 can be recovered within 23.04 hours on average. The simulation was implemented on AMD Athlon(tm) 64 X2 Dual Core Processor 4400+, CPU 2.31GHz, 768 Gb RAM, OS Windows XP Pro SP3. These experimental results corroborate our assertion.

5 Conclusions

RAKAPOSHI is a hardware-oriented stream cipher designed by Carlos Cid et al. in 2009. The RAKAPOSHI stream cipher offers 128-bit security. Until now, no attack on the cipher has been published. In this paper, we point out some weaknesses in the cipher. Firstly, the result shows that there are 2^{192} weak (key, IV) pairs in RAKAPOSHI stream cipher. Secondly, for weak (key, IV) pairs of RAKAPOSHI, they are vulnerable to linear distinguishing attack and algebraic attack. Finally, we proposed a real time related key chosen IV attack on RAKAPOSHI. The time complexity of our related key chosen IV attack on RAKAPOSHI is about 2^{34}, requiring 39 related keys on average. The results show that there exist some weaknesses in the cipher.

We hope our results can be helpful in evaluating the security of RAKAPOSHI stream ciphers against related key attacks, and we look forward to further work in evaluating RAKAPOSHI stream ciphers against other kinds of cryptanalytic attacks.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (No. 61202491, 61272041, 61272488).

References

[1] NESSIE. New European Schemes for Signatures, Integrity, and Encryption. Available at http://www.cryptonessie.org, Accessed August 18, 2003

[2] ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available at http://www.ecrypt.eu.org/stream/, Accessed September 29, 2005

[3] Carlos Cid, Shinsaku Kiyomoto, and Jun Kurihara. The RAKAPOSHI Stream Cipher. Information and Communications Security, LNCS 5927[C]. 2009. 32-46.

[4] T. Beth and F.C. Piper. The stop-and-go-generator. In Proceedings of EUROCRYPT 1984, number 209 in LNCS, pages 88–92. Springer-Verlag, 1985.

[5] L.R. Simpson, E. Dawson, J. Golic, and W. Millan. LILI Keystream Generator. In Selected Areas in Cryptography 2000, number 2012 in LNCS, pages 248–261. Springer-Verlag, 2000.

[6] D. Horan and R. Guinee. A Novel Keystream Generator using Pseudo Random Binary Sequences for Cryptographic Applications. In Irish Signals and Systems Conference 2006, pages 451–456. IEEE, 2006.

[7] S. Kiyomoto, T. Tanaka, and K. Sakurai. K2: A Stream Cipher Algorithm Using Dynamic Feedback Control. In Proceedings of SECRYPT 2007, pages 204–213, 2008.

[8] T. Isobe, T. Ohigashi, and M. Morii, Slide cryptanalysis of lightweight stream cipher rakaposhi, in Advances in Information and Computer Security, G. Hanaoka and T. Yamauchi, eds., vol. 7631 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 138-155.

[9] Mohammad Ali Orumiehchiha, Josef Pieprzyk, Elham Shakour and Ron Steinfeld. Security Evaluation of Rakaposhi Stream Cipher. Cryptology ePrint Archive Report 2012/656, http://eprint.iacr.org/.

[10] M. Matsui. Linear Cryptanalysis Method for DES Cipher. In Proceedings of EUROCRYPT 1993, number 765 in LNCS, pages 386–397. Springer-Verlag, 1993.

[11] N. Courtois and W. Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In E. Biham, editor, Advances in Cryptology – EUROCRYPT 2003, volume 2656 of LNCS, pages 345–359. Springer–Verlag, 2003.

[12] C. Diem, The XL-Algorithm and a Conjecture from Commutative Algebra, In Pil Joong Lee, editor, Advances in Cryptology-ASIACRYPT2004, LNCS 3329, pp.323-337, 2004.