Key Recovery Attack on Stream Cipher Mir-1 Using a Key-Dependent S-Box

Yukiyasu Tsunoo[1], Teruo Saito[2], Hiroyasu Kubo[2], and Tomoyasu Suzaki[1]

[1] NEC Corporation
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan
{tsunoo@BL,t-suzaki@cb}.jp.nec.com
[2] NEC Software Hokuriku, Ltd.
1, Anyoji, Hakusan, Ishikawa 920-2141, Japan
{t-saito@qh,h-kubo@ps}.jp.nec.com

Abstract. Mir-1 is a stream cipher proposed for Profile 1 at the ECRYPT Stream Cipher Project (eSTREAM). The Mir-1 designer claims a security level of at least 2^{128}, meaning that the secret key cannot be recovered or that the Mir-1 output sequence cannot be distinguished from a truly random number sequence more efficiently than an exhaustive search. At SASC 2006, however, a distinguishing attack on Mir-1 was proposed making use of vulnerabilities in Mir-1 initialization. This paper shows that unknown entries in the key-dependent S-box used by Mir-1 can be classified into partially equivalent pairs by extending the SASC 2006 technique. It also demonstrates an attack that applies that information to recovering the Mir-1 secret key more efficiently than an exhaustive search. To the best of the authors' knowledge, the results described in this paper represent the first successful key recovery attack on Mir-1.

Keywords: eSTREAM, key-dependent S-box, key recovery attack, Mir-1, stream cipher.

1 Introduction

Recently, there have been efforts around the world to standardize encryption. For example, the selection of AES [13] as a standard cipher has been announced in the United States, and the NESSIE project [14] was established in Europe to select standard ciphers. The NESSIE project aimed, in particular, to select secure encryption primitives choosing stream ciphers as one member of that category. However, the stream ciphers offered by the NESSIE project were tackled by numerous cryptanalyses during a 3-year evaluation phase, and ultimately not even a single secure candidate stream cipher remained. In response, the design and analysis of stream ciphers has been receiving increasing attention.

In February 2004, European Network of Excellence for Cryptology (ECRYPT) [5] was established with the objective of encouraging cooperation among European researchers working on information security. The ECRYPT Stream Cipher Project (henceforth "eSTREAM") [4] was established in 2005 by the ECRYPT

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 128–140, 2008.

working group Symmetric Techniques Virtual Lab (STVL), and new stream ciphers were made public. As a result, 34 candidates were submitted to eSTREAM. After more than three years and three phases of evaluation at eSTREAM, 8 stream ciphers have been chosen for the final portfolio.

Mir-1 [10] is a stream cipher proposed at eSTREAM having a 128-bit secret key and a 64-bit initial vector (IV). Although proposed for Profile 1 as a stream cipher for high-speed software implementation, Mir-1 was archived in phase 1. Mir-1 uses a multi-word T-function and a key-dependent S-box function whose security has not been sufficiently studied. Its designer, however, claimed a security level of at least 2^{128}, meaning that the secret key cannot be efficiently recovered or that the Mir-1 output sequence cannot be efficiently distinguished from a truly random number sequence faster than an exhaustive search.

At SASC 2006, however, a distinguishing attack was proposed on Mir-1 [18]. This cryptanalysis method exploits vulnerabilities in T-function characteristics and Mir-1 initialization. The method can distinguish with high probability the output sequence of Mir-1 from a truly random number sequence by choosing only three or four IV pairs. The amount of data needed here is theoretically no more than 2^{10} words. A countermeasure against this technique was also proposed in 2007 [19]. A key recovery attack on Mir-1, however, has not been reported.

In this paper, we show that unknown entries in the key-dependent S-box used by Mir-1 can be classified into partially equivalent pairs by extending the technique of [18]. We also demonstrate an attack for recovering the Mir-1 secret key more efficiently than an exhaustive search based on that information. This attack can recover a secret key with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{110.76}$ table lookups, and a memory complexity of about $2^{32.86}$ bytes. Furthermore, under conditions with no memory limitations, the attack can recover a secret key with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{79.32}$ table lookups, and a memory complexity of about $2^{65.81}$ bytes. We show that Mir-1 incorporating the countermeasure described in [19] is robust to this attack.

To the best of our knowledge, the results described in this paper represent the first successful key recovery attack on Mir-1. We expect the results provided here to be useful in evaluating the security of the many previously proposed block ciphers [2,3,12,16,17] and stream ciphers [1,6,7,9,11,15,20] that use a key-dependent S-box.

Section 2 describes the structure of Mir-1. Section 3 describes a key recovery attack on Mir-1, Section 4 discusses the complexity of this attack and a countermeasure to it, and Section 5 concludes the paper.

2 Description of Mir-1

This section describes the structure of Mir-1, the stream cipher proposed by Maximov at eSTREAM 2005. Ciphertexts are computed by exclusive ORing

plaintexts with the keystream generated by the cipher. The keystream generation and initialization of Mir-1 is explained below.

2.1 Notation and Definition

In this paper, bit-wise XOR, AND, and OR are represented by $\oplus$, &, and |, respectively. Addition and multiplication on mod 2^{64} are denoted by + and $\cdot$, respectively. A 64-bit word X rotated to the left by t bits is represented by either $X \lll t$ or $ROL_t(X)$. The byte unit and bit unit of 64-bit word X are set as follows, where $\|$ represents data concatenation.

$$\begin{aligned} X &= X.byte_7 \,\|\, X.byte_6 \,\|\, \cdots \,\|\, X.byte_0 \\ &= X.bit_{63} \,\|\, X.bit_{62} \,\|\, \cdots \,\|\, X.bit_0 \end{aligned}$$

The a^{th} through the b^{th} bits of 64-bit word X are represented by $X[a, b]$. Using the notation described above, we express them as

$$X[a, b] = X.bit_b \,\|\, X.bit_{b-1} \,\|\, \cdots \,\|\, X.bit_a$$

The secret key KEY of Mir-1 is 128-bit long and its initial vector IV is 64-bit long. They are defined as follows.

$$\begin{aligned} KEY &= k_{15} \,\|\, k_{14} \,\|\, \cdots \,\|\, k_0 \\ IV &= IV_7 \,\|\, IV_6 \,\|\, \cdots \,\|\, IV_0 \end{aligned}$$

Here, k_i $(0 \le i \le 15)$ and IV_i $(0 \le i \le 7)$ are one-byte variables.

2.2 Keystream Generation

This section treats Mir-1's keystream generation, which consists of roughly two parts: the loop state update (LS update) and the automata state update (AS update).

The LS update has four registers of 64-bit words $x_i (i = 0, 1, 2, 3)$. Register x_i is updated by a multiword T-function [8]. The LS update is shown in Fig. 1. It guarantees the cycle of maximum length 2^{256}.

The AS update holds two words of 64-bit registers A and B, and it computes A' and B' using the update function shown in Fig. 2. A' and B' correspond to A and B in the next time step. When A' and B' are computed, the register value from the LS update is two 64-bit words obtained by concatenating the upper 32 bits of each of the four registers x_0, x_1, x_2, and x_3. Each 64-bit word is denoted as follows.

$$(x_{i+2}[32, 63] \,\|\, x_i[32, 63]) \quad (i = 0, 1)$$

The keystream generation part of Mir-1 performs the LS update and AS update at each clock, and outputs keystream z, that is, the 64-bit B' computed by the AS update.

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 + (s) & + 2 \cdot x_2 \cdot (x_1 \mid C_1) \\ x_1 + (s \,\&\, x_0) & + 2 \cdot x_2 \cdot (x_3 \mid C_3) \\ x_2 + (s \,\&\, x_0 \,\&\, x_1) & + 2 \cdot x_0 \cdot (x_3 \mid C_3) \\ x_3 + (s \,\&\, x_0 \,\&\, x_1 \,\&\, x_2) & + 2 \cdot x_0 \cdot (x_1 \mid C_1) \end{pmatrix}$$

$$s = (x_0 \& x_1 \& x_2 \& x_3 + C_0) \oplus x_0 \& x_1 \& x_2 \& x_3$$

$$C_0 = 0\text{x}1248842112488421$$
$$C_1 = 0\text{x}1248124812481248$$
$$C_3 = 0\text{x}4812481248124812$$

Fig. 1. Loop state update

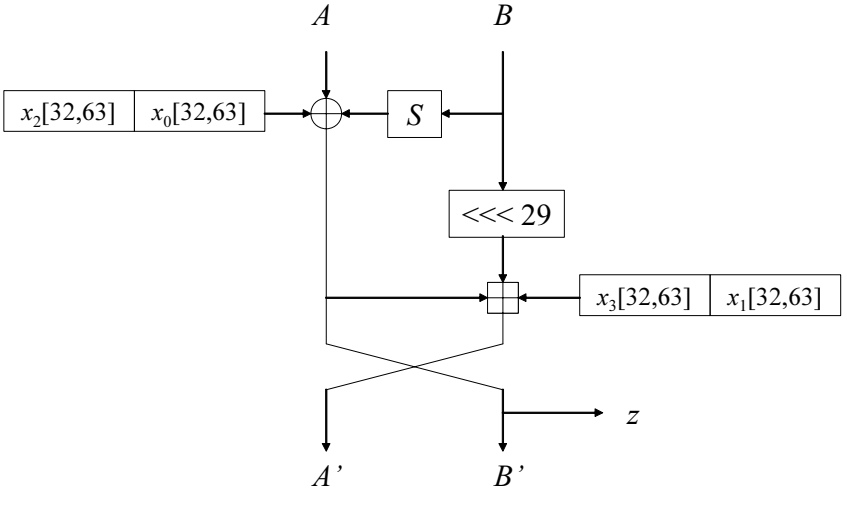

Fig. 2. Automata state update

2.3 Initialization

This section describes Mir-1's initialization part, which also consists of roughly two parts: the key setup and the IV setup.

The key setup initializes register $x_i (i = 0, 1, 2, 3)$ and registers A and B, using a 128-bit secret key. The key setup is shown in Fig. 3.

First, the key setup computes an 8-bit S-box, which varies depending on the secret key value referred to as the secret S-box, using the equation shown below.

1. Initialize secret S-box
2. $A = x_1 = (k_7 \| \ldots \| k_0)$
   $B = x_3 = (k_{15} \| \ldots \| k_8)$
   $x_0 = C_0$
   $x_2 = C_1$
3. Repeat 8 times
   Loop State Update
   Automata State Update

Fig. 3. Key setup

Here, $SR[\cdot]$ means the S-box of AES. Each entry is computed for $i = 0, \ldots, 255$.

$$S[i] = SR[\cdots SR[SR[i \oplus k_0] \oplus k_1] \oplus \cdots \oplus k_{15}]$$

The IV setup uses a 64-bit initial vector to update register $x_i (i = 0, 1, 2, 3)$ as well as registers A and B. The IV setup is shown in Fig. 4.

3 Key Recovery Attack

This section explains how to apply a key recovery attack against the Mir-1 stream cipher. On applying this attack, it is assumed that the following preconditions are met in an actual situation using stream ciphers.

- The secret key is fixed during the attack.
- Attackers can choose the IV freely.
- Attackers can obtain the keystream generated using the given IV.

3.1 Previous Distinguisher and Extended Distinguisher

This following describes the distinguisher reported in [19] and the distinguisher obtained by extending that distinguisher. To begin with, Theorem 1 holds with respect to the keystream of Mir-1.[1]

Theorem 1. *Given keystreams za and zb generated by IVa ($IV_i = a$, 0x00 $\leq a \leq$ 0xff, $0 \leq i \leq 7$) and IVb ($IV_i = b \neq a$, 0x00 $\leq b \leq$ 0xff, $0 \leq i \leq 7$) that satisfy Eq. (1), Eq. (3) will be satisfied at time t satisfying Eq. (2).*

$$S[a] \equiv S[b] \pmod 2 \tag{1}$$

$$za^{(t)}.byte_0 = zb^{(t)}.byte_0 \tag{2}$$

$$ROL_{29}(za^{(t-1)} \oplus zb^{(t-1)}) \equiv za^{(t+1)} \oplus zb^{(t+1)} \pmod 2 \tag{3}$$

[1] See [19] for proof of Theorem 1.

1. $x_0.byte_4 = x_0.byte_4 \oplus S[IV_0] \oplus S[IV_1] \oplus S[IV_2]$
   $x_1.byte_4 = x_1.byte_4 \oplus S[IV_0] \oplus S[IV_3] \oplus S[IV_4]$
   $x_2.byte_4 = x_2.byte_4 \oplus S[IV_2] \oplus S[IV_5] \oplus S[IV_7]$
   $x_3.byte_4 = x_3.byte_4 \oplus S[IV_3] \oplus S[IV_6] \oplus S[IV_7]$
2. $x_0.byte_0 = x_0.byte_0 \oplus S[IV_3] \oplus S[IV_5]$
   $x_1.byte_0 = x_1.byte_0 \oplus S[IV_7] \oplus S[IV_6]$
   $x_2.byte_0 = x_2.byte_0 \oplus S[IV_0] \oplus S[IV_1]$
   $x_3.byte_0 = x_3.byte_0 \oplus S[IV_2] \oplus S[IV_4]$
3. $A.byte_0 = A.byte_0 \oplus S[IV_0] \oplus S[IV_5] \oplus S[IV_6]$
   $A.byte_4 = A.byte_4 \oplus S[IV_1] \oplus S[IV_3] \oplus S[IV_5]$
   $B.byte_0 = B.byte_0 \oplus S[IV_1] \oplus S[IV_4] \oplus S[IV_7]$
   $B.byte_4 = B.byte_4 \oplus S[IV_2] \oplus S[IV_4] \oplus S[IV_6]$
4. Repeat 2 times
   Loop State Update
   Automata State Update

Fig. 4. IV setup

A distinguishing attack using Theorem 1 was proposed in [19]. This attack requires about 2^{10} words at most to distinguish the output sequence of Mir-1 from a truly random number sequence. Theorem 2 below can be easily obtained by extending Theorem 1.

Theorem 2. *Given keystreams za and zb generated by IVa ($IV_i = a$, 0x00 $\le a \le$ 0xff, $0 \le i \le 7$) and IVb ($IV_i = b \ne a$, 0x00 $\le b \le$ 0xff, $0 \le i \le 7$) that satisfy Eq. (4), Eq. (6) will be satisfied at time t satisfying Eq. (2) and Eq. (5).*

$$S[a] \equiv S[b] \pmod{2^n, 1 \le n \le 7} \tag{4}$$

$$ROL_{29}(za^{(t-1)}) \equiv ROL_{29}(zb^{(t-1)}) \pmod{2^n, 1 \le n \le 7} \tag{5}$$

$$za^{(t+1)} \equiv zb^{(t+1)} \pmod{2^n, 1 \le n \le 7} \tag{6}$$

Proof. Denoting register x_i updated by IVa and register x_i updated by IVb as xa_i and xb_i, respectively, the condition of Eq. (1) in Theorem 1 can be replaced by Eq. (4) so that register x_i satisfies the following relation.

$$xa_i[0, 31+n] = xb_i[0, 31+n] \quad (0 \le i \le 3\ ,\ 1 \le n \le 7)$$

This leads to Eq. (7):

$$ROL_{29}(za^{(t-1)} \oplus zb^{(t-1)}) \equiv za^{(t+1)} \oplus zb^{(t+1)} \pmod{2^n, 1 \le n \le 7} \tag{7}$$

As a result, substituting Eq. (5) in Eq. (7) gives Eq. (6). □

3.2 Classification of Key-Dependent S-Box

In this section, we describe a method for classifying unknown entries of the key-dependent S-box into partially equivalent pairs using Theorem 2. Specifically, for the case of $n = 7$, the following procedure classifies key-dependent S-box entries into 128 pairs.

1-1. Obtain an N-word keystream for IVa where a is any of 256 values ($IV_i = a$, 0x00 $\leq a \leq$ 0xff, $0 \leq i \leq 7$).[2]
1-2. For keystreams za and zb generated by IVa and IVb ($a < b \leq$ 0xff), check whether Theorem 2 is satisfied for w successive times.[3] If Theorem 2 is satisfied, the IVa and IVb in question are taken to be the 1st pair.
1-3. Repeat step 1-2 for all combinations of (a, b).

The above procedure can classify unknown entries of the key-dependent S-box into 128 pairs whose lower 7 bits are equal. Note, however, that this procedure cannot determine entry values of the key-dependent S-box.

3.3 Key Recovery Method

This section describes a method for recovering a secret key using information on the key-dependent S-box obtained by the method presented in Section 3.2. Specifically, the following procedure - where step 0 signifies a precomputation step - can recover the secret key.

0. For all (x, y) combinations (0x00 $\leq x < y \leq$ 0xff), store secret-key m-byte $(k_{15-m}, \cdots, k_{15})$ candidates satisfying Eq. (8) in table tbl.

$$\begin{aligned} &SR[\cdots SR[x \oplus k_{15-m}] \oplus \cdots \oplus k_{15}] \\ &\equiv SR[\cdots SR[y \oplus k_{15-m}] \oplus \cdots \oplus k_{15}] \pmod{2^7} \end{aligned} \tag{8}$$

   Here, the index of tbl is a 16-bit value formed by concatenating x and y; it can take on 2^{15} values.
1. Classify key-dependent S-box entries using the method of Section 3.2.
2. Guess l-byte of the secret key $(k_0, \cdots, k_{l-1},\ l = 16 - m)$.
3. Choose the first pair (a, b) obtained in step 1 and calculate (x, y) using the following equations.

$$\begin{aligned} x &= SR[\cdots SR[a \oplus k_0] \oplus \cdots \oplus k_{l-1}] \\ y &= SR[\cdots SR[b \oplus k_0] \oplus \cdots \oplus k_{l-1}] \end{aligned}$$

4. Using the value of (x, y) calculated in step 3 as an index value, obtain a secret-key m-byte candidate $(k_{15-m}, \cdots, k_{15})$ from tbl.

[2] Here, N represents the size necessary for classifying key-dependent S-box entries.
[3] Here, w represents the number of times ts needed for classifying key-dependent S-box entries.

5. Choose the 2^{nd} to 128^{th} (a, b) pairs and narrow down the secret-key m-byte $(k_{15-m}, \cdots, k_{15})$ candidates by steps 3 and 4. Once no more candidates remain, guess another l-byte $(k_0, \cdots, k_{l-1})$ of the secret key in step 2 and repeat steps 2 to 5.
6. If a secret-key m-byte $(k_{15-m}, \cdots, k_{15})$ candidate becomes uniquely determined in steps 2 to 5, the 16 bytes that include the l-byte $(k_0, \cdots, k_{l-1})$ guessed in step 2 are taken to be the correct secret key.

Given that the key-dependent S-box can be correctly classified in step 1, the above procedure can recover the secret key more efficiently than an exhaustive search.

4 Discussion

4.1 Complexity of Attack

This section theoretically examines the complexity of each step in the procedure presented in Section 3.3.

First, we investigate the complexity of step 0. The computational complexity T_0 for looking up SR $2m$ times for all combinations of $(x, y, k_{15-m}, \cdots, k_{15})$ is given by

$$\begin{aligned} T_0 &= 2^8 \times 2^7 \times 2^{8m} \times 2m \\ &= m \times 2^{8m+16} \end{aligned}$$

Furthermore, the size of memory M_0 for storing *tbl* is the product of index size, entry size, and number of entries as follows.

$$\begin{aligned} M_0 &= 2^{15} \times m \times 2^{8(m-1)} \\ &= m \times 2^{8m+7} \end{aligned}$$

Next, we examine the complexity of step 1. Since the probability that a keystream that satisfies the conditions of Eqs. (2) and (5) exists is 2^{-15}, keystream generation size is $w \times 2^{15}$ words for one secret-key/IV-pair. Now, to classify the key-dependent S-box into 128 pairs, we consider that the probability of erroneously satisfying Eq. (6) due to variation in the key-dependent S-box should be made sufficiently small, and we therefore set $w = 20$ on the basis of $2^{-128} \gg 2^{-7w}$.[4] Accordingly, the computational complexity T_1 of step 1 is the sum of computational complexity T_1' of step 1-1 and computational complexity T_1'' of steps 1-2 and 1-3 as follows.

$$\begin{aligned} T_1 &= T_1' + T_1'' \\ T_1' &= 2^8 \times 20 \times 2^{15} \\ &= 2^{27.32} \end{aligned}$$

[4] Consider that $2^{-128} \gg \frac{2^{-128}}{1000}$.

$$\begin{aligned} T_1'' &= 20 \times 2^{15} \times (\frac{255}{2} + \frac{253}{2} + \cdots + \frac{1}{2}) \\ &= 2^{18.32} \times \frac{128(1+255)}{2} \\ &= 2^{32.32} \end{aligned}$$

In addition, the amount of required data D_1 and memory M_1 can be given as follows.

$$\begin{aligned} D_1 = M_1 &= 1 \times 2^8 \times 2^{19.32} \times 2^3 \\ &= 2^{30.32} \end{aligned}$$

Now, we examine the complexity of steps 2 to 5. Here, we guess l bytes of the secret key and narrow down the remaining m bytes using *tbl* generated in step 1. If l-byte is correctly guessed, key candidates will be narrowed down to one correct candidate via 128 table lookups. On the other hand, if the l-byte guess is incorrect, key candidates will be narrowed down to one candidate on average on the m^{th} table lookup, and all erroneous key candidates will be rejected on the $(m+1)^{\text{th}}$ table lookup. Computational complexity T_2 for steps 2 to 5 is therefore given as follows.

$$\begin{aligned} T_2 &= 1 \times (2l+1) \times 128 + (2^{8l} - 1) \times (2l+1) \times (m+1) \\ &\approx (2l+1)(m+1) \times 2^{8l} \end{aligned}$$

The complexity of the attack presented in this paper is consequently the sum of the complexities for each of the above steps. Computational complexity T,[5] amount of data D, and memory M are each given as follows:

$$\begin{aligned} T &= T_0 + T_1 + T_2 \\ &= m \times 2^{8m+16} + 2^{27.32} + 2^{32.32} + (2l+1)(m+1) \times 2^{8l} \\ &\approx m \times 2^{8m+16} + (33 - 2m)(m+1) \times 2^{128-8m} \\ D &= D_1 \\ &= 2^{30.32} \\ M &= M_0 + M_1 \\ &= m \times 2^{8m+7} + 2^{30.32} \end{aligned}$$

Finally, we examine the case for which the attack presented in this paper is most efficient. Assuming no memory limitations, computational complexity T when satisfying

$$m \times 2^{8m+16} \approx (33 - 2m)(m+1) \times 2^{128-8m}$$

takes on a minimum value for $m = 7$. At this time, computational complexity T is:

$$\begin{aligned} T &\approx 7 \times 2^{72} + 19 \times 8 \times 2^{72} \\ &\approx 2^{79.32} \end{aligned}$$

[5] Although the units of computational complexity differ, the number of table lookups effectively dominates.

In this case, memory M becomes:

$$\begin{aligned} M &= 7 \times 2^{63} + 2^{30.32} \\ &\approx 2^{65.81} \end{aligned}$$

This amount of data cannot, however, be realistically stored. If we therefore assume that the maximum amount of memory that can be realistically provided is 1 terabyte, then, from $M = m \times 2^{8m+7} + 2^{30.32} < 2^{40}$, we get $m = 3$. Computational complexity T in this case turns out to be:

$$\begin{aligned} T &\approx 3 \times 2^{40} + 27 \times 4 \times 2^{104} \\ &\approx 2^{110.76} \end{aligned}$$

Memory M at this time is:

$$\begin{aligned} M &= 3 \times 2^{31} + 2^{30.32} \\ &\approx 2^{32.86} \end{aligned}$$

4.2 Experiment

We performed an experiment to determine whether a secret key could indeed be recovered by the method presented in Section 3.3. The following conditions were established for this experiment.

- Let $(l, m) = (15, 1)$. However, because an exhaustive search across 15 bytes is computationally difficult, the correct secret key was substituted for the 12 bytes $(k_0, \cdots, k_{11})$ and an exhaustive search was performed over the 3 bytes $(k_{12}, \cdots, k_{14})$.[6]
- Letting $w = 20$, a 2^{21}-word keystream was prepared beforehand for one secret-key/IV-pair.
- The probability of success was determined from the results of 100 trials with randomly set secret keys.
- The size of *tbl* generated by the precomputation step was determined.
- The number of *tbl* lookups that dominate computational complexity T was determined.

On performing the experiment based on the above conditions, we successfully classified the key-dependent S-box and derived the secret key for all of the secret keys used in the experiment. The size of *tbl* needed for the precomputation step turned out to be about 2^{15} bytes, which agreed with the theoretical value given in Section 4.1.

On the other hand, the number of *tbl* lookups required for the attack was $2^{28.55}$, which was less than the theoretical value[7] given in Section 4.1. This is because *tbl* generated by the precomputation step is ideally not a table with 1 entry per index value.[8] If we therefore consider the number of *tbl* lookups taking

[6] The 3 bytes used here for performing an exhaustive search can be any 3 bytes from k_0 to k_{14} and not just k_{12} to k_{14}.

[7] $31 \times 2 \times 2^{24} = 2^{29.96}$

[8] Since the SR differential probability is 2^{-6}, the number of entries per index value is divided into 4, 2, and 0.

into account the number of entries per index value in *tbl*, a correctly guessed key would coincide with a correct key obtained on average in 2^{23} attempts in a 24-bit exhaustive search. Here, we have 1 *tbl* lookup (number of entries = 0) at probability 1/2 per index value and 2 *tbl* lookups (number of entries = 2; all candidates are rejected at the 2^{nd} *tbl* lookup) at probability 1/2, so that expectation E of the number of *tbl* lookups needed for the attack would be as follows.

$$E = 2^{23} \times 31 \times \frac{1}{2}(1 + 2) \approx 2^{28.54}$$

The result obtained by experiment is therefore about the same as the expected value.

4.3 Countermeasure

This section discusses a countermeasure to the attack presented in this paper. The following three structural factors can be given as vulnerabilities in the Mir-1 cipher with respect to this attack.

1. A difference intended for register x_i can be input by choosing the IV.
2. If the difference in the lower n bits of register x_i is 0, it will remain 0 regardless of the number of times LS update is performed.
3. The input value of the key-dependent S-box can be freely chosen by choosing the keystream.

In particular, items 1 and 2 describe vulnerabilities related to IV setup and LS update using the T-function, and item 3 describes a vulnerability related to AS update using the key-dependent S-box. In [19], a distinguishing attack based on vulnerabilities 1 and 2 is proposed along with a countermeasure to that attack. This countermeasure has the following two features.

1. Modifies the method for inserting IV and improves IV setup.
2. Adds bit-permutation processing in which register x_i straddles word boundaries (loop state permutation).

Either of the above enhancements aims to eliminate vulnerabilities in the Mir-1 initialization process. Enhancement 1 aims to prevent a chosen IV attack from being mounted while enhancement 2 aims to spread differential characteristics of the T-function throughout words.

It is shown in [19] that Theorem 1 does not hold as a result of this countermeasure, which means that Theorem 2 likewise does not hold. It can therefore be seen that, in the same way that the distinguishing attack cannot be mounted, neither can classification of the key-dependent S-box be performed. In short, the countermeasure presented in [19] is also robust to the attack presented in this paper.

In addition to the countermeasure proposed in [19], we can consider a means of eliminating the vulnerability in AS update that uses a key-dependent S-box

(vulnerability 3 above). Such an enhancement, however, could have a big effect on security with respect to other types of attacks, and for this reason, a careful study must be made. Nevertheless, we have seen that an attack of the kind presented here is capable of recovering the secret key by grouping the entries of the key-dependent S-box in a certain way even though the values of those entries are unknown. Care must therefore be taken when designing a cipher using a key-dependent S-box.

5 Conclusion

In this paper, we showed how unknown entries in the key-dependent S-box used by Mir-1 could be classified into 128 pairs by extending the distinguisher presented in [19] to 7 bits. We described an attack for recovering the Mir-1 secret key more efficiently than an exhaustive search based on that information.

With this method, the secret key can be recovered with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{110.76}$ table lookups, and a memory complexity of about $2^{32.86}$ bytes. And under conditions with no memory limitations, the attack can recover the secret key with a data complexity of about $2^{30.32}$ bytes, a computational complexity of about $2^{79.32}$ table lookups, and a memory complexity of about $2^{65.81}$ bytes. We also reported that the countermeasure presented in [19] is effective in resisting this type of attack.

To the best of our knowledge, the results described in this paper represent the first successful key recovery attack on Mir-1. The attack described here can recover the secret key without having to directly determine the values of key-dependent S-box entries. This result shows the possibility of uncovering information on the secret key in ciphers that do not use the key-dependent S-box appropriately. Accordingly, we expect this result to be useful in evaluating the security of block ciphers and stream ciphers that use a key-dependent S-box.

References

1. Anashin, V., Bogdanov, A., Kizhvatov, I., Kumar, S.: ABC: A New Fast Flexible Stream Cipher. eSTREAM submission (2005)
2. Crowley, P.: Mercy: A Fast Large Block Cipher for Disk Sector Encryption. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 49–63. Springer, Heidelberg (2001)
3. Cusick, T.W., Wood, M.C.: The REDOC II Cryptosystem. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 545–563. Springer, Heidelberg (1991)
4. ECRYPT Stream Cipher Project (eSTREAM), `http://www.ecrypt.eu.org/stream/`
5. European Network of Excellence for Cryptology (ECRYPT), `http://www.ecrypt.eu.org/`
6. Halevi, S., Coppersmith, D., Jutla, C.S.: Scream: A Software-Efficient Stream Cipher. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 195–209. Springer, Heidelberg (2002)

7. Hawkes, P., Paddon, M., Rose, G.G., de Vries, M.W.: Primitive Specification for SSS. eSTREAM submission (2005)
8. Klimov, A., Shamir, A.: New Cryptographic Primitives Based on Multiword T-functions. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 1–15. Springer, Heidelberg (2004)
9. Li, A.-P.: A New Stream Cipher: Dicing. eSTREAM submission (2005)
10. Maximov, A.: A New Stream Cipher "Mir-1". eSTREAM submission (2005)
11. McGrew, D.A., Fluhrer, S.R.: The Stream Cipher LEVIATHAN. NESSIE submission (2000)
12. Merkle, R.C.: Fast Software Encryption Functions. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 476–501. Springer, Heidelberg (1991)
13. National Institute of Standards and Technology (NIST), Federal Information Processing Standard (FIPS) 197, Advanced Encryption Standard (AES)
14. New European Schemes for Signature, Integrity, and Encryption (NESSIE), `https://www.cosic.esat.kuleuven.be/nessie/`
15. Rose, G.G., Hawkes, P.: Turing: A Fast Stream Cipher. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 290–306. Springer, Heidelberg (2003)
16. Schneier, B.: Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809, pp. 191–204. Springer, Heidelberg (1994)
17. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128-Bit Block Cipher. NIST AES proposal (1998)
18. Tsunoo, Y., Saito, T., Kubo, H., Shigeri, M.: Cryptanalysis of Mir-1, a T-function Based Stream Cipher. In: Proceedings of SASC 2006, pp. 185–197 (2006), `http://www.ecrypt.eu.org/stvl/sasc2006/`
19. Tsunoo, Y., Saito, T., Kubo, H., Suzaki, T.: Cryptanalysis of Mir-1: A T-function-Based Stream Cipher. IEEE Transactions on Information Theory 53(11), 4377–4383 (2007)
20. Wu, H.: A New Stream Cipher HC-256. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 226–244. Springer, Heidelberg (2004)