Cryptanalysis of Mir-1, a T-function Based Stream Cipher

Yukiyasu Tsunoo¹, Teruo Saito², Hiroyasu Kubo², and Maki Shigeri²

¹ NEC Corporation

1753 Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan

tsunoo@BL.jp.nec.com

² NEC Software Hokuriku Ltd.

1 Anyoji, Hakusan, Ishikawa 920-2141, Japan

{t-saito@qh, h-kubo@ps, m-shigeri@pb}.jp.nec.com

Abstract. This paper describes the cryptanalysis of Mir-1, a T-function based stream cipher proposed at eSTREAM (the ECRYPT Stream Cipher Project) in 2005. It uses a multiword T-function, with four 64-bit words, as its basic structure. Mir-1 operations process the data in every 64 bits (one word) to generate a keystream.

This paper discusses a distinguishing attack against Mir-1, one that exploits the T-function characteristics and the Mir-1 initialization. With merely three or four IV pairs, this attack can distinguish a Mir-1 output sequence from a true random sequence. In this case, the amount of data theoretically needed for cryptanalysis is only 2^{10} words.

Key words: Mir-1, ECRYPT, eSTREAM, stream cipher, pseudo-random number generator, distinguishing attack

1 Introduction

Over the past two decades, a variety of stream ciphers have been proposed. Many of these use a linear feedback shift register (LFSR) with a non-linear Boolean function to generate a keystream. However, attacks that exploit the linear characteristics of LFSR have been proposed [2, 12, 15]. LFSR-based stream ciphers might be vulnerable to such algebraic cryptanalysis.

In 2003, Klimov and Shamir proposed the T-function as a new primitive that can be used as an alternative to LFSR. The T-function is suitable for software implementation. Though it is a form of non-linear mapping, it uses a combination of operations including ADD, SUB, MUL, XOR, AND, and OR for a single cycle of maximal length. Klimov and Shamir insist that the T-function can be used not only for a stream cipher, but also for a block cipher and a hash function.

Various T-function based stream ciphers were then proposed. In 2005, Hong et al. proposed single-cycle T-functions using the S-box properties, as well as TSC-1/2 which are stream ciphers using these proposed T-functions [3]. They reported that TSC-1 is suitable for hardware implementation while TSC-2 is a stream cipher suitable for software implementation. At the FSE 2005 rump session, though, these ciphers were broken using the T-function properties [5]. In

the same year, Hong et al. proposed TSC-3, a new version of TSC, at eSTREAM [4]. However, shortly afterwards Muller and Peyrin broke this version [14].

At eSTREAM 2005, Maximov proposed Mir-1, a T-function based stream cipher [11]. The cipher uses data updated using a T-function as a key each time, and it generates a keystream through randomization by an S-box whose entries change depending on a secret key. In this paper, we propose a new cryptanalysis that exploits the T-function characteristics and the Mir-1 initialization. This method makes it possible to distinguish the output sequence of Mir-1 from a true random sequence with only three or four initial vector (IV) pairs. The amount of data theoretically needed by the method is only about 2^{10} words. Thus, with a practical amount of computation, this attack could be a threat to Mir-1.

The following section provides an overview of T-functions and recently proposed T-function based stream ciphers. Section 3 describes the structure of Mir-1. Section 4 explains how the output sequence of Mir-1 can be distinguished from a true random sequence through the T-function properties and the Mir-1 initialization. Section 5 concludes this paper.

2 T-function

This section provides a basic explanation of the T-function. The details are provided in the original paper published by Klimov and Shamir.

2.1 T-function Proposed by Klimov and Shamir

In 2002, Klimov and Shamir proposed the T-function as a new class for invertible mapping [6]. Their T-function is a single-word T-function and features single n -bit word mapping. The i -th bit of a single-word T-function output depends only on the 0th through i -th bits of its input. Single-word T-functions include arithmetic operations such as ADD, SUB, and MUL, and logical operations such as OR, AND, and XOR. These operations are referred to as primitive operations, and they are very useful because they can be processed within one clock and one cycle on many kinds of processor.

Klimov and Shamir used various combinations of these operations to design many kinds of T-functions. These T-functions feature a single cycle of maximal length. This kind of function could be used as an alternative to LFSR. However, a single-word T-function is not so useful, because its bit size n is limited to 32 or 64 in today's processors.

In 2004, Klimov and Shamir proposed multiword T-functions, which were expanded versions of single-word T-functions [8]. Multiword T-functions define m n -bit words for mapping, and they offer a single-cycle of maximal length as is offered by single-word T-functions.

The following is a more specific description of multiword T-functions with m n -bit words. If each of the m n -bit words is represented by x_k ($k = 0, \dots, m-1$),

the set of m words x is expressed as $\mathbf{x} = (x_k)_{k=0}^{m-1}$. The i -th bit of each word $[x_k]_i$ is then denoted as

$$[x_k] = \sum_{i=0}^{n-1} [x_k]_i 2^i$$

The layer of the i -th bit of word $\mathbf{x}$ is expressed as

$$[\mathbf{x}]_i = \sum_{k=0}^{m-1} [x_k]_i 2^k$$

Figure 1 outlines the multiword T-function defined below, where $m = 4$.

Fig. 1. Multiword T-function, where $m = 4$

Definition 1. A (multiword) T-function is a map

$$\mathbf{T} : \begin{cases} (\{0, 1\}^n)^m \mapsto (\{0, 1\}^n)^m \\ \mathbf{x} \mapsto \mathbf{T}(\mathbf{x}) = (T_k(\mathbf{x}))_{k=0}^{m-1} \end{cases}$$

sending an m -tuple of n -bit words to another m -tuple of n -bit words, where each resulting n -bit word is denoted as $T_k(\mathbf{x})$, such that for each $0 \leq i < n$, the i -th bits of the resulting words $[\mathbf{T}(\mathbf{x})]_i$ are functions of just the lower input bits $[\mathbf{x}]_0, [\mathbf{x}]_1, \dots, [\mathbf{x}]_i$.

Thus, as for multiword T-functions, the i -th bit of any output word depends only on the 0th through i -th bit of each input word.

2.2 T-function Based Stream Ciphers and Their Cryptanalysis

This section introduces T-function based stream ciphers. The paper written by Klimov and Shamir [8] gave some examples of multiword T-functions. However, Mitra and Sarkar reported in 2004 that a stream cipher employing a simple output function can be broken by a time-memory trade-off attack [13].

In 2005, Hong et al. proposed a new single-cycle T-function, which uses the S-box properties, as a T-function based stream cipher. They also proposed TSC-1/2, stream ciphers that use their proposed T-function. Both of these, however,

were broken by Junod et al. at the FSE 2005 rump session. In the same year, Hong et al. proposed TSC-3, an improved algorithm of TSC at eSTREAM. Not long afterwards, though, Muller and Peyrin broke TSC-3.

Mir-1 is a stream cipher proposed at eSTREAM by Maximov, and it basically uses the multiword T-function proposed by Klimov and Shamir. Mir-1 uses a T-function that is intended to reduce the size of internal state.

This paper describes cryptanalysis against Mir-1 that exploits the T-function characteristics and the Mir-1 initialization.

3 Description of Mir-1

This section describes the structure of Mir-1, the T-function based 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.

3.1 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. $X \lll t$ denotes a t -bit rotating shift to the left of 64-bit word X . The byte unit and bit unit of 64-bit word X are set as follows, where $\parallel$ represents data concatenation.

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

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

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

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

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

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

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

$$\begin{aligned} s &= (x_0 \& x_1 \& x_2 \& x_3 + C_0) \oplus x_0 \& x_1 \& x_2 \& x_3 \\ C_0 &= 0x1248842112488421 \\ C_1 &= 0x1248124812481248 \\ C_3 &= 0x4812481248124812 \end{aligned}$$

Fig. 2. Loop state update

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. 3. 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 four registers, x_0, x_1, x_2, x_3 . Each 64-bit word is denoted as

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

Fig. 3. Automata state update

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.

3.3 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 function is shown in Fig. 4.

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

Fig. 4. Key setup

First, the key setup computes an S-box, which varies depending on the secret key value referred to as the secret S-box, using the equation shown below. Here, $SR[\cdot]$ means the S-box of Rijndael. Each entry is computed for $i = 0, \dots, 255$.

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

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 function is shown in Fig. 5.

4 Cryptanalysis of Mir-1

This section describes the method to attack the Mir-1 stream cipher. Section 4.1 explains the structural properties of the IV setup and LS update, which are necessarily exploited for the cryptanalysis, and section 4.2 describes the cryptanalysis using these properties. Section 4.3 discusses the results of an experimental attack made on Mir-1.

4.1 Properties of IV Setup and LS Update

This section describes the properties of the IV setup and LS update, on which the key setup has no particular influence.

First, we explain the structural properties of the IV setup. As shown in Fig. 5, the IV setup divides a 64-bit IV into eight 8-bit data, each of which is substituted

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

Fig. 5. IV setup

in the secret S-box to be XORed with each register. Thus, if entries of the secret S-box are unknowns, then each register is XORed with an unknown. Here, we assume that each byte inputs the same value, $IVa = (a \parallel a \parallel \dots \parallel a)$ to the IV setup. Then the data XORed with each register in the IV setup are as shown in Fig. 6.

As shown in Fig. 6, the IV has no influence over registers x_0 , x_1 , x_2 , or x_3 at step 2, regardless of the value of a . At steps 1 and 3, all the data XORed with each register becomes $S[a]$. Though entries of the secret S-box are unknown because of their dependence on the secret key, it is apparent that all the registers are XORed with the same value.

Next, we describe the LS update properties. As explained in Section 2.1, as far as multiword T-functions are concerned, the n -th bit of any output word depends only on the 0th through n -th bit of each input word. Thus, if differential Δ_i is given as the initial value of register x_i ($i = 0, 1, 2, 3$), and if all of the 0th through n -th bits of differential Δ_i are 0, then the differential of the 0th through n -th bits of register x_i is always 0, regardless of the number of times the LS update is performed.

As shown in Fig. 6, if the same IV value, $IVa = (a \parallel a \parallel \dots \parallel a)$ is input to IV setup, the IV has no influence over the 0th through 31st bits of register x_i ($i = 0, 1, 2, 3$). Consequently, while the secret key is fixed, no changes are made on the 0th through 31st bits of register x_i , regardless of the number of times the LS update is performed, even if IVa is changed.

Fig. 6. IV setup where each byte inputs the same IVa

The following section describes the cryptanalysis where these two properties are exploited.

4.2 Attack Method

This section describes an attack method that exploits the two properties described in Section 4.1. For this attack to succeed, the following preconditions must be met.

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

First, a pair of $IVa = (a \parallel a \parallel \dots \parallel a)$ and $IVb = (b \parallel b \parallel \dots \parallel b)$ is provided for the IV setup. Note that all the bytes of IVa as well as those of IVb have the same value. Because of the structural properties of the IV setup described in Section 4.1, the difference between each of the lower 32 bits of register $x_i (i = 0, 1, 2, 3)$ updated by IVa and the corresponding bits updated by IVb becomes 0. In other words, the equation given below holds true, where xa_i and xb_i respectively represent the register $x_i (i = 0, 1, 2, 3)$ updated by IVa and that updated by IVb .

$$xa_i[0, 31] = xb_i[0, 31] \quad (i = 0, 1, 2, 3) \quad (1)$$

When IVa and IVb are given, byte 4 of register x_i is XORed with $S[a]$ and $S[b]$, respectively. This is expressed by the following equations.

$$\begin{aligned} xa_i.byte_4 &= x_i.byte_4 \oplus S[a] & (i = 0, 1, 2, 3) \\ xb_i.byte_4 &= x_i.byte_4 \oplus S[b] & (i = 0, 1, 2, 3) \end{aligned}$$

Here, the entries of the secret S-box are unknowns. We can assume, though, that the equation below is satisfied:

$$S[a] \& 1 = S[b] \& 1 \quad (2)$$

If the condition of Eq. (2) is met, the relation described in Eq. (3) holds true for bit 32 of register xa_i and bit 32 of register xb_i .

$$xa_i.bit_{32} = xb_i.bit_{32} \quad (i = 0, 1, 2, 3) \quad (3)$$

Thus, if all the bytes of IVa as well as those of IVb have the same value, and if the condition of Eq. (2) is satisfied, the equation below is supported by Eqs. (1) and (3):

$$xa_i[0, 32] = xb_i[0, 32] \quad (i = 0, 1, 2, 3) \quad (4)$$

Consequently, the difference between each of the lower 33 bits of register x_i ($i = 0, 1, 2, 3$) updated by IVa and the corresponding bit updated by IVb becomes 0. As described in Section 4.1, because of the LS update properties Eq. (4) always holds true for the IV setup and the keystream generation, regardless of the number of times the LS update is performed.

Here, we consider the keystream generation, presuming that the condition of Eq. (2) is met. Figure 7 outlines the AS update for three clocks. Though the AS update uses addition in mod 2^{64} , this operation can be substituted by XOR if only the least significant bit is used for cryptanalysis. In Fig. 7, addition in mod 2^{64} is substituted by XOR, taking only the least significant bit into account. To simplify the explanation given hereafter, the two 64-bit words inserted by the LS update are represented by $x20 = (x_2[32, 63] \parallel x_0[32, 63])$, $x31 = (x_3[32, 63] \parallel x_1[32, 63])$, and data X at time t is denoted as $X^{(t)}$.

Here, we describe the method to create a distinguisher. The keystreams generated by IVa and IVb are denoted as za and zb , respectively. The data inserted by the LS update, when IVa and IVb are given, are represented by $(x20a, x31a)$ and $(x20b, x31b)$, respectively. The least significant bit at the position where the secret S-box is output at time t is then expressed as follows. Note that these equations mean $ROL29(X) = X \lll 29$.

$$\begin{aligned} \{ROL29(za^{(t-1)}) \oplus za^{(t)} \oplus x31a^{(t-1)} \oplus x20a^{(t)} \oplus za^{(t+1)}\}.bit_0 &= S[za^{(t)}] \& 1 \\ \{ROL29(zb^{(t-1)}) \oplus zb^{(t)} \oplus x31b^{(t-1)} \oplus x20b^{(t)} \oplus zb^{(t+1)}\}.bit_0 &= S[zb^{(t)}] \& 1 \end{aligned}$$

Here, Eq. (3) satisfies Eqs. (5) and (6).

$$x20a^{(t)}.bit_0 = x20b^{(t)}.bit_0 \quad (5)$$

$$x31a^{(t-1)}.bit_0 = x31b^{(t-1)}.bit_0 \quad (6)$$

Fig. 7. AS update (LSB) for three clocks

If time t satisfying

$$za^{(t)} = zb^{(t)} \quad (7)$$

is chosen for the keystreams generated by IVa and IVb , the equation described below holds true because the unknown entries of the secret S-box take the same input, resulting in the same secret S-box output.

$$S[za^{(t)}] = S[zb^{(t)}] \quad (8)$$

Thus, Eqs. (5), (6), (7), and (8) support Eq. (9):

$$\{ROL29(za^{(t-1)} \oplus zb^{(t-1)}) \oplus za^{(t+1)} \oplus zb^{(t+1)}\}.bit_0 = 0 \quad (9)$$

In summary, if any given pair of IVa and IVb satisfies Eq. (2), then Eq. (9) always holds true at the time t where Eq. (7) holds. Thus, Eq. (9) can be used as a distinguisher that distinguishes a Mir-1 output sequence from a true random sequence.

Since the probability that Eq. (2) is satisfied is the probability that the least significant bits of two randomly chosen secret S-box entries match each other, it becomes $1/2$. The probability that Eq. (7) is satisfied becomes 2^{-8} , assuming that the keystream of Mir-1 is a true random number sequence. Thus, according to Mantin and Shamir [10], when arbitrary IV pairs are chosen, the amount of data T required to distinguish the Mir-1 output sequence from a true random number sequence is theoretically determined by

$$T = (1/2)^{-2} \times 2^8 = 2^{10}$$

4.3 Experimental Results

In this section we discuss the outcome of an experimental attack like the one described in Section 4.2. Preconditions for the experimental attack are defined in Section 4.2, and the steps described below were taken to make the experimental attack.

1. Generate keystreams corresponding to $IV0 = (0 \parallel 0 \parallel \dots \parallel 0)$ and $IV1 = (1 \parallel 1 \parallel \dots \parallel 1)$.
2. Find w values of t , where t represents the time at which the least significant byte of the keystream generated by $IV0$ matches that for $IV1$. If we assume that the keystream of Mir-1 is a true random number sequence, the existence probability of t becomes 2^{-8} .
3. Check to see if the distinguisher of Eq. (9) holds true at the w values of t that satisfy the condition described in step 2.
4. If the distinguisher described in step 3 holds true for all w values of t , it is judged to be a Mir-1 keystream sequence. If there is any t for which the distinguisher does not hold true, increment each byte of $IV1$ by 1 and repeat steps 2 and 3.

Given 100 randomly generated secret keys, we made the experimental attack as described above to obtain the number of times IV was changed and the number of secret keys with which the Mir-1 output sequence was distinguished from a true random number sequence, where $w = 128$.¹ Table 1 shows the results of this attack.

Table 1. Number of times IV was changed and the number of distinguishable secret keys with which a Mir-1 output sequence was distinguished from a true random number sequence

IV Changes	Number of Secret Keys Used as a Distinguisher
0	57
1	79
2	91
3	95
4	98
5	100

Satisfying Eq. (2) means that the output sequence of Mir-1 can be distinguished from a true random number sequence. Thus, if an IV pair is given randomly, the Mir-1 output sequence must be distinguished from a true random number sequence at a probability of $1/2$. This means that as the number

¹ If $w = 128$, the probability that the distinguisher holds true accidentally is 2^{-128} .

of times the IV is changed is incremented by 1, with one-half of the remaining secret keys, the Mir-1 output sequence must be distinguished from a random number sequence. Thus, we consider the distinguisher to hold true at the probability we expected. Since the entries of the secret S-box are unknown, we cannot say that the distinguisher holds true with any given IV pair. However, as the S-box is a function of bijection, at worst it is apparent that Eq. (2) is necessarily satisfied if the IV is changed 128 times.

The attack proposed in this paper can distinguish a Mir-1 output sequence from a true random number sequence if the chosen IV pairs are provided. With about three or four distinct IV pairs, the distinguisher holds true at a probability of approximately 90%. Under the worst conditions, the distinguisher holds true if 128 distinct IV pairs are provided. We have verified that the proposed attack is applicable to Mir-1 and that it distinguishes a Mir-1 output sequence from a true random number sequence with a very small amount of data.

5 Conclusion

This paper describes the cryptanalysis of Mir-1, a new T-function based stream cipher. The IV used for stream cipher is a parameter that users can choose freely. Thus, an attack using the chosen IVs can be a threat. This paper proposes an effective distinguisher that uses the chosen IVs and the structural properties of the Mir-1 initialization. With a mere three or four chosen IV pairs, the attack method proposed in this paper distinguishes a Mir-1 output sequence from a true random sequence at a high probability. The theoretical amount of data required for the attack is no more than about 2^{10} words.

The attack method proposed in this paper makes effective use of the T-function properties. This is an effective way to attack a T-function based stream cipher. Stream ciphers based on T-functions will probably be used as an alternative to LFSR. However, designers of T-function based stream ciphers should pay attention to this vulnerability to make their ciphers resistant to such attacks.

Note that the attack proposed in this paper has not been developed into a key recovery attack. However, this paper describes the first Mir-1 cryptanalysis. The attack is strong, because it can distinguish a Mir-1 output sequence from a true random number sequence with only small amounts of data and computation needed.

Acknowledgement

The authors would like to thank Alexander Maximov and Masashi Kogiso for their useful comments.

References

1. eSTREAM, the ECRYPT Stream Cipher Project.
Available at <http://www.ecrypt.eu.org/stream/>

2. N. Courtois and W. Meier: "Algebraic Attacks on Stream Ciphers with Linear Feedback," *Advances in Cryptology - EUROCRYPT 2003*, LNCS 2656, pp.345-359, Springer Verlag, 2003.
3. J. Hong, D. Lee, Y. Yeom, and D. Han: "A New Class of Single Cycle T-functions," *Fast Software Encryption, FSE 2005*, LNCS 3557, pp.68-82, Springer Verlag, 2005.
4. J. Hong, D. Lee, Y. Yeom, and D. Han: "T-function Based Stream Cipher TSC-3," *ECRYPT Stream Cipher Project*, Report 2005/031, 2005.
5. P. Junod, S. Kunzli, and W. Meier: "Attacks against TSC," Rump Session at FSE 2005. Available at http://crypto.junod.info/rump_session_fse05.pdf
6. A. Klimov and A. Shamir: "A New Class of Invertible Mappings," *Cryptographic Hardware and Embedded Systems, CHES 2002*, LNCS 2523, pp.470-483, Springer Verlag, 2002.
7. A. Klimov and A. Shamir: "Cryptographic Applications of T-functions," *Selected Areas in Cryptography, SAC 2003*, LNCS 3006, pp.248-261, Springer Verlag, 2004.
8. A. Klimov and A. Shamir: "New Cryptographic Primitives Based on Multiword T-functions," *Fast Software Encryption, FSE 2004*, LNCS 3017, pp.1-15, Springer Verlag, 2004.
9. A. Klimov and A. Shamir: "New Applications of T-functions in Block Ciphers and Hash Functions," *Fast Software Encryption, FSE 2005*, LNCS 3557, pp.18-31, Springer Verlag, 2005.
10. I. Mantin and A. Shamir: "A Practical Attack on Broadcast RC4," *Fast Software Encryption, FSE 2001*, LNCS 2355, pp.152-164, Springer Verlag, 2001.
11. A. Maximov: "A New Stream Cipher "Mir-1"," *ECRYPT Stream Cipher Project*, Report 2005/017, 2005.
12. W. Meier and O. Staffelbach: "Fast Correlation Attacks on Certain Stream Ciphers," *Journal of Cryptology*, pp.159-176, Springer Verlag, 1989.
13. J. Mitra and P. Sarkar: "Time-memory Trade-off Attacks on Multiplications and T-functions," *Advances in Cryptology - ASIACRYPT 2004*, LNCS 3329, pp.468-482, Springer Verlag, 2004.
14. F. Muller and T. Peyrin: "Linear Cryptanalysis of TSC Stream Ciphers - Applications to the ECRYPT Proposal TSC-3," *ECRYPT Stream Cipher Project*, Report 2005/042, 2005.
15. T. Siegenthaler: "Decryption a class of stream ciphers using ciphertext only," *IEEE Transactions on Computers*, vol. C-34, no.1, pp.81-85, January 1985.