On the Security of Stream Cipher CryptMT v3 [⋆]

Haina Zhang[1], and Xiaoyun Wang[1,2]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
hnzhang.cn@gmail.com
[2] Center for Advanced Study, Tsinghua University, Beijing 100084, China
xiaoyunwang@mail.tsinghua.edu.cn

Abstract. CryptMT v3 is a stream cipher submitted to eStream project, and has entered the third evaluation phase. Any attack has not been found until now. In this paper, we mainly discuss the security of the state initialization process of CryptMT v3. For the key and IV setup function f_K, we can construct a probabilistic testing algorithm A^{f_K} with a distinguishing probability 1, which indicates that for each key K, f_K is a non-PRF. However, we have not found any non-randomness about the keystream output.

Key Words. stream cipher, IV setup, CryptMT v3, PRF, PRNG.

1 Introduction

In the state initialization of many stream ciphers, there are two inputs; a secret key K and a non-secret initialization vector IV. The IV has two main uses: provides randomized encryption, and helps in synchronizing communication between sender and receiver. As it turns out, however, incorporating an IV into a stream cipher is not an easy task. An important reason for this is that the IV is an input parameter which is partially under the control of the adversary. Giving the adversary control over an input parameter significantly increases the range of available attacks he has. During the evaluation phase of eStream project, a numbers of stream ciphers have been broken as a result of this, such as Pomaranch [2], DECIM [5], WG [6], LEX [6], TSC-4 [3, 7], and so on.

At FSE 2007, Berbain and Gilbert proposed a sufficient conditions on the security of IV dependent stream ciphers. For nearly all existing IV dependent stream ciphers, they split the keystream derivation into two following separate phases:

– **Key and IV setup:** an m-bit initial state value is derived from the key and IV. Let the key size and the IV size be k bits and n bits respectively, then

[⋆] Supported by the National Natural Science Foundation of China (NSFC Grant No.90604036) and 973 Project (No.2007CB807902).

we can consider the key and IV setup as a function generator, i.e., a function family $F = \{f_K : \{0,1\}^n \longrightarrow \{0,1\}^m\}$ indexed by a key K randomly chosen from $\{0,1\}^k$.

– **Keystream generation:** the keystream is derived from the m-bit initial state obtained in the key and IV setup phase. For that purpose, the m-bit initial state is taken as the seed input of a number generator which can be defined as a function $g : \{0,1\}^m \longrightarrow \{0,1\}^L$, where L is the length of the keystream output.

Consequently, an IV dependent stream cipher can be viewed as a composition G of a function family F and a function g, i.e., $G = \{g_K = g \circ f_K : \{0,1\}^n \longrightarrow \{0,1\}^L\}$. Furthermore, we can conclude that an IV dependent stream cipher is secure if and only if g_K is a Pseudo-Random Function (PRF). In order to guarantee that, we generally require that f_K is a PRF.

CryptMT v3 [4] is a stream cipher submitted to eStream project, and enters the third evaluation phase. Any attack has not been found until now. In this paper, we focus on the security of the state initialization process of CryptMT v3. For the key and IV setup function f_K, we can construct a probabilistic testing algorithm A^{f_K}, which allows to distinguish a randomly chosen function f_K of F from a perfect random function f^* with a distinguishing probability 1, which indicates that for each key K, f_K is not a PRF, and which maybe result in a potential attack to CryptMT v3. However, we have not found any non-randomness about the keystream output.

This paper is organized as follows. In Section 2, a brief description of CryptMT v3 is given. Section 3 presents that the key and IV setup function f_K is not a PRF. Section 4 discusses the security on the IV dependent stream cipher furthermore. Finally, we conclude the paper in Section 5.

2 A Brief Description of CryptMT v3

CryptMT v3 [4] is composed of an $\mathbb{F}_2$-linear generator of a word-size-integer sequence with a huge state space and a filter with one word-size memory, based on the accumulative integer multiplication. In this paper, we omit the details of the keystream generator.

2.1 Some Notations

Firstly, we introduce some notations for 128-bit integers. Denote X as a 128-bit integer, which is concatenation of four 32-bit registers, each of which is denoted by $X[3]$, $X[2]$, $X[1]$, $X[0]$, respectively. $X[i_3][i_2][i_1][i_0]$ represents a 128-bit integer obtained from four 32-bit integers $X[i_3]$, $X[i_2]$, $X[i_1]$, $X[i_0]$, where $(i_3 i_2 i_1 i_0)$ is a permutation of $\{3,2,1,0\}$.

128-bit Oriented Operations. In this subsection, we define some 128-bit oriented operations.

The operation '$+_{32}$' on 128-bit registers which is executed for each 32-bit integer is denoted as follows:

$$X +_{32} Y := [(X[3] + Y[3]), (X[2] + Y[2]), (X[1] + Y[1]), (X[0] + Y[0])]_{\mathrm{mod} 2^{32}},$$

here, the first 32-bit part is the addition of $X[3]$ and $Y[3]$ modulo 2^{32}, the second 32-bit part is that of $X[2]$ and $Y[2]$ (without the carry from the second 32-bit part to the first 32-bit part, differently from the addition of 128-bit integers). The outer most [ ] in the right side is to emphasize that they are concatenated to give a 128-bit integer.

The operation '$-_{32}$'is defined similarly to '$+_{32}$'.

For an integer s $(0 \leq s \leq 31)$,

$$X >>_{32} s := [(X[3] >> s), (X[2] >> s), (X[1] >> s), (X[0] >> s)]$$

means that the shift right by s bits are applied to each of the four 32-bit integers.

The product of a constant $c \in Z/(2^{32})$ multiplying a 128-bit register modulo 2^{32} can be represented as

$$c \times_{32} X := [c \times X[3], c \times X[2], c \times X[1], c \times X[0]]_{\mathrm{mod} 2^{32}},$$

and the product of two 128-bit register modulo 2^{32} is denoted as

$$X \times_{32} Y := [(X[3] \times Y[3]), (X[2] \times Y[2]), (X[1] \times Y[1]), (X[0] \times Y[0])]_{\mathrm{mod} 2^{32}}.$$

Based on the operations described above, we can define two 128-bit oriented functions as follows.

Function 1. For two 128-bit integers X and Y, we define

$$\begin{aligned} R(X,Y) = {} & 2 \times_{32} X \times_{32} (Y[1][0][2][3] \oplus (Y >>_{32} 11) \\ & +_{32} X +_{32} (Y[1][0][2][3] \oplus (Y >>_{32} 11)). \end{aligned}$$

Function 2. For three 128-bit integers X, Y and Z, we define

$$S(X,Y,Z) = (X +_{32} Y)[2][1][0][3] \oplus ((X +_{32} Y) >>_{32} 13) -_{32} Z.$$

2.2 The Key and IV Setup Function f_K

In this paper, we consider the case the sizes of key K and the initial vector IV are both 128 bits. The key and IV setup function f_K is constructed as follows.

Let $T_j (j \geq 0)$ be a 128-bit integer sequence. Then, the internal states of T_0, T_1, T_2, and T_3 are initialed as

$$\begin{aligned} T_0 &= IV, \\ T_1 &= K, \\ T_2 &= IV, \\ T_3 &= (K +_{32} Con), \end{aligned}$$

where Con is a 128-bit constant integer $(846264, 979323, 265358, 314159)_{10}$.

Denote a_i as a 128-bit integer sequence with initial state $a_0 = K|(1, 1, 1, 1)$, that is, the least significant bit of each the 32-bit integer of a_0 is set to 1. Then, the internal state T_{i+3} $(i \geq 1)$ can be transferred by the following two recursions with Function 1 and Function 2 described in subsection 2.1:

For $i = 1, 2, 3, \cdots$,

$$a_i = R(a_{i-1}, T_{i+2}), \tag{1}$$

$$T_{i+3} = S(T_{i-1}, T_{i+1}). \tag{2}$$

We call the combination of the recursions (1) and (2) as a booter.

The booter can be utilized to initialize the state of number generator. On the other hand, it can be used to generate keystream output for the message in small size case. We consider the 156 128-bit integers $T_{10}, \cdots, T_{165}$ as the initial state of the number generator, i.e., the key and IV setup function f_K is a 128-bit to 19968-bit initial state function parameterized by a 128-bit key:

$$f_K : \{0, 1\}^{128} \to \{0, 1\}^{19968}.$$

3 The Security of the Function f_K in CryptMT v3

Let us consider the function generator of CryptMT v3, i.e., a function family $F = \{f_K\}$ of $\{0, 1\}^{128} \to \{0, 1\}^{19968}$ indexed by a key K randomly chosen from $\{0, 1\}^{128}$. In this section, we try to explore the security of the function f_K.

We consider the differential with a same key K, and suppose that there is a differential bit at the k-th least significant bit position of $\varDelta IV[i] = IV[i] \oplus IV'[i]$, which is denoted as $\varDelta IV[i]_k = IV[i]_k \oplus IV'[i]_k$ $(0 \leq i \leq 3,\ 0 \leq k \leq 31)$.

Furthermore, we explore the behavior of the differential sequences

$$\varDelta a_i = a_i -_{32} a'_i = R(a_{i-1}, T_{i+2}) -_{32} R(a'_{i-1}, T'_{i+2}),$$

$$\varDelta T_{i+3} = T_{i+3} -_{32} T'_{i+3} = S(T_{i-1}, T_{i+1}, a_{i-1}) -_{32} S(T'_{i-1}, T'_{i+1}, a'_{i-1}),$$

where $i = 1, 2, \cdots$.

It is obvious that $T_0 = T_2 = IV$. In order to eliminate the differential in T_4, from the characteristic of function $S(X, Y, Z)$, we choose $\varDelta IV$ satisfying that IV and IV' are only different at the most significant bit position of $\varDelta IV[3]$. Then from the state initialization process, we can get

$$\varDelta a_0 = \varDelta T_1 = \varDelta T_3 = [0, 0, 0, 0],$$

$$\varDelta T_0 = \varDelta T_2 = [0x80000000, 0, 0, 0].$$

We call running the booter once as a round operation in the state initialization. Consequently, after the first round, we can obtain $\varDelta a_1 = [0, 0, 0, 0]$ and $\varDelta T_4 = [0, 0, 0, 0]$.

Similarly, after the second round, we get $\varDelta a_2 = [0, 0, 0, 0]$ and $\varDelta T_5 = [0, 0, 0, 0]$.

However, the differential occurs after the third round. In order to denote the 32-bit differential as a block similar to hex representation of a 32-bit integer, we adopt following signs. '0' means that the differential bit is zero, and '1' represents the differential bit is one. If a differential bit is uncertain, it is denoted as '$\cdot$', and 'x' is denoted as the combination of four successive uncertain differential bits, i.e., $[\cdot,\cdot,\cdot,\cdot]$, and '$\bar{4}$' represents the combination $[\cdot, 1, 0, 0]$. Then, "0$x$xxx$\bar{4}$0000" represents 32 successive differential bits

$$\cdot,\cdot,\cdot,\cdot,||\cdot,\cdot,\cdot,\cdot,||\cdot,\cdot,\cdot,\cdot,||\cdot 1,0,0,||0,0,0,0,||0,0,0,0,||0,0,0,0,||0,0,0,0.$$

After that, we can calculate the differentials after the third round as follows.

$$\varDelta a_3 = [0,0,0,0], \varDelta T_6 = [0x\mathrm{xxx}\bar{4}0000, 0, 0, 0x80000000].$$

In the same way, we can obtain the differentials of $\varDelta a_4$ to $\varDelta a_7$ and $\varDelta T_7$ to $\varDelta T_{10}$, and the results are listed in Table 1.

Table 1. The differential path of CryptMT v3

	3	2	1	0
$\varDelta T_0$	80000000	0	0	0
$\varDelta T_1$	0	0	0	0
$\varDelta T_2$	80000000	0	0	0
$\varDelta T_3$	0	0	0	0
$\varDelta a_0$	0	0	0	0
$\varDelta a_1$	0	0	0	0
$\varDelta T_4$	0	0	0	0
$\varDelta a_2$	0	0	0	0
$\varDelta T_5$	0	0	0	0
$\varDelta a_3$	0	0	0	0
$\varDelta T_6$	xxx$\bar{4}$0000	0	0	80000000
$\varDelta a_4$	xxxxxx80	80000000	0	xxx$\bar{4}$0000
$\varDelta T_7$	xxxxxx80	80000000	0	xxx$\bar{4}$0000
$\varDelta a_5$	xxxxxxxx	xxx$\bar{4}$0000	80000000	xxxxxx00
$\varDelta T_8$	xxxxxxxx	xxx$\bar{4}$0000	0	xxxxxx00
$\varDelta a_6$	xxxxxxxx	xxxxxx80	xxx$\bar{4}$0000	xxxxxxxx
$\varDelta T_9$	xxxxxxxx	xxxxxx80	xxx$\bar{4}$0000	xxxxxxxx
$\varDelta a_7$	xxxxxxxx	xxxxxxxx	xxxxxxx8	xxxxxxxx
$\varDelta T_{10}$	xxxxxxxx	xxxxxxxx	xxxxxxx8	xxxxxxxx

Denote the k-th least significant bit of $T[10]$ as $T[10]_k$. Then for the chosen pair (IV, IV') which satisfies that $\varDelta IV = [0x80000000, 0, 0, 0]$ with a same key, from Table 1, we can obtain that

$$\Pr(\varDelta T[10]_k = 0) = 1, \text{ for } k = 0, 1, 2.$$
$$\Pr(\varDelta T[10]_3 = 0) = 1.$$

We call the chosen IV attack to the function f_K described above as a probabilistic testing algorithm A^{f_K}.

It is clear that we have constructed a probabilistic testing algorithm A^{f_K}, and such algorithm allows to distinguish a randomly chosen function f_K from a perfect random function f^* randomly chosen in the set F^* of all $\{0,1\}^{128} \rightarrow \{0,1\}^{19968}$ functions with a distinguishing advantage 1.

Consequently, each function f_K is not a PRF, for each random key K.

4 The Security of IV Dependent Stream Ciphers Whose Function f_K is a Non-PRF

In [1], Berbain and Gilbert present a composition lemma which results in a secure construction allowing to derive a secure IV dependent stream cipher from a PRF and a PRNG (Pseudo-Random Number Generator).

Definition 1. The composition $G = g \circ F$ of an n-bit to m-bit family of functions $F = \{f_K\}$ and of an m-bit to L-bit function g is the n-bit to L-bit family of functions

$$G = \{g \circ f_K\}.$$

Lemma 1. Let us consider a PRF family $F = \{f_K\}$ where $f_K : \{0,1\}^n \rightarrow \{0,1\}^m$ and a PRNG $g : \{0,1\}^m \rightarrow \{0,1\}^L$ that produces L bits in time T_g^L. The advantage in time t with q queries of $G = g \circ F = \{g \circ f_K\}$ can be upper bounded as follows.

$$\mathbf{Adv}_G^{prf}(t,q) \leqslant \mathbf{Adv}_F^{prf}(t + qT_g^L) + q\mathbf{Adv}_g^{prng}(t + qT_g^L),$$

where $\mathbf{Adv}_F^{prf}(t + qT_g^L)$ is the maximal advantage for distinguishing the family F in time $t + qT_g^L$, and $\mathbf{Adv}_g^{prng}(t + qT_g^L)$ is the maximal advantage for distinguishing the function g in time $t + qT_g^L$.

The details can be referred in [1], and Lemma 1 provides a method to construct a security IV dependent stream cipher.

Application to the Security of IV Dependent Stream Ciphers. An IV dependent stream cipher can be considered secure if and only if the IV to keystream function $g_K = g \circ f_K$ parameterized by the key is a PRF. Lemma 1 implies that this is indeed the case, i.e., that the stream cipher is secure if:

1) the n-bit to m-bit IV to initial state function parameterized by the key representing the IV setup of a stream cipher is a PRF;
2) the m-bit to L-bit initial state to keystream function is a PRNG;
3) the upper bounds on the advantage for distinguishing $\{g_K\}$ given by Lemma 1 guarantee a sufficient resistance against attacks.

However, Lemma 1 can not be applied for the case that the function f_K is non-PRF seriously, and it is hard to prove that the IV dependent steam ciphers whose function f_K is non-PRF are provable security. In this case, the only way to estimate whether the stream ciphers are secure is to test whether the composition function $g \circ f_K$ is a PRF.

Therefore, although we can construct a probabilistic testing algorithm A^{f_K} to distinguish the function f_K with probability 1, we can not construct a probabilistic testing algorithm $B^{g \circ f_K}$ to distinguish the composition function $g \circ f_K$ so far. The security of CryptMT v3 need to be explored furthermore.

5 Conclusion

In this paper, we mainly discussed the security of the state initialization process of CryptMT v3. For the key and IV setup function f_K, we constructed a probabilistic testing algorithm A^{f_K} with a distinguishing advantage 1, which indicates that for each key K, f_K is a non-PRF. However, we have not found any non-randomness about the keystream output.

References

1. C. Berbain and H. Gilbert. On the Security of IV Dependent Stream Ciphers. In A.Biryukov, editor, FSE 2007, LNCS 4593, pp.254-273. Springer, 2007.
2. C. Cid, H. Gilbert, and T. Johansson. Cryptanalysis of Pomaranch. IEE Information Security, 153(2):51 - 53, June 2006.
3. S. Fischer, W. Meier, C. Berbain, etc. Non-randomness is eSTREAM Candidates Salsa20 and TSC-4, INDOCRYPT 2006, LNCS 4329, pp. 2 - 16. Springer, 2006.
4. M. Matsumoto, M. Saito, T. Nishimura, and M. Hagita, CRYPTMT STREAM CIPHER VERSION 3, http://www.ecrypt.eu.org/stream/p3ciphers/cryptmt/cryptmt_p3.pdf.
5. H. Wu and B. Preneel. Cryptanalysis of the stream cipher DECIM. In M. Robshaw, editor, FSE 2006, LNCS 4047, pages 30 - 40. Springer, 2006.
6. H. Wu and B. Preneel. Resynchronization attacks on WG and LEX. In M. Robshaw, editor, FSE 2006, LNCS 4047, pages 422 - 432. Springer, 2006.
7. H. Zhang, X. Wang, Differential Cryptanalysis of T-function Based Stream Cipher TSC-4, In K.-H. Nam and G. Rhee (Eds.), ICISC 2007, LNCS 4817, pp. 227 - 238. Springer, 2007.