社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2007－ＣＳＥＣ－38 (31)
2007／7／20

MICKEYの鍵スケジューリングアルゴリズムの解析

藤川　香顕[†]　大東　俊博[†]　桑門　秀典[††]　森井　昌克[††]

† 神戸大学大学院自然科学研究科
〒657-8501 神戸市灘区六甲台町 1-1
†† 神戸大学大学院工学研究科
〒657-8501 神戸市灘区六甲台町 1-1
E-mail: †{y.fujikwa,ohigashi}@stu.kobe-u.ac.jp, ††{kuwakado,mmorii}@kobe-u.ac.jp

あらまし　MICKEY 及び MICKEY-128 は eSTREAM に提案されているストリーム暗号であり，それぞれ 80 ビット及び 128 ビットの秘密鍵を用いる．eSTREAM は Phase 3 まで評価が進んでいるが，MICKEY 及び MICKEY-128 に対する致命的な攻撃法はまだ提案されておらず，現在も安全性評価が続けられている．本稿では，MICKEY 及び MICKEY-128 の鍵スケジューリングアルゴリズム (KSA) に関する構造，特に KSA の一方向性について考察する．また，全数探索より大幅に少ない計算量で初期状態から秘密鍵を導出可能な meet-in-the-middle attack に基づいた手法を提案する．提案手法を用いることで MICKEY では 2^{44}，MICKEY-128 では 2^{68} の計算量で秘密鍵を復元できることを示す．

キーワード　ストリーム暗号，eSTREAM，MICKEY，鍵スケジューリングアルゴリズム，一方向性

Analysis on the Key-Scheduling Algorithm of MICKEY

Yoshiaki FUJIKAWA[†], Toshihiro OHIGASHI[†], Hidenori KUWAKADO[††], and Masakatu MORII[††]

† Graduate School of Science and Technology, Kobe University,
1–1 Rokkodai, Nada-ku, Kobe-shi, 657–8501 Japan.
†† Graduate School of Engineering, Kobe University,
1–1 Rokkodai, Nada-ku, Kobe-shi, 657–8501 Japan.
E-mail: †{y.fujikwa,ohigashi}@stu.kobe-u.ac.jp, ††{kuwakado,mmorii}@kobe-u.ac.jp

Abstract MICKEY and MICKEY-128, which are stream ciphers submitted to the ECRYPT Stream Cipher Project (eSTREAM), use a 80-bit secret key and a 128-bit one, respectively. MICKEY and MICKEY-128 are candidates on Phase 3 of eSTREAM. However, no critical attack against MICKEY and MICKEY-128 have been proposed, and the security evaluation against them is continued now. In this paper, we study the one-wayness of the key-scheduling algorithm of MICKEY and MICKEY-128. We propose a method based on the meet-in-the-middle attack for recovering a secret key from a given initial state. The computational complexity of proposed method is less than the computational complexity of the brute force attack. We show that the computational complexity of recovering a secret key from a given initial state is about 2^{44} in MICKEY and about 2^{68} in MICKEY-128.

Key words stream cipher, eSTREAM, MICKEY, key-scheduling algorithm, one-wayness

1. Introduction

Stream ciphers [1], [2] are one of the classifications of the symmetric-key encryption. In stream ciphers, encryption is performed by XORing of a plaintext and a pseudo-random sequence (called a keystream), which is generated by a secret key and an initialization vector (IV). Decryption is performed by XORing of a ciphertext and the same pseudo-random sequence. The body of a stream cipher is generating the keystream from the secret key and the IV. It consists of a key-scheduling algorithm (KSA) and a pseudo-random generation algorithm (PRGA). The KSA initializes the internal state with the secret key and the IV. The PRGA generates the keystream from the initial state, which is an internal state when the KSA finished.

The goal of key recovery attacks against stream ciphers is recovering the secret key from the keystream. Since stream ciphers consist of the KSA and the PRGA, the key recovery attack is divided into two steps: an internal state reconstruction method and a key reconstruction method. The internal state reconstruction method recovers the initial state from a given keystream. The key reconstruction method recovers the secret key from a given initial state. The key reconstruction method was not been discussed while internal state reconstruction methods against stream ciphers (e.g., SNOW1.0 [3], SOSEMANUK [4], and Polar Bear [5], etc.) have been proposed [6]~[9]. If the internal state reconstruction method succeeded against one keystream, the internal state can be obtained and the ciphertext generated by the internal state can be deciphered. However, this decipherment is effective for only the IV, because the initial state is generated by the secret key and the IV. Thus, it is necessary to obtain the secret key for generating the keystream of other IVs. If the secret key can be easily recovered from the initial state, it any ciphertext can be deciphered when the internal state reconstruction method succeeded. Therefore, to prevent the secret key from being recovered from the initial state, the KSA should have one-wayness.

We have been analyzed the one-wayness of the KSAs for Dragon [10], Grain-128 [11](Grain-1.0 [12]), HC-256 [13] (HC-128 [14]), Trivium [15], LEX [16] and SOSEMANUK in [18]. These stream ciphers are candidates on Phase 3 of the ECRYPT Stream Cipher Project (eSTREAM) [17] which is project that selects next generation's stream cipher. As a result of analysis, we showed that the KSAs for Dragon, Grain-128(Grain-1.0), HC-256(HC-128) and Trivium are not one-way and the KSAs for LEX and SOSEMANUK are one-way.

In this paper, we analyze the one-wayness of the KSAs for MICKEY [19] and MICKEY-128 [20] which remain as candidate on Phase 3 of eSTREAM. And we propose a method based on the meet-in-the-middle attack [21] for recovering the secret key from a given initial state. The computational complexity of recovering the secret key from a given initial state is about 2^{44} in MICKEY and about 2^{68} in MICKEY-128.

2. MICKEY

MICKEY and MICKEY-128 have been proposed by S. Babbage and M. Dodd. In MICKEY, the 80-bit secret key and the variable length IV are used, where the IV length is between 0 and 80 bits. In MICKEY-128, the 128-bit secret key and the variable length IV are used, where the IV length is between 0 and 128 bits. There is no major difference in the KSAs of MICKEY and MICKEY-128. Therefore, we discuss the KSA of MICKEY in below sections.

2.1 Operations of the KSA

MICKEY consists of two registers R and S. Each register is 100 stages long, each stage containing 1 bit. Let $r_{i,t}$ and $s_{i,t}$ ($0 \leqq i \leqq 99$) be the bits in the registers, respectively, where t is time of clocking registers. The registers are clocked with three operations: $CLOCK\_KG(R, S, MIXING, IB)$, $CLOCK\_R(R, IB\_R, CB\_R)$ and $CLOCK\_S(S, IB\_S, CB\_S)$.

The operation $CLOCK\_KG(R, S, MIXING, IB)$, which is an operation for clocking the overall generator, is defined as the following equations:

$$CB\_R = s_{34,t} \oplus r_{67,t}, \tag{1}$$

$$CB\_S = s_{67,t} \oplus r_{33,t}, \tag{2}$$

$$IB\_R = \begin{cases} IB \oplus s_{50,t} & (MIXING = TRUE), \\ IB & (MIXING = FALSE), \end{cases} \tag{3}$$

$$IB\_S = IB, \tag{4}$$

$$R = CLOCK\_R(R, IB\_R, CB\_R), \tag{5}$$

$$S = CLOCK\_S(S, IB\_S, CB\_S). \tag{6}$$

The operation $CLOCK\_R(R, IB\_R, CB\_R)$, which is an operation for clocking register R, is defined as the following equations:

$$FB\_R = r_{99,t} \oplus IB\_R, \tag{7}$$

$$r'_{i,t+1} = \begin{cases} 0 & (i = 0), \\ r_{i-1,t} & (i \neq 0), \end{cases} \tag{8}$$

$$r''_{i,t+1} = \begin{cases} r'_{i,t+1} \oplus FB\_R & (i \in RTAPS), \\ r'_{i,t+1} & (i \notin RTAPS), \end{cases} \tag{9}$$

$$r_{i,t+1} = \begin{cases} r''_{i,t+1} \oplus r_{i,t} & (CB\_R = 1), \\ r''_{i,t+1} & (CB\_R = 0), \end{cases} \tag{10}$$

where $RTAPS$ is a set of feedback tap positions for R. The detail of $RTAPS$ is shown in Appendix 1. Figure 1, 2 show clocking the register R with $CB\_R = 0$ and $CB\_R = 1$.

The operation $CLOCK\_S(S, IB\_S, CB\_S)$, which is an operation for clocking register S, is defined as the following equations:

$$FB\_S = s_{99,t} \oplus IB\_S, \tag{11}$$

$$s'_{i,t+1} = \begin{cases} 0 & (i = 0), \\ s_{i-1,t} \oplus ((s_{i,t} \oplus COMP0_i) \cdot (s_{i+1,t} \oplus COMP1_i)) & (i \neq 0, 99), \\ s_{98,t} & (i = 99), \end{cases} \tag{12}$$

$$s_{i,t+1} = \begin{cases} s'_{i,t+1} \oplus (FB0_i \cdot FB\_S) & (CB\_S = 0), \\ s'_{i,t+1} \oplus (FB1_i \cdot FB\_S) & (CB\_S = 1), \end{cases} \tag{13}$$

where $x \cdot y$ is the AND operation of x and y, $COMP0_i$ and $COMP1_i$ are input masks and $FB0_i$ and $FB1_i$ are feedback

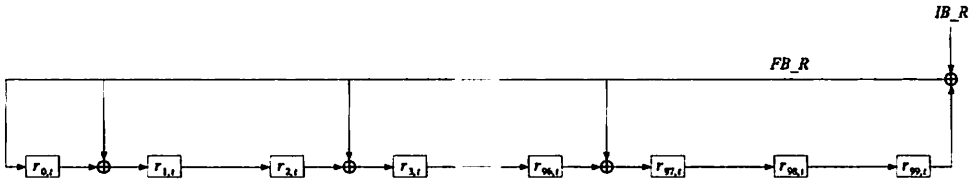

Figure 1 Clocking the R register with $CB\_R = 0$.

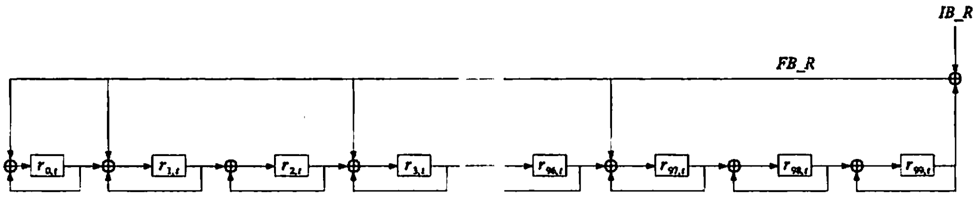

Figure 2 Clocking the R register with $CB\_R = 1$.

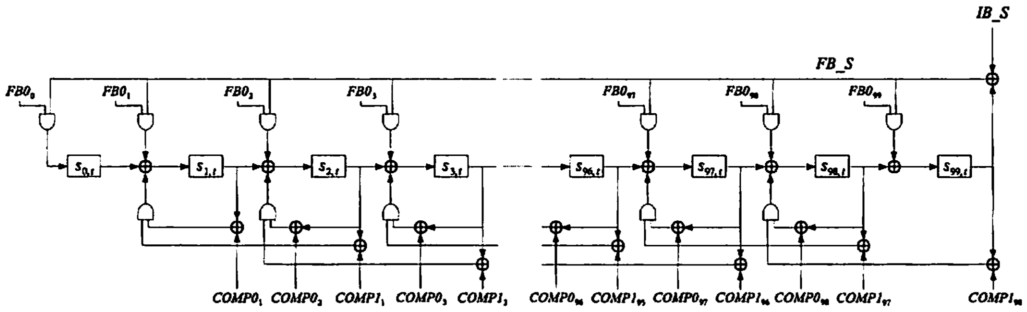

Figure 3 Clocking the S register with $CB\_S = 0$.

masks. The details of these masks are shown in Appendix 2. Figure 3 shows clocking the register S with $CB\_S = 0$. The figure of clocking the register S with $CB\_S = 1$ is obtained by substituting $FB1_i$ for $FB0_i$ in Fig. 3.

2.2 Generating the initial state

The initial state is generated with the 80-bit secret key K_j $(0 \leqq j \leqq 79)$ and the variable length initialization vector IV_k $(0 \leqq k \leqq IVLENGTH - 1)$ as the following process. Firstly, the registers R and S are initialized with all zero:

$$r_{i,0} = s_{i,0} = 0. \qquad (14)$$

Secondly, the registers R and S are clocked by loading the bitwise IV_k (this step is called Loading IV Step):

For $0 \leqq k \leqq IVLENGTH - 1$,

$$\{R, S\} = CLOCK\_KG(R, S, TRUE, IV_k). \qquad (15)$$

On end, the registers R and S are clocked by loading the bitwise K_j (this step is called Loading Key Step):

For $0 \leqq j \leqq 79$,

$$\{R, S\} = CLOCK\_KG(R, S, TRUE, K_j). \qquad (16)$$

Finally, the registers R and S are clocked by loading zero for 100 times (this step is called Preclock):

For $0 \leqq l \leqq 99$,

$$\{R, S\} = CLOCK\_KG(R, S, TRUE, 0). \qquad (17)$$

The initial state is $r_{i,IVLENGTH+180}$ and $s_{i,IVLENGTH+180}$, which are the bits in registers when Preclock finished.

3. Analyzing the KSA for MICKEY

The KSA for MICKEY is divided into Loading IV Step, Loading Key Step and Preclock. Since the registers R and S are initialized with all zero and the IV is the public value, the internal states $r_{i,IVLENGTH}$ and $s_{i,IVLENGTH}$ are obtained easily. Our purpose of analyzing one-wayness is to analyze the possibility of recovering the secret key from a given initial state. Therefore, we analyze Loading Key Step and Preclock. Notice that if $r_{i,IVLENGTH}$ and $s_{i,IVLENGTH}$ correspond to the internal state which is obtained by tracing Preclock and Loading Key Step, the secret key is recovered.

Let $\hat{r}_{i,t}$ and $\hat{s}_{i,t}$ be the internal state which is obtained by tracing Preclock or Loading Key Step in the following section.

3.1 One-wayness of Preclock

In Preclock, IB is constantly zero from Eq. (17). Substituting $IB = 0$ into Eqs. (3) and (4) always gives the following equations:

$$IB\_R = \hat{s}_{50,t-1}, \tag{18}$$

$$IB\_S = 0. \tag{19}$$

Since $\hat{s}_{50,t-1}$ is an unknown value, it is necessary to previously compute $\hat{s}_{i,t-1}$ from $\hat{s}_{i,t}$ and, next, compute $\hat{r}_{i,t-1}$ from $\hat{r}_{i,t}$.

We compute $\hat{s}_{i,t-1}$ from $\hat{s}_{i,t}$. Firstly, we notice that $\hat{s}_{i,t-1}$ is independent of CB_S. From Eqs. (12), (13) and $FB0_0 = FB1_0 = 1$, FB_S is equal to $\hat{s}_{0,t}$. Substituting Eq. (19) and $FB\_S = \hat{s}_{0,t}$ into Eq. (11) gives

$$\hat{s}_{99,t-1} = \hat{s}_{0,t}.$$

Next, we compute $\hat{s}_{i,t-1}$ which is dependent on CB_S. From Eqs. (12) and (13), the formula of $\hat{s}_{98,t-1}$ is obtained as follows:

$$\hat{s}_{98,t-1} = \begin{cases} \hat{s}_{99,t} \oplus (FB0_{99} \cdot \hat{s}_{0,t}) & (CB\_S = 0), \\ \hat{s}_{99,t} \oplus (FB1_{99} \cdot \hat{s}_{0,t}) & (CB\_S = 1). \end{cases}$$

Using the above equation, $\hat{s}_{98,t-1}$ can be computed. From Eqs. (12) and (13), the formula of $\hat{s}_{97-i,t-1}$ for $0 \leqq i \leqq 97$ is obtained as follows:

$$\begin{aligned} \hat{s}_{97-i,t-1} &= \hat{s}_{98-i,t} \oplus x \oplus \\ &((\hat{s}_{98-i,t-1} \oplus COMP0_{98-i}) \cdot (\hat{s}_{99-i,t-1} \oplus COMP1_{98-i})), \end{aligned}$$

where x is a value defined as the following equation:

$$x = \begin{cases} FB0_{98-i} \cdot FB\_S & (CB\_S = 0), \\ FB1_{98-i} \cdot FB\_S & (CB\_S = 1). \end{cases}$$

Using the above equation, $\hat{s}_{i,t-1} (0 \leqq i \leqq 97)$ can be computed.

Using $\hat{s}_{50,t-1}$, we compute $\hat{r}_{i,t-1}$ from $\hat{r}_{i,t}$. Firstly, we discuss the case of $CB\_R = 0$. From Eqs. (8)-(10), FB_R is equal to $\hat{r}_{0,t}$. Substituting Eq. (18) and $FB\_R = \hat{r}_{0,t}$ into Eq. (7) gives

$$\hat{r}_{99,t-1} = \hat{r}_{0,t} \oplus \hat{s}_{50,t-1}.$$

From Eqs. (8)-(10), the formula of $\hat{r}_{i,t-1}$ for $0 \leqq i \leqq 98$ is obtained as follows:

$$\hat{r}_{i,t-1} = \begin{cases} \hat{r}_{i+1,t} \oplus \hat{r}_{0,t} & (i \in RTAPS), \\ \hat{r}_{i+1,t} & (i \notin RTAPS). \end{cases}$$

Using the above equation, $\hat{r}_{i,t-1}$ for $0 \leqq i \leqq 98$ can be computed. Next, we discuss the case of $CB\_R = 1$. From Eqs. (8)-(10), the formula of $\hat{r}_{i,t-1}$ for $0 \leqq i \leqq 99$ is obtained as follows:

$$\hat{r}_{i,t-1} = \begin{cases} \hat{r}_{0,t} \oplus FB\_R & (i = 0), \\ \hat{r}_{i,t} \oplus FB\_R \oplus \hat{r}_{i-1,t-1} & (i \neq 0, i \in RTAPS), \\ \hat{r}_{i,t} \oplus \hat{r}_{i-1,t-1} & (i \neq 0, i \notin RTAPS). \end{cases} \tag{20}$$

From Eq. (20), $\hat{r}_{99,t-1}$ is computed as follows:

$$\begin{aligned} \hat{r}_{99,t-1} &= \hat{r}_{99,t} \oplus \hat{r}_{98,t-1} = \hat{r}_{99,t} \oplus \hat{r}_{98,t} \oplus \hat{r}_{97,t-1}, \\ &= \hat{r}_{99,t} \oplus \hat{r}_{98,t} \oplus \hat{r}_{97,t} \oplus \hat{r}_{96,t-1} \oplus FB\_R, \\ &= \cdots = \bigoplus_{i=0}^{99} \hat{r}_{i,t} \oplus \bigoplus_{m=0}^{49} FB\_R = \bigoplus_{i=0}^{99} \hat{r}_{i,t}. \end{aligned}$$

Substituting the above equation and Eq. (18) into Eq. (7) gives

$$FB\_R = \bigoplus_{i=0}^{99} \hat{r}_{i,t} \oplus \hat{s}_{50,t-1}. \tag{21}$$

Substituting Eq. (21) into Eq. (20), $\hat{r}_{i,t-1} (0 \leqq i \leqq 98)$ can be computed.

Actually, since CB_R and CB_S are computed using Eq. (1) and Eq. (2), respectively, it is necessary to presume CB_R and CB_S in computing $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$. So, in each step of tracing Preclock, it is necessary to try by four patterns. Tracing Preclock is performed as follows:

Step A1 CB_R and CB_S are presumed as any one of four patterns and Step A2 is performed. If all patterns are performed, all candidates of $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$ are obtained.

Step A2 $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$ are computed with $\hat{r}_{i,t}$, $\hat{s}_{i,t}$ and presumed CB_R and CB_S.

Step A3 CB_R' and CB_S' are computed from $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$ using Eq. (1) and Eq. (2), respectively.

Step A4 $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$ computed in Step A2 are checked with 2 bits data: CB_R' and CB_S'. CB_R' and CB_S' are compared with presumed CB_R and CB_S. If corresponded, $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$ computed in Step A2 are considered as candidates of $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$. Return to Step A1.

The above four steps are performed from $t = IVLENGTH + 180$ to $t = IVLENGTH + 81$.

All computed $\hat{r}_{i,IVLENGTH+80}$ and $\hat{s}_{i,IVLENGTH+80}$ are candidates of $r_{i,IVLENGTH+80}$ and $s_{i,IVLENGTH+80}$ which are obtained when Loading Key Step finished. If checking in Step A4 is not effective, then the number of candidates is $4^{100} = 2^{200}$. However, CB_R' and CB_S' in each patterns are not necessarily corresponded with presumed CB_R and CB_S. Thus, the number of candidates does not come up to even 100. These are equivalent internal states which are generated the same initial state. We will statistically show the number of equivalent internal states in Section 4.

3.2 One-wayness of Loading Key Step

The method of tracing Loading Key Step is same as the method of tracing Preclock. However, since IB is the bit-wise secret key K_j, it is necessary to presume IB in computing $\hat{r}_{i,t-1}$ and $\hat{s}_{i,t-1}$. So, in each step of tracing Loading Key Step, it is necessary to try by eight patterns. As well as the consideration of tracing Preclock, CB_R' and CB_S' in each patterns are not necessarily corresponded with presumed CB_R and CB_S. However, since it is necessary to presume IB in each step of tracing Loading Key Step, the computational complexity of tracing Loading Key Step is $\alpha \cdot 2^{80}$, where α is the value that is approximated the number of equivalent internal states in tracing Preclock. Therefore, the computational complexity of tracing Loading Key Step nearly equals to the computational complexity of the brute force attack. Then, we will reduce the computational complexity of tracing Loading Key Step by using a method based on the meet-in-the-middle attack.

3.3 Proposed method based on the meet-in-the-middle attack

Since the IV is the public value, $r_{i,IVLENGTH}$ and $s_{i,IVLENGTH}$ are obtained from Eq. (15). Candidates of $\hat{r}_{i,IVLENGTH+80}$ and $\hat{s}_{i,IVLENGTH+80}$, which are the internal states when Loading Key Step finished, are obtaind from a given initial state by tracing Preclock. Thus, the meet-in-the-middle attack is applicable, because the starting internal states and the ending internal states in Loading Key Step are known.

Tracing Loading Key Step applied the meet-in-the-middle atack is performed as follows:

<u>Step B1</u> 2^{40} patterns of $\hat{r}_{i,IVLENGTH+40}$ and $\hat{s}_{i,IVLENGTH+40}$ are generated from $r_{i,IVLENGTH}$, $s_{i,IVLENGTH}$ and 2^{40} presume upper half keys $uK_j = K_j$ $(0 \leq j \leq 39)$.

<u>Step B2</u> $\hat{r}'_{i,IVLENGTH+40}$ and $\hat{s}'_{i,IVLENGTH+40}$ are computed from $\hat{r}_{i,IVLENGTH+80}$, $\hat{s}_{i,IVLENGTH+80}$ and a presume lower half key $lK_j = K_{j+40}$ $(0 \leq j \leq 39)$.

<u>Step B3</u> $\hat{r}'_{i,IVLENGTH+40}$ and $\hat{s}'_{i,IVLENGTH+40}$ are compared with each $\hat{r}_{i,IVLENGTH+40}$ and $\hat{s}_{i,IVLENGTH+40}$. If there is corresponding one, the secret key K is obteind from the concatenation of uK and lK. If there is corresponding nothing, Step B2 is done with other lK.

In Step B1, 2^{40} internal states are generated. Since the number of candidates of generated internal state are 2^{200}, the possibility of colliding in these internal states is low if internal states were randomly generated in the KSA of MICKEY.

In Step B1, 2^{40} computational complexity and 2^{40} memory are required. In Step B2 and Step B3, $\beta \cdot 2^{40}$ computational complexity is required, where β is the value that changes because of the number of equivalent internal states generated in tracing Preclock. We will show β in Section 4. Tehrefore, tracing Loading Key Step required $(\beta+1) \cdot 2^{40}$ computational complexity and 2^{40} memory.

4. Evaluation of a proposed method

In this section, we show computational complexity of tracing Preclock and Loading Key Step by a simulation. From a problem of execution time, we simulate cut-down versions of MICKEY and MICKEY-128 by using fixed IV. Cut-down versions of MICKEY and MICKEY-128 are MICKEY and MICKEY-128 to which the size of registers R and S is not changed and the length of secret key is shortened. We simulate against four patterns of the length of secret keys: 16-bit, 24-bit, 32-bit and 40-bit. Trial number of secret keys is 10^4, respectively. And using IV in a simulation is IV=0x9c53, which is randomly selected. Table 1 shows the result of a simulation.

Table 1 shows about the following four items: the number of equivalent internal states which is obtained in tracing Preclock, the computational complexity of tracing Preclock, the number of compared internal states (i.e. presumed keys) which is obtained in tracing Loading Key Step against one obtained equivalent state and the computational complexity of tracing Loading Key Step which is required the computational complexity of Step B2 and Step B3. As a result of a simulation, the computational complexity of Step B2 and Step B3 is about $2^{l/2+3}$ in MICKEY and about $2^{l/2+4}$ in MICKEY-128, where l is the length of secret keys. That is, β is about 2^3 in MICKEY and about 2^4 in MICKEY-128. Thus, the computational complexity of tracing the KSA (i.e. the recovering secret key from a given initial state) is about 2^{44} in MICKEY and about 2^{68} in MICKEY-128.

5. Conclusion

We have analyzed the one-wayness of the KSAs of

Table 1 The result of a simulation.

	MICKEY				MICKEY-128			
The length of secret key	16-bit	24-bit	32-bit	40-bit	16-bit	24-bit	32-bit	40-bit
The number of equivalent internal states	$2^{5.86}$	$2^{5.85}$	$2^{5.84}$	$2^{5.85}$	$2^{6.43}$	$2^{6.43}$	$2^{6.43}$	$2^{6.42}$
The computational complexity of tracing Preclock	$2^{4.89}$	$2^{4.89}$	$2^{4.88}$	$2^{4.88}$	$2^{6.15}$	$2^{6.13}$	$2^{6.13}$	$2^{6.13}$
The number of compared keys	$2^{8.06}$	$2^{12.06}$	$2^{16.06}$	$2^{20.06}$	$2^{8.03}$	$2^{12.03}$	$2^{16.03}$	$2^{20.02}$
The computational complexity of tracing Loading Key Step	$2^{11.32}$	$2^{15.32}$	$2^{18.90}$	$2^{22.91}$	$2^{11.86}$	$2^{15.87}$	$2^{19.46}$	$2^{23.44}$

MICKEY and MICKEY-128. We have proposed the method based on meet-in-the-middle attack for recovering a secret key from a given initial state. Using the proposed method, we showed that the computational complexity is about 2^{44} in MICKEY and about 2^{68} in MICKEY-128.

References

[1] R. A. Ruppel, Analysis and Design of Stream Ciphers, Springer-Verlag, Berlin, 1986.

[2] R. A. Ruppel, "Stream ciphers," Contemporary Cryptology, ed. G. J. Simmons, pp.65–134, IEEE Press, New York, 1992.

[3] P. Ekdahl and T. Johansson, "SNOW - a new stream cipher," NESSSIE, available at `http://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/snow.zip`.

[4] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minner, T. Pornin, and H. Sibert, "Sosemanuk, a fast software-oriented stream cipher," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/sosemanuk/sosemanuk_p3.pdf`.

[5] J.Håstad, J.Mattsson, and M. Näslund, "A New Version of the Stream Cipher Polar Bear," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p2ciphers/polarbear/polarbear_p2.pdf`.

[6] P. Hawkes and G. Rose, "Guess-and-determine attacks on SNOW," Proc. SAC'02, Lecture Notes in Computer Science, vol.22595, pp.2778–2791, Oct. 2005.

[7] H. Ahmadi, T. Eghlidos, and S. Khazaei, "Improved guess and determine attack in SOSEMANUK," eSTREAM, Report 2005/085, 2005, available at `http://www.ecrypt.eu.org/stream/papersdir/085.pdf`.

[8] Y. Tsunoo, T. Saito, M. Shigeri, T. Suzaki, H. Ahmadi, T. Eghlidos, and S. Khazaei, "Evaluation of SOSEMANUK with regard to guess-and-determine attacks," eSTREAM, Report 2006/009, 2006, available at `http://www.ecrypt.eu.org/stream/papersdir/2006/009.pdf`.

[9] J. Mattsson, "A guess-and-determine attack on the stream cipher Polar Bear," eSTREAM, Report 2006/019, 2006, available at `http://www.ecrypt.eu.org/stream/papersdir/2006/019.pdf`.

[10] K. Chen, M. Henricksen, W. Millan, J. Fuller, L. Simpson, E. Dawson, H. Lee, and S. Moon, "Dragon: A Fast Word Based Stream Cipher," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/dragon/dragon_p3.pdf`.

[11] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A Stream Cipher Proposal: Grain-128," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/grain128_p3.pdf`.

[12] M. Hell, T. Johansson, and W. Meier, "Grain - A Stream Cipher for Consistrained Environments," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/grain_p3.pdf`.

[13] H. Wu, "Stream Cipher HC-256," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/hc256_p3.pdf`.

[14] H. Wu, "Stream Cipher HC-128," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/hc128_p3.pdf`.

[15] C. De Cannière and B. Preneel, "Trivium Specifications," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/trivium_p3.pdf`.

[16] A. Biryukov, "A New 128-bit Key Stream Cipher LEX," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/lex_p3.pdf`.

[17] eSTREAM, ECRYPT Stream Cipher Project, IST-2002-507932, `http://www.ecrypt.eu.org/stream/`.

[18] Y. Fujikawa, T. Ohigashi, H. Kuwakado, and M. Morii, "On Onewayness of Key-Scheduling Algorithms for eSTREAM's Ciphers," IEICE Technical Report, ISEC2007-14(2007-05), May 2007. (in Japanese)

[19] S. Babbage and M. Dodd, "The stream cipher MICKEY 2.0," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf`.

[20] S. Babbage and M. Dodd, "The stream cipher MICKEY-128 2.0," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey128_p3.pdf`.

[21] W. Diffie and M. E. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," Computer, vol. 10, no. 6, pp. 74–84, Jun. 1977.

Appendix

1. *RTAPS*

RTAPS of MICKEY is defined as follows:

$$RTAPS =$$
$$\{0,1,3,4,5,6,9,12,13,16,19,20,21,22,25,28,37,38,$$
$$41,42,45,46,50,52,54,56,58,60,61,63,64,65,66,67,$$
$$71,72,79,80,81,82,87,88,89,90,91,92,94,95,96,97\}.$$

RTAPS of MICKEY-128 is defined as follows:

Table A·1 $COMP0_i$, $COMP1_i$, $FB0_i$ and $FB1_i$ of MICKEY.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$COMP0_i$		0	0	0	1	1	0	0	0	1	0	1	1	1	1	0	1	0	0	1	0	1	0	1	0
$COMP1_i$		1	0	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	0	0	0	1	1	0	1
$FB0_i$	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1
$FB1_i$	1	1	1	0	1	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0
i	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
$COMP0_i$	1	0	1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1
$COMP1_i$	0	1	1	1	0	1	1	1	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	1
$FB0_i$	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	0	1
$FB1_i$	0	1	1	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0	1	1	0
i	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74
$COMP0_i$	0	0	0	0	1	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1	1	1
$COMP1_i$	0	0	0	1	0	1	1	1	0	0	0	1	1	1	1	1	1	0	1	0	1	1	1	0	1
$FB0_i$	0	1	0	0	1	0	1	1	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
$FB1_i$	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1	1	0	1	0	1	0	0	1	0	1
i	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
$COMP0_i$	1	1	1	0	1	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	1	1	
$COMP1_i$	1	1	1	0	0	0	1	0	0	0	0	1	1	1	0	0	0	1	0	0	1	1	0	0	
$FB0_i$	1	1	0	1	0	0	0	1	1	0	1	1	1	0	0	1	1	1	0	0	1	1	0	0	0
$FB1_i$	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	1

$$RTAPS = \{0,4,5,8,10,11,14,16,20,25,30,32,35,36,38,42,43,46,$$
$$50,51,53,54,55,56,57,60,61,62,63,65,66,69,73,74,76,$$
$$79,80,81,82,85,86,90,91,92,95,97,100,101,105,106,$$
$$107,108,109,111,112,113,115,116,117,127,128,129,$$
$$130,131,133,135,136,137,140,142,145,148,150,152,$$
$$153,154,156,157\}.$$

The number of elements of $RTAPS$ of MICKEY-128 is 78. Therefore, tracing Preclock and Loading Key Step is done same as MICKEY.

2. Masks for $CLOCK\_S(S, IB\_S, CB\_S)$

Table A·1 shows the details of four masks; $COMP0_i$, $COMP1_i$, $FB0_i$ and $FB1_i$ of MICKEY. Table A·2 shows the details of four masks for MICKEY-128.

Table A·2 $COMP0_i$, $COMP1_i$, $FB0_i$ and $FB1_i$ of MICKEY-128.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$COMP0_i$		1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	0	1	1	0	1	0	1
$COMP1_i$		0	0	0	1	1	0	0	1	1	1	1	1	0	0	0	1	0	0	1	1	0	0
$FB0_i$	1	1	1	1	0	1	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0
$FB1_i$	1	1	0	1	0	1	0	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1	1
i	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
$COMP0_i$	1	1	0	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0
$COMP1_i$	0	1	0	1	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	1	1	1	1
$FB0_i$	0	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0	1
$FB1_i$	1	1	1	0	1	1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	1	0	0
i	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
$COMP0_i$	0	0	0	1	1	0	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	0	0
$COMP1_i$	0	0	0	1	1	0	1	1	0	1	0	1	1	1	1	1	1	1	0	0	0	0	0
$FB0_i$	0	1	1	1	1	1	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0
$FB1_i$	0	1	1	1	1	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	1	0	1
i	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91
$COMP0_i$	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1
$COMP1_i$	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1
$FB0_i$	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1	1	0	1	1	0
$FB1_i$	1	0	1	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1	1
i	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114
$COMP0_i$	1	1	1	0	1	0	0	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	1
$COMP1_i$	1	1	1	0	1	0	1	0	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0
$FB0_i$	0	1	1	0	1	0	0	0	1	0	0	1	1	1	0	1	0	0	1	0	0	0	1
$FB1_i$	0	1	1	0	1	0	0	0	1	0	0	1	1	1	0	1	1	1	1	0	1	1	0
i	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137
$COMP0_i$	1	0	1	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	1	0	1
$COMP1_i$	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	1	0	1	1	1
$FB0_i$	0	1	0	1	0	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1	1	1	1
$FB1_i$	1	0	0	1	0	0	0	1	1	0	1	1	0	1	1	1	1	0	1	1	1	0	0
i	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	
$COMP0_i$	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0		
$COMP1_i$	0	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1		
$FB0_i$	0	1	0	0	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0	0	1	
$FB1_i$	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	0	0	1	0	0	0	