T-function based streamcipher TSC-4

Dukjae Moon, Daesung Kwon, Daewan Han, Jooyoung Lee, Gwon Ho Ryu,
Dong Wook Lee, Yongjin Yeom, and Seongtaek Chee

National Security Research Institute
161 Gajeong-dong, Yuseong-gu
Daejeon, 305-350, Korea
{djmoon,ds_kwon,dwh,jlee05,jude,dwlee,yjyeom,chee}@etri.re.kr

Abstract. In this article, we present a synchronous stream-cipher named TSC-4, together with security analysis and implementation results. TSC-4 is designed to be well suited for constrained hardware with an intended security level of 80 bits. With 4×4 s-boxes at its core, the design leaves open the possibility for implementations of very low power consumption. As an improvement of TSC-3, TSC-4 shows better resiliency against distinguishing attacks.

Keywords: TSC-4, T-function, single cycle, streamcipher, s-box, nonlinear filter

1 Introduction

Few years ago, Klimov and Shamir started developing the theory of T-functions[1–3]. A T-function is a function acting on a collection of memory words, with a weak one-wayness property. It started out as a tool for block ciphers, but is now more of a building block for a stream cipher.

An important class of T-functions consists of those with *single cycle* property. Any T-function with single cycle property is equivalent to a LFSR of maximum length, and has potential to construct a very fast stream cipher. Unfortunately, only a small family of single cycle T-functions are known for now.

In 2004, we presented a new class of single cycle T-function[4, 5]. Although previous T-functions targeted software implementations, our T-function was designed to be light and was well suited for constrained hardware. Also, we proposed the stream cipher based on this T-function, TSC-1, TSC-2[5] and TSC-3[6]. We used the T-function to resist against the powerful attacks which are applied to the stream ciphers based on LFSR, such as algebraic attacks [10–12] and correlation attacks[8, 9] and to be possible to work out the period. However, Künzli *et al.* and Muller *et al.* described distinguishing and key recovery attacks against TSC family[13, 14]. This attack was used that our T-function did not offer a sufficient level of diffusion. In order to prevent distinguishing attacks, we modified the cipher by carefully choosing an s-box and a nonlinear function in it.

In this article, we present a synchronous stream-cipher named TSC-4 (T-function based Stream-Cipher ver 4), together with security analysis and implementation results. The main environment of the cipher is targeted to constrained

hardware with an intended security level of 80 bits. With 4×4 s-boxes at its core, the design leaves open the possibility for implementations of very low power consumption.

2 Cipher specification

In this section, we describe specifications of TSC-4, including the internal state, the cipher body and state initialization. As seen in Fig. 1, TSC-4 is a filter generator based on T-functions, whose internal state consists of two 128-bit states of T-functions. After each update, an 8-bit output keystream is produced from the states through a nonlinear filter.

2.1 Internal state of T-function

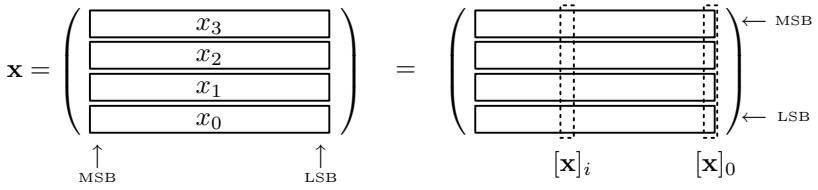
We denote a 128-bit state by

$$\mathbf{x} = (x_k)_{k=0}^{3},$$

where each word $x_k, k = 0, \ldots, 3$ has 32 bits in length. Let $[x]_i, i = 0, \ldots, n-1$ denote the i-th bit of an n-bit word x. Then the word(vector) x will interchangeably represent an integer, if necessary, by the following equation:

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

With the above notations, we can represent each internal state in a matrix form as follows:



Here $[\mathbf{x}]_i$ denotes the i-th column of state $\mathbf{x}$.

2.2 Main body

TSC-4 takes an 80-bit length secret key K and an 80-bit length public initialization vector IV. The structure of TSC-4 is illustrated in Fig. 1.

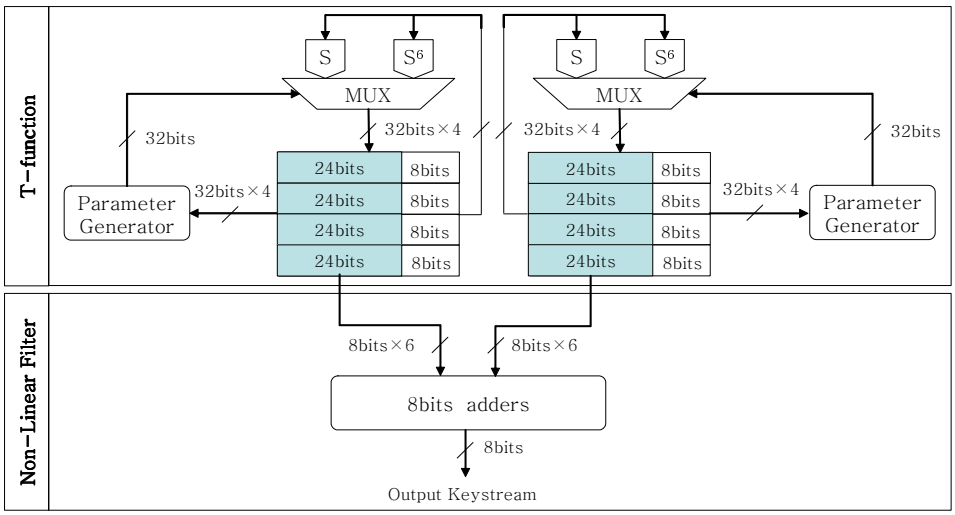


Fig. 1. The structure of TSC-4

Parameters: Two parameters $p_1(\mathbf{x})$ and $p_2(\mathbf{y})$ are defined with a number of temporary variables as follows:

$$\begin{aligned} \pi(\mathbf{x}) &= x_0 \wedge x_1 \wedge x_2 \wedge x_3, \\ o_1(\mathbf{x}) &= \pi(\mathbf{x}) \oplus (\pi(\mathbf{x}) + \texttt{0x51291089}), \\ e(\mathbf{x}) &= (x_0 + x_1 + x_2 + x_3)_{\ll 1}, \\ p_1(\mathbf{x}) &= o_1(\mathbf{x}) \oplus e(\mathbf{x}), \\ \\ \pi(\mathbf{y}) &= y_0 \wedge y_1 \wedge y_2 \wedge y_3, \\ o_2(\mathbf{y}) &= \pi(\mathbf{y}) \oplus (\pi(\mathbf{y}) + \texttt{0x12910895}), \\ e(\mathbf{y}) &= (y_0 + y_1 + y_2 + y_3)_{\ll 1}, \\ p_2(\mathbf{y}) &= o_2(\mathbf{y}) \oplus e(\mathbf{y}), \end{aligned} \tag{2}$$

where $\wedge$, $\oplus$ and $\ll$ denote bitwise `AND`, bitwise `XOR` operation, and left shift of 32-bit words, respectively. The additions are done modulo 2^{32} using the equation (1). Note that $o_i, i = 1, 2$ are *odd parameters* and e is an *even parameter* [5].

S-box application: We fix a 4×4 s-box S, defined in C-language style as follows:

$$S\texttt{[16] = \{9,2,11,15,3,0,14,4,10,13,12,5,6,8,7,1\};} \tag{3}$$

Now T-functions $\mathbf{T}_i, i = 1, 2$ on input states $\mathbf{x}$, $\mathbf{y}$ are defined as follows:

$$[\mathbf{T}_1(\mathbf{x})]_i = \begin{cases} S\ ([\mathbf{x}]_i) & \text{if } [\mathbf{p}_1(\mathbf{x})]_i = 1, \\ S^6([\mathbf{x}]_i) & \text{if } [\mathbf{p}_1(\mathbf{x})]_i = 0, \end{cases} \tag{4}$$

$$[\mathbf{T}_2(\mathbf{y})]_i = \begin{cases} S\ ([\mathbf{y}]_i) & \text{if } [\mathbf{p}_2(\mathbf{y})]_i = 1, \\ S^6([\mathbf{y}]_i) & \text{if } [\mathbf{p}_2(\mathbf{y})]_i = 0, \end{cases} \tag{5}$$

where the columns $[\mathbf{x}]_i$, $[\mathbf{T}_1(\mathbf{x})]_i$, $[\mathbf{y}]_i$ and $[\mathbf{T}_2(\mathbf{y})]_i$ are regarded as 4-bit integers by the equation (1).

Nonlinear filter: The filter produces the actual output keystream from the current internal states. We compute six 8-bit temporary variables $(a_0, \cdots, a_5)$ as follows:

$$\begin{aligned} a_0 &= ((x_3)_{\gg 24} \ \wedge\ \texttt{0xff}) + ((y_1)_{\gg 8} \ \wedge\ \texttt{0xff}), \\ a_1 &= ((x_0)_{\gg 24} \ \wedge\ \texttt{0xff}) + ((y_2)_{\gg 8} \ \wedge\ \texttt{0xff}), \\ a_2 &= ((x_2)_{\gg 16} \ \wedge\ \texttt{0xff}) + ((y_3)_{\gg 16} \ \wedge\ \texttt{0xff}), \\ a_3 &= ((x_1)_{\gg 16} \ \wedge\ \texttt{0xff}) + ((y_0)_{\gg 16} \ \wedge\ \texttt{0xff}), \\ a_4 &= ((x_3)_{\gg 8} \ \wedge\ \texttt{0xff}) + ((y_2)_{\gg 24} \ \wedge\ \texttt{0xff}), \\ a_5 &= ((x_0)_{\gg 8} \ \wedge\ \texttt{0xff}) + ((y_1)_{\gg 24} \ \wedge\ \texttt{0xff}), \end{aligned} \tag{6}$$

where the additions are done modulo 2^8. Now the 8-bit keystream z is defined to be

$$z = a_0 \ \oplus\ (a_1)_{\ggg 5} \ \oplus\ (a_2)_{\ggg 2} \ \oplus\ (a_3)_{\ggg 5} \ \oplus\ (a_4)_{\ggg 6} \ \oplus\ (a_5)_{\ggg 2}, \tag{7}$$

where $\ggg$ denote rotation to the right.

2.3 State initialization

We now describe how the state is initialized from a given key and an IV. The internal state consists of 8 words as seen in Fig. 2.

$$\mathbf{x} = \begin{pmatrix} \boxed{x_3} \\ \boxed{x_2} \\ \boxed{x_1} \\ \boxed{x_0} \end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix} \boxed{y_3} \\ \boxed{y_2} \\ \boxed{y_1} \\ \boxed{y_0} \end{pmatrix}$$

Fig. 2. Internal state of TSC-4

Key/IV Loading: Let $K = (k_{79}, k_{78}, \cdots, k_1, k_0)$ and $IV = (iv_{79}, iv_{78}, \cdots, iv_1, iv_0)$ be an 80-bit key and an 80-bit IV, respectively. Then the internal state is initialized as follows:

1. $x_0 = (k_{31}, k_{30}, \cdots, k_1, k_0)$
2. $x_1 = (k_{63}, k_{62}, \cdots, k_{33}, k_{32})$
3. $x_2 = (iv_{31}, iv_{30}, \cdots, iv_1, iv_0)$
4. $x_3 = (iv_{63}, iv_{62}, \cdots, iv_{33}, iv_{32})$
5. $y_0 = (iv_{15}, \cdots, iv_0, iv_{79}, \cdots, iv_{64})$
6. $y_1 = (iv_{47}, iv_{46}, \cdots, iv_{17}, iv_{16})$
7. $y_2 = (k_{15}, \cdots, k_0, k_{79}, \cdots, k_{64})$
8. $y_3 = (k_{47}, k_{46}, \cdots, k_{17}, k_{16})$

Warm-up: Once the internal state is initialized, the K and IV are mixed by the following process.

1. Run cipher body once to produce a single 8-bit output.
2. Rotate x_1 and y_0 to the left by 8 bits.
3. `XOR` the output to the least significant 8 bits of x_1 and y_0.

The key and IV setup is completed by repeating the above three steps by eight times.

3 Security

TSC-4 is intended for 80-bit security. For the moment, the best attack on TSC-4 we know of is the brute force attack of complexity 2^{80}.

3.1 Statistical tests

We have done tests similar to the ones presented in [7] and have verified that this proposal gives good statistical results.

3.2 Period

The period of TSC-4 is 2^{128}. To see this, we already know that the period of each T-function is 2^{128}, as guaranteed by the single cycle property [5]. So, first note that the period of TSC-4 has to be a divisor of 2^{128}. Now, initialize two register contents with the all zero state and consider what each content of the registers would be after 2^{124} iterated applications of the T-function. Since the period of each T-function restricted to the lower 31 columns is 2^{124}, all columns except the most significant column should be zero. Now we can show that there exists a nonzero bit in the output 8-bit keystream, since the most significant columns determine the i-th output bit for i=1, 2, 5, 7. Furthermore, when observed every 2^{124} iterations apart, due to description (4) and (5) and the definition of an odd

parameter, the change of the most significant columns follow some fixed odd power of the S-box, which is of cycle length 16. Explicit calculation of the 16 keystream output words for each odd power of the s-box confirms that, in all odd power cases, one has to go through all 16 points before reaching the starting point. Hence the period of the cipher is $16 \cdot 2^{124} = 2^{128}$.

3.3 Correlation attack

Difficulty of correlation attacks can also be obtained from the rotations in the filter. In the last step of a correlation attack, one needs to guess a part of the state and compare calculated outputs with the actual keystream, checking for the occurrence of expected correlation.

In our situation, any correlation found to exist with a single output bit will involve multiple input bits. Hence correlation attacks do not seem to be applicable.

3.4 Algebraic attack

In many cases, algebraic attacks are possible on stream ciphers built on LFSRs. Once a single equation connecting the internal state to the output keystream is worked out, the cipher logic can be run forward to produce more such equations. During this process, the linear property of LFSRs keep the degree of new equations equal to the first equation. And this is the main reason for the success of algebraic attacks on streamciphers.

In the case of TSC-4, the source of randomness, i.e., the T-function, is already nonlinear. During the action of T-functions $\mathbf{T}_1$ and $\mathbf{T}_2$ on internal states $\mathbf{x}$ and $\mathbf{y}$, the degree of new equation increase in the degree of a previous equation. Hence algebraic attacks do not seem to be applicable.

3.5 Guess-then-determine attack

One property of T-functions, that could be bad from the viewpoint of security, is that it can be restricted to any number of its lower columns. In other words a part of internal state of T-function can be guessed and run forward indefinitely, opening up the possibility of a guess-then-determine attack.

The rotations used in the filter eliminates this weakness. They have been chosen so that any single output bit receives direct effect of more twelve bits that are spread widely apart within two states. So it is not possible to calculate any output bit with the information of any small number of internal states.

Even if all modular additions in the filter were replaced with XORs, in order to calculate any one of the 8 output bits continuously, one would need to guess 96 bits (8×12 bits), so no meaningful attack can be achieved through this approach.

3.6 Distinguishing attack

Bit-flip probability: We have chosen the s-box (3) to satisfy the following conditions.

1. At the application of S, each of the four bits has bit-flip probability of $\frac{1}{2}$.
2. The same is true for S^6.

More precisely, the first condition states that

$$\#\{\, 0 \le t < 16 \mid \text{the } k\text{-th bit of } t \oplus S(t) \text{ is } 1\} = 8,$$

for each $k = 0, 1, 2, 3$. Due to this property, regardless of the behavior of the odd parameters $\mathbf{p}_1(\mathbf{x})$ and $\mathbf{p}_2(\mathbf{y})$, every bit in the state is guaranteed to have bit-flip probability $\frac{1}{2}$ at the action of $\mathbf{T}$.

Bit-flip bias of multiple applications of T-function: There are strong distinguishing attacks[13, 14] applicable to previous versions[5, 6] of this cipher. The main observation used in the attack is that even though the bit-flip probability of T-function is close to $\frac{1}{2}$, this is not true for its multiple applications. This property is still present in the current design. However, TSC-4 is designed to be resistant to the distinguishing attacks by taking the following cases into account:

Case 1 The strongest bit-flip bias between the same bit position for multiple applications. The algorithms TSC-1 and TSC-2[5] are analyzed using this property[13, 14]. In this case, we deal with the bias of $[z]_i^t \oplus [z]_i^{t+\delta}$, where δ is the number of iterations of T-function.

Case 2 The strongest bit-flip bias between the distinct bit position in the same column for multiple applications. The algorithm TSC-3[6] is analyzed using this property[14]. In this case, we deal with the bias of $[z]_i^t \oplus [z]_j^{t+\delta}, i \neq j$.

Case 3 The strongest bit-flip bias between the linear relations of the same bits for multiple applications. This property is considered in this paper. In this case, we deal with the bias of $[z]_i^t \oplus [z]_j^t \oplus [z]_j^{t+\delta} \oplus [z]_j^{t+\delta}, i \neq j$.

Table 1. Bit-flip bias of $[x_k]_i^t = [x_k]_i^{t+\delta}$ $(1 \le \delta \le 15)$

δ	1	2	3	4	5	6	7	8
$\lvert log_2\varepsilon \rvert$	∞	∞	**5**	6	7	6	∞	8.42
δ	9	10	11	12	13	14	15	$\cdots$
$\lvert log_2\varepsilon \rvert$	9.42	7.42	13	9.91	6.25	7.94	10.71	$\cdots$

First of all, we could obtain the property that a bit-flip bias between the same bit positions for δ $(1 \le \delta \le 1000)$ iterations of T-function is less than 2^{-5} through the experiments (Fig. 3). The pattern of the plot in Fig. 3 suggests that the property holds for $\delta > 1000$ iterations. Table 1 shows the exact bit-flip bias

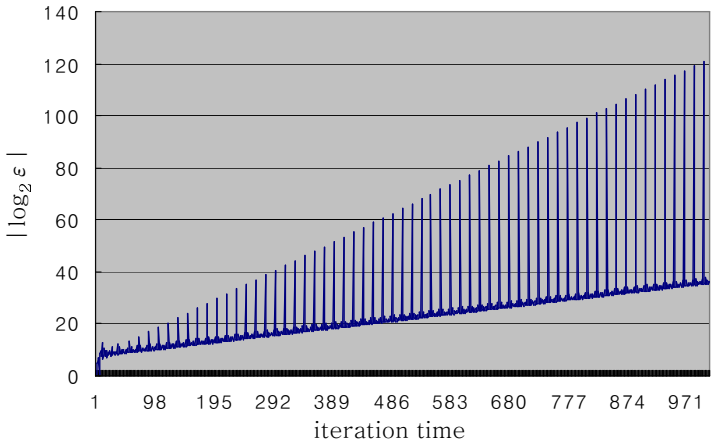


Fig. 3. Bit-flip bias of $[x_k]_i^t = [x_k]_i^{t+\delta}$ $(1 \leq \delta \leq 1000)$

"ε"[1] between the same bit positions after δ $(1 \leq \delta \leq 15)$ times iteration, where $\mathbf{x}^{t+\delta}$ denote $\mathbf{T}^{\delta}(\mathbf{x}^t)$. By using the nonlinear filter, we can obtain a linear relation of the output filter like this $(i = 0, \cdots, 7)$:

$$\begin{aligned}[z]_i^t \oplus [z]_i^{t+\delta} = &([a_0]_i^t \oplus [a_0]_i^{t+\delta}) \oplus ([a_1]_{i+5(\bmod 8)}^t \oplus [a_1]_{i+5(\bmod 8)}^{t+\delta}) \\ &\oplus ([a_2]_{i+2(\bmod 8)}^t \oplus [a_2]_{i+2(\bmod 8)}^{t+\delta}) \oplus ([a_3]_{i+5(\bmod 8)}^t \oplus [a_3]_{i+5(\bmod 8)}^{t+\delta}) \\ &\oplus ([a_4]_{i+6(\bmod 8)}^t \oplus [a_4]_{i+6(\bmod 8)}^{t+\delta}) \oplus ([a_5]_{i+2(\bmod 8)}^t \oplus [a_5]_{i+2(\bmod 8)}^{t+\delta}).\end{aligned}$$

In this relation, each $[a_k]_i^t \oplus [a_k]_i^{t+\delta}$ $(k = 0, \cdots, 5)$ is approximated as a linear relation like this:

$$\begin{aligned}
[a_0]_i^t \oplus [a_0]_i^{t+\delta} &= [x_3]_{i+24}^t \oplus [x_3]_{i+24}^{t+\delta} \oplus [y_1]_{i+8}^t \oplus [y_1]_{i+8}^{t+\delta} \oplus R_0(i), \\
[a_1]_{i+5(\bmod 8)}^t \oplus [a_1]_{i+5(\bmod 8)}^{t+\delta} &= [x_0]_{i+24}^t \oplus [x_0]_{i+24}^{t+\delta} \oplus [y_2]_{i+8}^t \oplus [y_2]_{i+8}^{t+\delta} \oplus R_1(i), \\
[a_2]_{i+2(\bmod 8)}^t \oplus [a_2]_{i+2(\bmod 8)}^{t+\delta} &= [x_2]_{i+16}^t \oplus [x_2]_{i+16}^{t+\delta} \oplus [y_3]_{i+16}^t \oplus [y_3]_{i+16}^{t+\delta} \oplus R_2(i), \\
[a_3]_{i+5(\bmod 8)}^t \oplus [a_3]_{i+5(\bmod 8)}^{t+\delta} &= [x_1]_{i+16}^t \oplus [x_1]_{i+16}^{t+\delta} \oplus [y_0]_{i+16}^t \oplus [y_0]_{i+16}^{t+\delta} \oplus R_3(i), \\
[a_4]_{i+6(\bmod 8)}^t \oplus [a_4]_{i+6(\bmod 8)}^{t+\delta} &= [x_3]_{i+8}^t \oplus [x_3]_{i+8}^{t+\delta} \oplus [y_2]_{i+24}^t \oplus [y_2]_{i+24}^{t+\delta} \oplus R_4(i), \\
[a_5]_{i+2(\bmod 8)}^t \oplus [a_5]_{i+2(\bmod 8)}^{t+\delta} &= [x_0]_{i+8}^t \oplus [x_0]_{i+8}^{t+\delta} \oplus [y_1]_{i+24}^t \oplus [y_1]_{i+24}^{t+\delta} \oplus R_5(i),
\end{aligned}$$

where $R_k(i)$ $(k = 0, \cdots, 5)$ represents the carry bit. By using the above linear approximation, we have a plausible argument that show the bit-flip bias of filter output to be much less than $2^{-49} (= 2^{-1} \times (2^{-4})^{12})$. The bit-flip bias is approximated using the *Piling-up Lemma* in case of $\delta = 3$. In order to detect this bias, data size of more than 2^{98} is needed.

[1] If $\varepsilon = 0$ then we represent $|log_2 \varepsilon|$ as "∞"

Table 2. Bit-flip bias of $[x_k]_i^t = [x_{k'}]_i^{t+\delta}$ ($|\ log_2\varepsilon|$)

case $\delta = 1$

input \ output	$[x_0]_i^{t+1}$	$[x_1]_i^{t+1}$	$[x_2]_i^{t+1}$	$[x_3]_i^{t+1}$
$[x_0]_i^t$	∞	4	4	∞
$[x_1]_i^t$	3	∞	∞	4
$[x_2]_i^t$	∞	3	∞	4
$[x_3]_i^t$	4	∞	3	∞

case $\delta = 2$

input \ output	$[x_0]_i^{t+2}$	$[x_2]_i^{t+2}$	$[x_2]_i^{t+2}$	$[x_3]_i^{t+2}$
$[x_0]_i^t$	∞	∞	∞	∞
$[x_1]_i^t$	∞	∞	∞	∞
$[x_2]_i^t$	∞	∞	∞	∞
$[x_3]_i^t$	∞	∞	∞	∞

case $\delta = 3$

input \ output	$[x_0]_i^{t+3}$	$[x_1]_i^{t+3}$	$[x_2]_i^{t+3}$	$[x_3]_i^{t+3}$
$[x_0]_i^t$	5	∞	5	5
$[x_1]_i^t$	∞	5	6	5
$[x_2]_i^t$	5	6	5	∞
$[x_3]_i^t$	5	5	∞	5

case $\delta = 4$

input \ output	$[x_0]_i^{t+4}$	$[x_1]_i^{t+4}$	$[x_2]_i^{t+4}$	$[x_3]_i^{t+4}$
$[x_0]_i^t$	6	5	4	∞
$[x_1]_i^t$	6	6	∞	4
$[x_2]_i^t$	∞	5	6	5
$[x_3]_i^t$	4	∞	6	6

case $\delta = 5$

input \ output	$[x_0]_i^{t+5}$	$[x_1]_i^{t+5}$	$[x_2]_i^{t+5}$	$[x_3]_i^{t+5}$
$[x_0]_i^t$	7	4.6	5.4	8
$[x_1]_i^t$	5.6	7	6.8	5.4
$[x_2]_i^t$	8	6.5	7	4.6
$[x_3]_i^t$	5.4	8	6	7

The second, we observe a certain pair of distinct bit positions in the same column yields a bit-flip bias worse than any bias between the same bit positions, as seen in Table 2. These pairs with this property are like this:

The pair (x_0, x_1)**:** The bit-flip bias of $[x_0]_i^t = [x_1]_i^{t+1}$ is 2^{-4} and the bit-flip bias of $[x_1]_i^t = [x_0]_i^{t+1}$ is 2^{-3}.
The pair (x_2, x_3)**:** The bit-flip bias of $[x_2]_i^t = [x_3]_i^{t+1}$ is 2^{-4} and the bit-flip bias of $[x_3]_i^t = [x_2]_i^{t+1}$ is 2^{-3}.
The other pair: At least one case of the bit-flip bias is "0". For example, the bit-flip bias of $[x_0]_i^t = [x_3]_i^{t+1}$ is "0", the bit-flip bias of $[x_3]_i^t = [x_0]_i^{t+1}$ is 2^{-4}.

By using the property, we remove the nonlinear filter from relation of the pair (x_0, x_1), (x_2, x_3). The nonlinear filter of TSC-4 is carefully chosen such that its linear approximation contains the minimum number of pairs whose bit-flip bias is less than 2^{-5}.

Finally, we check the bit-flip bias between the linear relations of the same bits for multiple applications. Those linear relations are as follows:

1. $[x_0]_i^t \oplus [x_1]_i^t = [x_0]_i^{t+\delta} \oplus [x_1]_i^{t+\delta}$, $[x_2]_i^t \oplus [x_3]_i^t = [x_2]_i^{t+\delta} \oplus [x_3]_i^{t+\delta}$.
2. $[x_0]_i^t \oplus [x_2]_i^t = [x_0]_i^{t+\delta} \oplus [x_2]_i^{t+\delta}$, $[x_1]_i^t \oplus [x_3]_i^t = [x_1]_i^{t+\delta} \oplus [x_3]_i^{t+\delta}$.
3. $[x_0]_i^t \oplus [x_3]_i^t = [x_0]_i^{t+\delta} \oplus [x_3]_i^{t+\delta}$, $[x_1]_i^t \oplus [x_2]_i^t = [x_1]_i^{t+\delta} \oplus [x_2]_i^{t+\delta}$.

Since the first relation is removed in the nonlinear filter, we consider other two relations. Table 3 shows the bit-flip biases for each case.

Table 3. Bit-flip bias of $[x_k]_i^t \oplus [x_{k'}]_i^t = [x_k]_i^{t+\delta} \oplus [x_{k'}]_i^{t+\delta}$ ($|\, log_2 \varepsilon|$)

δ	$(k,k') = (0,2)$	$(k,k') = (1,3)$	$(k,k') = (0,3)$	$(k,k') = (1,2)$
1	**2.4150**	**2.4150**	2.4150	∞
2	∞	∞	∞	3.0000
3	∞	∞	∞	3.4150
4	∞	∞	∞	2.6781
5	3.7521	3.7521	3.7521	5.6781
6	3.4150	3.4150	**3.4150**	**2.1926**
7	4.9556	4.9556	4.9556	2.6163
8	4.3561	4.3561	4.3561	4.3561
9	8.5406	8.5406	8.5406	2.9860
10	9.4150	9.4150	9.4150	3.0170
11	4.8707	4.8707	4.8707	3.8401
12	7.2996	7.2996	7.2996	3.1703
13	3.7527	3.7527	3.7527	2.3618
14	5.3276	5.3276	5.3276	4.9125
15	5.7574	5.7574	5.7574	3.3714
16	8.3927	8.3927	8.3927	3.0438
⋮	⋮	⋮	⋮	⋮

Combining the two relation $(k, k') = (0, 2)$ and $(k, k') = (1, 3)$ in $\delta = 1$, we get the maximum bit-flip bias of this relation as $2^{-3.83}(= 2^{-1} \times (2^{-1.415})^2)$. Similarly, In case of $(k, k') = (0, 3)$ and $(k, k') = (1, 2)$, the maximum bit-flip bias is $2^{-4.6076}(= 2^{-1} \times 2^{-2.415} \times 2^{-1.1926})$ in $\delta = 6$. So, we use the relation of the pair $(x_0, x_3),\ (x_1, x_2)$ in the nonlinear filter.

Therefore, we can assume that the distinguishing attack is not applicable to the algorithm TSC-4.

3.7 Time-memory trade-off

We analyze the security of TSC-4 against time-memory-data(TMD) tradeoffs presented in [18, 19]. Then, it guarantees the security against two well-known TMD tradeoffs [15–17].

Simple case[18]: Since TSC-4 takes 80-bit key with 80-bit IV, Search space of an attacker is the entropy space of size $N = 2^k (k = 160)$. The cost of TMD attacks is $O(2^{k/2})$. So, TMD attacks are expected to have complexity not lower than $O(2^{80})$.

Sampling case[19]: Since TSC-4 takes 256-bit internal state and we can find the set of all 256-bit keystream segments which starts with 8 zeros, search space of an attacker is the entropy space of size $N = 2^k (k = 248)$. The cost of TMD attacks is $O(2^{k/2})$. So, TMD attacks are expected to have complexity not lower than $O(2^{124})$.

3.8 State initialization

We consider security issues related to key setup in this section. Our state retains 160-bit entropy after state initialization.

Entropy loss: Let us consider the question of whether our state initialization process allows every possible 160-bit state to occur with equal possibility. This question is closely related to whether each step of the rekeying process is invertible. Checking all the steps of Key/IV Loading and warm-up presented in Section 2.3, we can see that all step is invertible. So, the states produced through our state initialization process has exactly 160-bit entropy. Therefore no equivalent keys are present.

Statistical property: For a good state initialization process, we would expect one bit difference in key or IV to result in about half the state bits changing. We did some basic experiments to verify this on our warm-up process.

4 Implementation

4.1 Hardware Implementation

TSC-4 consists of two T-functions and a nonlinear filter. In hardware implementation, critical path is an even parameter of a T-function, and 4×4 s-boxes are components which requires large area. In updating internal states, s-box is applied to all 64 columns.

In normal hardware design, one implement 64 s-boxes to maximize the throughput. On the other hand, we can reduce the area by implementing one s-box for each T-function, or by implementing one T-function instead of two.

Let Type A, Type B, Type C denote normal implementation, implementation with one s-box for each T-function, implementation with one T-function and one s-box respectively.

In Table 4 we summarize hardware figures when the implementation was simulated on ASIC using Samsung 0.13μm library.

Table 4. Hardware related figures for TSC-4

Type	State Initialization	Gate Count	Max. Clock /Throughput	Throughput/Power drain (100KHz clock)
A	X	10510	100MHz/800Mbps	800kbps/11.86μW
A	O	11878	100MHz/800Mbps	800kbps/12.78μW
B	X	3100	250MHz/62.5Mbps	25kbps/4.65μW
B	O	4027	198MHz/49.5Mbps	25kbps/5.52μW
C	X	3026	230MHz/28.75Mbps	12.5kbps/4.51μW
C	O	3958	198MHz/24.75Mbps	12.5kbps/5.50μW

4.2 Software Implementation

Our C-language implementation (not optimized) of TSC-4 shows the following performance.

machine	Pentium-IV 2.4GHz, 1GB RAM
OS	Windows XP (SP1)
compiler	Microsoft Visual C++ 6.0
encryption	150 cycles/byte

5 Conclusion

A synchronous streamcipher TSC-4 of 80-bit intended security level was presented with some security analysis and hardware related figures. As a result,

we failed to find an attack which is better than exhaust key search. The cipher is suitable for constrained hardware environments, allowing for a wide range of implementation choices.

References

1. A. Klimov and A. Shamir, A new class of invertible mappings. *CHES 2002*, LNCS 2523, Springer-Verlag, pp.470–483, 2003.
2. A. Klimov and A. Shamir, Cryptographic application of T-functions. *SAC 2003*, LNCS 3006, Springer-Verlag, pp.248–261, 2004.
3. A. Klimov and A. Shamir, New cryptographic primitives based on multiword T-functions. *FSE 2004*, LNCS 3017, Springer-Verlag, pp.1–15, 2004.
4. J. Hong, D. H. Lee, Y. Yeom, and D. Han, A new class of single cycle T-functions and a stream cipher proposal. *SASC*(State of the Art of Stream Ciphers, Brugge, Belgium, Oct. 2004) workshop record.
5. J. Hong, D. H. Lee, Y. Yeom, and D. Han, New class of single cycle T-functions. *FSE 2005*, LNCS 3557, pp.68–82, Springer-Verlag, 2005.
6. J. Hong, D. H. Lee, Y. Yeom, D. Han, S. Chee, T-function based streamcipher TSC-3. *SKEW*(Symmetric Key Encryption Workshop), Available from `http://www.cosic.esat.kuleuven.ac.be/ecrypt/stream/`, 2005.
7. NIST. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22.
8. F. Jonsson and T. Johansson, *A Fast Correlation Attack on LILI-128*, Information Processing Letters Vol 81, No. 3, 2001, pp.127-132.
9. W. Meier and O. Staffelbach, *Fast correlation attacks on certain stream ciphers*, J. Cryptology Vol 1, 1989, 159-176.
10. F. Armknecht and M. Krause, Algebraic attacks on combiners with memory, *Crypto 2003*, LNCS 2729, Springer-Verlag, pp.162–175, 2003.
11. N. Courtois and W. Meier, Algebraic attacks on stream ciphers with linear feedback, *Eurocrypt 2003*, LNCS 2656, Springer-Verlag, pp.345–359, 2003.
12. N. Courtois, Fast algebraic attack on stream ciphers with linear feedback, *Crypto 2003*, LNCS 2729, Springer-Verlag, pp. 176–194, 2003.
13. S. Künzli, P. Junod, and W. Meier, Distinguishing attacks on T-functions. *Mycrypt 2005,* LNCS 3715, pp. 2–15, Springer-Verlag, 2005.
14. F. Muller and T. Peyrin, Linear Cryptanalysis of TSC Stream Ciphers - Applications to the ECRYPT proposal TSC-3. *SKEW* Available from `http://www.cosic.esat.kuleuven.ac.be/ecrypt/stream/`, 2005.
15. S. H. Babbage, Improved exhaustive search attacks on stream ciphers. *European Convention on Security and Detection,* IEE Conference publication No. 408, pp. 161–166, IEE, 1995.
16. A. Biryukov and A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers. *Asiacrypt 2000,* LNCS 1976, pp. 1–13, Springer-Verlag, 2000.
17. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher. *Eurocrypt'97,* LNCS 1233, pp. 239–255, Springer-Verlag, 1997.
18. J. Hong and P. Sarkar, New Applications of Time Memory Data Tradeoffs. *Asiacrypt 2005,* LNCS 3788, pp. 353–372, Springer-Verlag, 2005.
19. J. Hong and W. Kim, TMD-Tradeoff and State Entropy Loss Considerations of Streamcipher MICKEY. *Indocrypt 2005,* LNCS 3797, pp. 169–182, Springer-Verlag, 2005.