Distinguishing Attacks on T-functions

Simon Künzli[1], Pascal Junod[2], and Willi Meier[1]

[1] FH Aargau, 5210 Windisch, Switzerland,
[2] Nagravision SA (Kudelski Group), 1033 Cheseaux, Switzerland

Abstract. Klimov and Shamir proposed a new class of simple cryptographic primitives named T-functions. For two concrete proposals based on the squaring operation, a single word T-function and a previously unbroken multi-word T-function with a 256-bit state, we describe an efficient distinguishing attack having a 2^{32} data complexity. Furthermore, Hong *et al.* recently proposed two fully specified stream ciphers, consisting of multi-word T-functions with 128-bit states and filtering functions. We describe distinguishing attacks having a 2^{22} and a 2^{34} data complexity, respectively. The attacks have been implemented.

Key words Stream cipher, T-function, square mapping, distinguishing attack, statistical cryptanalysis

1 Introduction

Binary additive stream ciphers encrypt a plaintext stream by combining it with a key stream by means of an XOR operation (the decryption simply being the XOR of the key stream with the ciphertext stream). The key stream consists of a pseudo-random bit sequence usually generated by iteration of an *update function*, the latter being initialized with a secret state. One expects that the sequence generated by a cryptographically secure stream cipher is statistically indistinguishable from a truly random sequence (and this for any adversary with some limited computational power), and that there exists no key-recovery attack better than brute-force.

Recently, Klimov and Shamir [7, 8, 9, 10, 6] proposed a new framework for highly efficient mappings which could be used as primitives in stream ciphers and other cryptographic schemes. These primitives consist of *triangular functions* (T-functions) which are built with help of fast arithmetic and Boolean operations widely available on high-end microprocessors or on dedicated hardware implementations; these mappings come with provable properties such as invertibility and a single-cycle structure. As an example, the mapping TF-0 is proposed in [7], which is defined by $x \mapsto x + (x^2 \vee C) \bmod 2^n$ for an n-bit state x and with $C \equiv 5, 7 \pmod 8$. As the maximal length of a cycle may be too short for typical values of n (e.g. $n = 64$), and as state-recovery attacks have been described [2, 8], TF-0 is not meant to be directly used for cryptographic purposes. Considering cryptographic applications, several efficient multi-word T-functions are proposed in [9]. Some of these proposals have been broken by Mitra and Sarkar [13] using time-memory tradeoffs. Based on the results of Klimov and Shamir, a new class of multi-word T-functions and two fully specified stream ciphers have been proposed by Hong *et al.* [3,4]. Their schemes TSC-1 and TSC-2 have a transparent design and allow for some flexibility.

1.1 Contributions of this Paper

In this paper, we analyse several proposals of T-functions and exhibit substantial weaknesses in some of these constructions. The flaws are extended to dedicated attacks. First we analyse the statistical properties of the pure square mapping, which allows us to find an efficient distinguisher (with an expected 2^{32} data complexity) on TF-0 as well as on a previously unbroken multi-word mapping described in [9] and labeled here as TF-0m, both based on the squaring operation. TF-0m operates on a 256-bit state and the output sequence consists of the 32 most significant bits. Then, we cryptanalyse the TSC-family of stream ciphers [4], which operates on a 128-bit state and outputs 32 bits of the state using a filtering function. We find a very efficient distinguisher for TSC-1 with an expected 2^{22} data complexity; for TSC-2, we describe a different distinguishing attack with an expected 2^{34} data complexity. To confirm our theoretical results, the distinguishing attacks have been implemented and run many times with success. Our distinguishers have a negligible error probability and a remarkably small time complexity.

1.2 Notational Conventions

We analyse cryptographic schemes consisting of an internal state $x \in \mathcal{X}$, an update function $\mathsf{f} : \mathcal{X} \to \mathcal{X}$ and an output function $\mathsf{g} : \mathcal{X} \to \mathcal{Y}$. In the case where time instants are relevant, we will denote x^t the state at time t (distinction of powers will be clear from the context). Hence, the iterative scheme maps the state x^t to $x^{t+1} = \mathsf{f}(x^t)$ and outputs $y^t = \mathsf{g}(x^t)$. The seed of the iteration is obtained from the secret key with help of a key scheduling process. The keystream K consists in the concatenation of successive outputs, namely $K = y^0||y^1||\cdots$. We assume throughout this paper the threat model of a known-plaintext attack, i.e., we assume to know some part of the keystream K. Our purpose is then to distinguish K from a uniformly distributed random sequence, or to recover the state at any time. In the case where the state is a vector formed by some words, we will denote a single word by x_j and the state as $x = (x_0, x_1, \ldots)$. Adopting the common notation, $[x]_i$ is the $(i+1)$-st least significant bit-slice of the state, $[x]_0$ denoting the rightmost bit-slice. Consequently, $[x_j]_i$ is the $(i+1)$-st least significant bit of word j. The operation $\mathrm{msb}_m(x)$ states for the m most significant bits of x. Arithmetic operations are performed modulo 2^n with typical word size $n = 32$ or 64 bit. Boolean operations are performed on all n bits in parallel and are denoted by $\wedge$ (AND), $\vee$ (OR), and by $\oplus$ (XOR). Finally, $\lll k$ denotes a cyclic left shift by k positions.

2 Cryptanalysis of Square Mappings

Klimov and Shamir have proposed different types of T-functions based on the squaring operation [7, 9]. After introducing the framework of this section, we focus on the pure square mapping and derive a hypothesis about their probability distribution. This distribution is used in order to distinguish the proposed mappings TF-0 and TF-0m with significant advantage. Let us consider a scheme which consists of an update function f and an output function g with the notation of Sect. 1.2. Let us

further define the random variables X and X' over the set $\mathcal{X} = \{0,1\}^n$, with uniformly distributed X and with $X' = \mathsf{f}(X)$. Equivalently, Y and Y' are random variables over $\mathcal{Y} = \{0,1\}^m$ with uniformly distributed Y and with $Y' = \mathsf{g}(\mathsf{f}(X))$. Given $\Pr_Y$, $\Pr_{Y'}$ and some uniform random or pseudo-random output respectively, we can perform a statistical test (e.g. a Neyman-Pearson test, see Appendix A for more details) in order to assign the output to a distribution. We are interested in the overall complexity of the distinguisher corresponding to some designated overall error probability π_e.

For small[3] word sizes n, the distribution $\Pr_{Y'}$ can be determined by an exhaustive computation of $\mathsf{g}(\mathsf{f}(x))$ for all 2^n elements x, resulting in a precomputation time complexity of $\mathcal{O}(2^n)$ and a memory complexity (measured with the number of required memory cells) of $\mathcal{O}(2^m)$. Given both distributions and a designated overall error probability, the data complexity of an optimal distinguisher is estimated with help of the squared Euclidean imbalance (see Appendix A). We assume that the test is performed in real-time, hence we do not need additional memory in order to store the data. The online time complexity is identical to the data complexity.

However, a precomputation of $\Pr_{Y'}$ might be infeasible for large values of n (e.g. $n = 64$ bit). We perform some detailed analysis of $\Pr_{Y'}$ for small word sizes n and establish an analytical hypothesis for the approximated distribution of Y', considering *only the most biased* elements. This significantly reduces the offline time and memory complexity, but might increase the online time and data complexity of the distinguisher, given some π_e. For small word sizes n, the hypothesis can be verified with the accurate distributions, and for large n, the quality of the hypothesis will be directly examined by the experimental data complexity of the distinguisher.

2.1 Distribution of the Pure Square Mapping

Let us define the pure square mapping $\mathsf{f}(x) = x^2 \bmod 2^n$ and $\mathsf{g}(x) = \mathrm{msb}_m(x)$ with $m = n/2$, which we will refer as PSM. Apart from the least significant bit, f is a T-function. Iteration produces some fixed points such as 0 or 1, hence f can not be considered as an update function for a real application. However, we will be able to reduce more complex single-cycle mappings to some modified square mappings and apply the results obtained in this section; in other words, we will consider the pure square mapping as an ideal case, resulting in distinguishers with minimal data complexity compared to modified square mappings. We first mention that Klimov and Shamir [7] found an analytical expression for probabilities of single bits of the square mapping. Applying the notation $X' = \mathsf{f}(X)$ for an uniformly distributed X, they found that $\Pr([X']_0 = 0) = \frac{1}{2}$, $\Pr([X']_1 = 0) = 1$ and $\Pr([X']_i = 0) = \frac{1}{2}(1 + 2^{-\frac{i}{2}})$ for $i > 1$. However, as we will have to deal with an additional carry bit later on (which would reduce this bias significantly), we are more interested in the distribution of words. We explain how to derive highly biased probability distributions for $X' = \mathsf{f}(X)$ and $Y' = \mathsf{g}(\mathsf{f}(X))$. As shown in the next proposition, f is not a permutation, resulting in an unbalanced distribution of X' (there are some predictable elements $\mathsf{f}(x)$ with exceptionally large bias).

[3] The term *small* is used with respect to current computational possibilities, i.e. $n \lesssim 40$ bit for personal computers nowadays.

Proposition 1. *Consider the function* $\mathsf{f} : \{0,1\}^n \to \{0,1\}^n$ *with* $\mathsf{f}(x) = x^2 \bmod 2^n$. *For successive elements* $x \in \{0,\ldots,2^n-1\}$, *the images* $\mathsf{f}(x)$ *have a cyclic structure with cycle length* 2^{n-2}. *Hence* f *is neither injective nor surjective.*

Proof. As $x^2 - (2^{n-1}+x)^2 = 0 \bmod 2^n$, we have two cycles of length 2^{n-1}, and as $(2^{n-2}+x)^2 - (2^{n-2}-x)^2 = 0 \mod 2^n$, both cycles have two mirrored sequences of length 2^{n-2}. Consequently, the output of successive numbers x has the shape $abc\ldots cbaabc\ldots cba$. □

Due to the specified output function in PSM, the bias is transferred to the distribution of Y'. For a truly random scheme, any element of the output occurs with probability $\pi_0 = 2^{-n/2}$. For the particular scheme PSM, we observed (for small word sizes n) that there exist 4 outcomes with biased probability $2 \cdot \pi_0$, 12 outcomes with biased probability $1.5 \cdot \pi_0$ and so on. This property appears to be independent of n, and we therefore can establish a hypothesis for the most biased elements (which are explicitly known). Let $\mathcal{Y}_i$ be the aggregate containing elements of constant biased probability π_i. The parameter s_i denotes the cardinality of $\mathcal{Y}_i$, and n_i denotes the minimal word size for a stable occurrence of π_i. The parameters n_i, s_i and π_i are summarized in Tab. 1. Then we have for $i = 0,\ldots,k$ (limited by the condition $n \geq n_k$)

$$\begin{aligned} \mathcal{Y}_0 &= \{2^{(n-n_0)/2} \cdot j^2; \qquad j = 0,\ldots,s_0\} \\ \mathcal{Y}_i &= \{2^{(n-n_i)/2} \cdot (1+8j); j = 0,\ldots,s_i\} \\ \mathcal{Y}_\infty &= \mathcal{Y} - \textstyle\sum \mathcal{Y}_i \ . \end{aligned} \tag{1}$$

The values in Tab. 1 are determined with empirical methods, however n_i and s_i are exact at least for word sizes within our computational possibilities. In the case of PSM, π_i is exact for $i = 0, 1$, but fluctuating for $i > 1$ so we have to take an average value. A further approximation is done with the remaining elements in $\mathcal{Y}_\infty$, which are assigned to a constant (standardised) probability. The number of aggregates k determines the accuracy of the approximation. However, k is constrained by the condition $n < n_k$, and as the values of π_i are only accurate for $n_i \approx 40$, we usually choose $k = 8$ for $n > 40$ bit. This corresponds to a memory complexity of 2^{17}. Regarding the complexities of a distinguisher, increasing the number of aggregates k is coupled with more time, more memory and less data.

Table 1. Parameters of the approximated distribution for the first 9 aggregates

i	0	1	2	3	4	5	6	7	8
$\pi_i 2^m$	2.000	1.500	1.200	1.100	1.050	1.030	1.002	1.005	1.003
$n_i 2^{-2}$	2	3	4	5	6	7	8	9	10
$\log_2(s_i)$	2	3	5	7	9	11	13	15	17

2.2 Attacking the Single-Word Mapping TF-0

Let us now consider the running single-word proposal TF-0 with the update function $\mathsf{f}(x) = x + (x^2 \vee C) \bmod 2^n$ where $C \equiv 5, 7 \pmod 8$, and with the output function

$\mathsf{g}(x) = \mathrm{msb}_m(x)$ where $1 \leq m \leq n/2$ as described in [7, 10]. As the low-order bits are known to be weak, the authors of the scheme proposed $m = 1, 8, 16, 32$ for the standard word size $n = 64$ bit. Klimov and Shamir showed that f is an invertible T-function over an n-bit state x with a single cycle of length 2^n. The number of extracted bits m controls a tradeoff between security and efficiency of the scheme. We give some relationship to PSM with the next proposition.

Proposition 2. *Consider the scheme* TF-0. *If one requires* $C < 2^{n-m}$, *it is* $\mathsf{g}(\mathsf{f}(x)) - \mathsf{g}(x) = \mathsf{g}(x^2) + \alpha \mod 2^m$ *for* $n - m > 2$ *and for a carry bit* $\alpha \in \{0, 1\}$.

Proof. As $\mathsf{f}(x) = y = x + (x^2 \vee C) \bmod 2^n$, we conclude $y - x \equiv x^2 \vee C \pmod{2^n}$ for $C < 2^{n-m}$. Hence, $\mathsf{g}(y - x) \equiv \mathsf{g}(x^2 \vee C) \pmod{2^m}$ and $\mathsf{g}(y - x) \equiv \mathsf{g}(x^2) \pmod{2^m}$ for $C < 2^{n-m}$. We finally have $\mathsf{g}(y) - \mathsf{g}(x) - \alpha \equiv \mathsf{g}(x^2) \pmod{2^m}$ for $C < 2^{n-m}$ and for some carry bit $\alpha \in \{0, 1\}$. □

Proposition 2 states that the difference of two consecutive outputs of TF-0 differs only by an additive carry bit $\alpha \in \{0, 1\}$ from the output of PSM. Therefore, we may accurately approximate the distribution of the random variable $\mathsf{g}(\mathsf{f}(X)) - \mathsf{g}(X)$ by the distribution of the random variable Y' of PSM (i.e., we neglect the influence of the carry bit). We choose standard parameters $C = 5$ and $m = n/2$. In order to perform a test for large values of n, we approximate the distribution $\Pr_{Y'}$ with the hypothesis described in Sect. 2.1, using an optimal number of aggregates. The data complexities are estimated according to (9) and verified with experiments. We got an experimental data complexity of 2^{32} for $n = 64$ bit, which turns out to be very close to the estimated value, and somewhat larger than the lower limit derived by extrapolation for the accurate probability distribution. If the scheme is used as a pseudo-random number generator in large computer simulations, the output may not be considered as random after 2^{32} iterations, although we have a single-cycle of 2^{64} states. This observation is consistent with the practice nowadays, not to use more data than $\sqrt{P}$ of a pseudo-random number generator (PRNG) with period P. However, we also examined modified output functions with a smaller number of extracted bits m. Experiments show that (independently of the word size n), decreasing m by one bit increases the data complexity by a factor of 2. We conclude that, in contradiction to previous assumptions, not only the lower bits of this T-function are weak, but also the higher bits. This is an intrinsic property of the scheme, which will have consequences for other square mappings and may have consequences for more complicated output functions.

We mention that state-recovery attacks on TF-0 have been described in [2, 8]. Moreover, Mitra and Sarkar [13] described a time-memory tradeoff for the squaring problem, which may be applied to consecutive output differences of TF-0. The most efficient algorithms have a complexity of about 2^{16}.

2.3 Attacking the Multi-Word Mapping TF-0m

Several multi-word update functions proposed in [9] have been attacked with a time-memory tradeoff by Mitra and Sarkar [13]. We now present a distinguishing attack against a multi-word proposal which has not been broken yet, and which we will

refer as TF-0m. The update function f corresponds to (12) in [9], it is an invertible T-function over a $4n$-bit state $x = (x_0, x_1, x_2, x_3)$ with a single cycle of length 2^{4n}:

$$\mathsf{f} : \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 + (s_0^2 \vee C_0) \\ x_1 + (s_1^2 \vee C_1) + \kappa_0 \\ x_2 + (s_2^2 \vee C_2) + \kappa_1 \\ x_3 + (s_3^2 \vee C_3) + \kappa_2 \end{pmatrix} . \tag{2}$$

It is $s_0 = x_0$, $s_1 = s_0 \oplus x_1$, $s_2 = s_1 + x_2$, $s_3 = s_2 \oplus x_3$. The constants are satisfying $[C_i]_0 = 1$ for $i \in \{0, 1, 2, 3\}$, and $[C_3]_2 = 1$. All operations are carried out on n bit words and κ_i denotes the carry bit of x_i. The output function is $\mathsf{g}(x) = \mathrm{msb}_m(x_3)$ with $m = n/2$. We choose the standard word size $n = 64$ bit. The multi-word update function (2) consists of 4 approximatively independent and identically distributed (iid) random variables similar to the single-word update function of TF-0. We may concentrate only on the most significant variable x_3. The argument to be squared s_3 can be approximated as uniformly distributed, and therefore produces the same output as x^2. The carry bit modifies the output with a probability of 2^{-33}; this infrequent event will not have a significant influence to the distinguisher. Therefore, we do not have to modify the approximate distribution used for the distinguisher. Theoretical data complexity remains the same, and simulations result in an experimental data complexity of 2^{32} for a 256 bit state with 224 unknown bits. We have performed 20 experiments, observing no incorrect decision of our distinguisher. The data complexity is very close to the complexity for TF-0, confirming our assumption on the influence of κ and s. We emphasize the practical applicability of this result and the small number of required data, compared to the large number of unknown bits. As before, we also considered to extract less bits $m < n/2$. Again, we found that decreasing m by one bit increases the data complexity by a factor of 2. Hence reduction of m may still not prevent practical attacks.

3 Cryptanalysis of TSC

We start this section with a description of the recent proposal of stream cipher family TSC [4]. We find a very efficient distinguishing attack on TSC-1, as well as a distinguishing attack on TSC-2.

3.1 Description of the Schemes

The hardware-oriented stream cipher family TSC consists of a state vector of 128 bits $x = (x_0, x_1, x_2, x_3)$, an update T-function f and an output function g. The update function consists of an odd 32-bit parameter $\alpha(x)$ and a single-cycle S-box S, mapping a 4 bit input to a 4 bit output. If $[\alpha]_i = 0$, then the mapping S^e is applied on bit-slice i of the state, otherwise the mapping S^o is applied. e (resp. o) is an even (resp. odd) number. This procedure is repeated for all 32 bit-slices in a single update period. With the satisfaction of these properties, f is a single-cycle T-function, hence the period of the cipher is 2^{128}. The odd parameter is defined by $\alpha = (p + C) \oplus p \oplus 2s$ with a constant C, $p = x_0 \wedge x_1 \wedge x_2 \wedge x_3$ and $s = x_0 + x_1 + x_2 + x_3$. Except for the lower few

bits, each output bit of α is equal to 1 almost half of the time. Due to the properties of an odd parameter, one has $[\alpha]_0 = 1$, meaning that the least significant bit-slice is always mapped by S^o. Consequently, the bits from the least significant bit-slice of the state will be referred as *irregular bits*. Let us define the specified proposals. In TSC-1, the powers of the S-box are $e = 2$ and $o = 1$, the constant used in the odd parameter is $C =$ `0x12488421`, and the S-box (in standard notation) and the output function are defined by

$$\begin{aligned} \mathsf{S} &= (3, 5, 9, 13, 1, 6, 11, 15, 4, 0, 8, 14, 10, 7, 2, 12) \\ \mathsf{g}(x) &= (x_{0 \lll 9} + x_1)_{\lll 15} + (x_{2 \lll 7} + x_3) \ . \end{aligned} \tag{3}$$

In TSC-2, one has $e = 0$ (hence, the identical mapping is used), $o = 1$ and $C =$ `0x00000001`. The S-box and the output function are defined by

$$\begin{aligned} \mathsf{S} &= (5, 2, 11, 12, 13, 4, 3, 14, 15, 8, 1, 6, 7, 10, 9, 0) \\ \mathsf{g}(x) &= (x_{0 \lll 11} + x_1)_{\lll 14} + (x_{0 \lll 13} + x_2)_{\lll 22} + (x_{0 \lll 12} + x_3) \ . \end{aligned} \tag{4}$$

The output functions have a period of 2^{128}, however, three state variables in the output equation determine the remaining variable, hence the maximum security of the ciphers is 96 bit. Furthermore, there are some time-memory tradeoffs on TSC with large precomputation time complexities.

3.2 Attacking the Stream Cipher TSC-1

In this section, we present a linearisation attack on TSC-1. Probabilistic linear relations in the update function (i.e. relations between state bits at different time instants) and in the output function (i.e. relations between state bits and output bits) are combined, in order to obtain relations between output bits at different time instants. Provided that the relations are biased, the output of TSC-1 can be distinguished from a random stream.

Let us first discuss a linear approximation of the T-function. We focus on a single bit $[x_j^t]_i$ and analyse the statistical effect of Δ iterations to this bit. Let Y_Δ be the indicator variable of the event $[x_j^t]_i = [x_j^{t+\Delta}]_i$, implying that a fixed bit is repeated after Δ iterations. After Δ iterations, bit-slice i (including the bit under observation) is mapped δ times by S, with $\Delta \le \delta \le 2\Delta$ (the mapping S is applied $2\Delta - \delta$ times, and the mapping S^2 is applied $\delta - \Delta$ times). Hence, in order to compute $\Pr(Y_\Delta = 1)$, we have to analyse the distribution of δ and the bit-flip probabilities of the mappings S^δ. Let us denote $b_\Delta(\delta)$ the probability that after Δ iterations, the S-box is applied δ times. For regular bit-slices, we reasonably assume equal probabilities for the application of S and S^2 (which is, however, a simplification for some lower bit-slices), and binomial distribution for b_Δ,

$$b_\Delta(\delta) = \binom{\Delta}{\delta - \Delta} \cdot \left(\frac{1}{2}\right)^\Delta \ . \tag{5}$$

For the irregular bit-slice, it is $b_\Delta(\delta) = 1$ for $\delta = \Delta$, and zero otherwise. In order to describe the effect of the mappings S^δ, let us analyse the S-box. We will denote w an uniform random number $0 \le w \le 15$, and i an index $0 \le i \le 31$. Let also X_δ be the

indicator variable of the event $[w]_i = [\mathsf{S}^\delta(w)]_i$ for any fixed bit position i. The S-box is designed such that the *bit-flip probability* for an application of S and S^2 is balanced. However, there is a huge bias of the bit-flip probability for some multiple applications of S, namely for $\Pr(X_\delta = 1)$ with $\delta = 0 \mod 4$ (this observation is of course portable to the mapping S^2). We find $\Pr(X_4 = 1) = \Pr(X_{12} = 1) = 1/8$, $\Pr(X_8 = 1) = 3/4$ and of course $\Pr(X_{16} = 1) = 1$. These results are independent of bit-position i, other values of δ result in balanced probabilities. Finally, the bit-flip probability $P(Y_\Delta)$ of a single bit in the state for Δ iterations simply becomes the weighted sum

$$\Pr(Y_\Delta = 1) = \sum_{\delta=\Delta}^{2\Delta} \Pr(X_\delta = 1) \cdot b_\Delta(\delta) \ . \tag{6}$$

We find a maximal bias for $\Delta = 3$ with $\Pr(Y_3 = 1) = 0.3594$, and still a large bias for many other values of Δ. The predicted probabilities are in good agreements with experiments. In the case of irregular bits, (6) simply becomes $\Pr(Y_\Delta = 1) = \Pr(X_\Delta = 1)$ with a large bias for $\Delta = 0 \mod 4$.

In the fictive case of a perfect single-cycle S-box (which, however, does not exist) with $\Pr(X_\delta = 1) = 1/2$ for $\delta \neq 16$ and $\Pr(X_{16} = 1) = 1$, (6) becomes $\Pr(Y_\Delta = 1) = (b_\Delta(16) + 1)/2$ for regular bits. A maximal bias is obtained for $\Delta = 11$, resulting in $\Pr(Y_{11} = 1) = 0.6128$.

Let us combine the relation (6) with a simple linear approximation of the output function. The bias of Y_Δ strikes through the output function, such that the loops in the state are also present in the output. We consider a single bit $[y^t]_i$ of the output and analyse the statistical effect of Δ iterations to this bit. Let Z_Δ be the indicator variable of the event $[y^t]_i = [y^{t+\Delta}]_i$, implying that a fixed bit of the output is repeated after Δ iterations. We approximate the output function by $[y]_i = [x_0]_{i+8} \oplus [x_1]_{i+17} \oplus [x_2]_{i+25} \oplus [x_3]_i \oplus c$, for $i = 0, \ldots, 31$ (additions of indices are performed modulo 32) and a carry bit $c \in \{0, 1\}$. For bit-positions $i = 0, 7, 15, 24$, one irregular bit is involved in the linear approximation of $[y]_i$; consequently, these output bits are called irregular. Neglecting the carry bit and availing the fact that the output bits are composed of independent state bits, the probability $\Pr(Z_\Delta = 1)$ is approximated using Matui's *Piling-up Lemma* [12]. For regular output bits, we obtain

$$\Pr(Z_\Delta = 1) = \frac{1}{2} + 2^3 \cdot \left(\Pr(Y_\Delta = 1) - \frac{1}{2} \right)^4 \ . \tag{7}$$

Notice that $\epsilon = \Pr(Y_\Delta = 1) - \frac{1}{2}$ is the probability bias. In the case of irregular output bits, one of the four factors ϵ in (7) is substituted by $\epsilon' = \Pr(X_\Delta = 1) - \frac{1}{2}$. Let us consider the case of $\Delta = 3$; it is $\Pr(Z_3 = 1) = 0.5031$ for regular output bits (and a balanced probability for irregular output bits). However, as we neglected the carry bit in this simple model, the above probability is considered as an upper limit. Notice that the carry is also biased and inclines towards absorbing itself. Experiments show that indeed, most of the regular output bits are biased for $\Delta = 3$. We emphasise that higher bits are affected equivalently to lower bits. Due to the integer addition, the exact bias depends on the bit-position. We find the maximum bias for bit-position $i = 1$ with $p' = 0.5003$. A similar result is obtained for $\Delta = 8$ and $i = 0$.

This biased probability is accessible to a cryptanalyst with known plaintext and may be used to distinguish the outcome of the cipher from a uniform random outcome. With the uniform probability $p = 1/2$ and the biased probability $p' = p(1+q)$, the required data complexity becomes $\mathcal{O}(1/pq^2)$, see Theorem 2 in [11]. Consequently, for $p' = 0.5003$ we expect a data and online time complexity of about 2^{22} (16 MB of keystream); offline time complexity is negligible. We performed a number of experiments (taking all biased bits into account) and verified the predicted complexity, given a small probability of error.

As described above, a variant of this attack even works without taking into account any specific property of the single-cycle S-box. Finally, we mention that the bias of Z_Δ can be transformed in a state-recovery attack by guess-and-determine. In a first step, we guess the least-significant bit-slice $[x^t]_0$, which may be iterated separately. The four corresponding bits are subtracted independently from appropriate output bits in order to construct a modified index variable. Considering (7), we expect the bias to significantly increase for a right guess, and we expect a balanced output for a false guess. After recovering $[x^t]_0$, we may continue with consecutive bit-slices. Considering all available equations, experiments showed that a single bit-slice may be accepted or rejected (with a reasonable probability of error) using 2^{22} iterations. Repeating this for all 2^4 values of a single bit-slice, and for all 2^5 bit-slices, we obtain an overall complexity of about 2^{31}. A similar result has also been obtained by Peyrin and Muller [14].

3.3 Attacking the Stream Cipher TSC-2

In both versions of TSC, the 32 bits of α determine the update of the 128 bits of the state. Hence we may wait for appropriate values of α in order to initiate some attacks. In TSC-2, an interesting case is the minimal-weight parameter $\alpha = 1$, for which only the least significant bit-slice is modified and two similar successive outputs may be detected. The *detector* is an algorithm which takes as input the keystream z and gives out 1 if $\alpha = 1$, and 0 otherwise. The detector can make two types of errors: it can either output 1 when $\alpha \neq 1$ (false positives) or 0 when $\alpha = 1$ (false negatives). The error probabilities are denoted by A and B, respectively.

The complete set of states $\mathcal{U}$ resulting in $\alpha(x^t) = 1$ is given with the conditions $\sum_{i=0}^{3} x_i^t \in \{\texttt{0x00000000}, \texttt{0x80000000}\}$ and $[x^t]_0 \in \{\texttt{0x0}, \texttt{0x3}, \texttt{0x5}, \texttt{0x6}, \texttt{0x9}, \texttt{0xA}, \texttt{0xC}\}$. In the following, let us assume that such a state occurs at time $t = 0$. Hence we have $\alpha^0 = 1$, and only the least significant bit-slice of the state is changed by the mapping $\mathsf{f}: x^0 \to x^1$; consequently, we suppose that the subsequent outputs y^0 and y^1 have low distance. Let us analyse the exemplary integer modular difference $y^0 - y^1$ for $x \in \mathcal{U}$ with $[x^0]_0 = \texttt{0x5}$; we find that $[x^1]_0 = \texttt{0x4}$ and $[x^0]_i = [x^1]_i$ for $i \neq 0$. The output function produces $y^0 = y^1 + \texttt{1}_{\lll 25} + \texttt{1}_{\lll 3} + \texttt{1}_{\lll 12}$ and hence $y^0 - y^1 = \texttt{0x02001008}$. In fact, we find that $y^0 - y^1 = \texttt{const}$ for any $x \in \mathcal{U}$, where the constant const depends only on the least-significant bit-slice $[x^0]_0$ in most of the cases, see Tab. 2. For less than 1% of the states in $\mathcal{U}$, the integer modular difference is not constant because an addition in the output function may cause a carry bit, which propagates from the msb to the lsb due to the cyclic shift. Detection of single constants only would result in a huge amount of false alarms. However, examining Tab. 2, we find a

Table 2. List of output differences for $\alpha = 1$, some of which will be applied in the attack

$[x^0]_0$	$[x^1]_0$	$y^0 - y^1$
0x0	0x5	0xFDBFEFF8
0x3	0xC	0x01C05007
0x5	0x4	0x02001008
0x6	0x3	0xFE3FEFF8
0x9	0x8	0x02001008
0xA	0x1	0xFE002FF9
0xC	0x7	0xFDFFAFF9

path for the iteration of $[x^0]_0$ with 0x6 $\rightarrow$ 0x3 $\rightarrow$ 0xC which is closed in $\mathcal{U}$, meaning that $\alpha^0 = \alpha^1 = \alpha^2 = 1$. Therefore, we may restrict the detector to detect only a subset of states $\mathcal{V} \subset \mathcal{U}$, where $\mathcal{V}$ is defined by the conditions $\sum_{i=0}^{3} x_i^t \in \{$0x00000000, 0x80000000$\}$ and $[x^t]_0 \in \{$0x6, 0x3$\}$. The detector takes three successive outputs, computes two differences of consecutive outputs and compares them with the fixed values; if there is a match of both, the detector returns 1, and 0 otherwise. The probability of $x \in \mathcal{V}$ is 2^{-33}, and a false detection due to random outputs[4] occurs with probability 2^{-64}. As the differences are constant almost all the time, the error B (which would increase the running time of the detector) is negligible, too. The time and data complexity is around 2^{33} (no precomputation and negligible memory).

The detector may be transformed in a distinguisher by feeding the detector with a fixed amount of data n. If the detector always returns 0, then the distinguisher returns 0 (random stream); if the detector returns 1 at least once, then the distinguisher returns 1 (keystream produced by TSC-2). The probability of false positives may be neglected, and the probability of false negatives is $B = (1 - 2^{-33})^n$. For $B = 0.05$, we obtain a data complexity of about $n = 2^{34}$. With a successful detection of $\alpha(x^t) = 1$, we obtain the information $\sum_{i=0}^{3} x_i^t \in \{$0x00000000, 0x80000000$\}$, as well as the value of bit-slice $[x^t]_0$ and the output equation $\mathsf{g}(x^t) = y^t$. This information may be used for a state-recovery attack with a complexity smaller than 2^{96}. However, TSC-2 appears to be seriously injured with our efficient distinguishing attack, and we did not study the state-recovery attack in more detail.

4 Conclusions

In this paper, we examined some specific proposals of stream ciphers based on T-functions. Two proposals by Klimov and Shamir are based on the squaring operation, namely a single word T-function as well as a previously unbroken multi-word T-function with a 256-bit state, both revealing some part of the state. It turned out that the integer differences of consecutive outputs have significant statistical deviation even in the high-order bits. Based on that deviation, we described efficient distinguishing attacks with a 2^{32} data complexity. We conclude that the squaring operation has some undesirable properties when used in the design of T-functions and possibly in other cryptographic primitives. The two proposals by Hong *et al.* have

[4] In order to increase the set $\mathcal{V}$, we do not make use of the connection of the whole path.

a 128-bit state, which are controlled by a 32-bit parameter and tiny S-boxes. The output function uses some integer additions and rotations. For one of the proposals, we found small loops in the state and in the output produced by the S-box, resulting in a distinguishing attack of complexity 2^{22}. For the other proposal, we wait for an appropriate value of the parameter, which produces some detectable structure in the output. This results in a distinguisher of complexity 2^{34}. We conclude that the small size of the parameter (and potentially also the tiny S-boxes) may be critical, and that the integer additions and rotations in the output functions have a very limited randomizing effect.

Acknowledgments

This work is supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the Swiss National Science Foundation under grant number 5005-67322. The third author also receives partial funding through Gebert Rüf Stiftung. We would like to thank Jin Hong and Peter Hellekalek for valuable discussions, and the anonymous reviewers for their helpful comments.

References

1. T. Baignères, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis ? In P. Lee, editor, *Advances in Cryptology – ASIACRYPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer-Verlag, 2004.
2. V. Benony, F. Recher, E. Wegrzynowski, and C. Fontaine. Cryptanalysis of a particular case of Klimov-Shamir pseudo-random generator. In T. Helleseth, D. Sarwate, H.-Y. Song, and K. Yang, editors, *Sequences and Their Applications – SETA 2004, Third International Conference, Seoul, Korea, October 24-28. Revised Selected Papers*, volume 3486 of *Lecture Notes in Computer Science*, pages 313–322. Springer-Verlag, 2004.
3. J. Hong, D. Lee, Y. Yeom, and D. Han. New class of single cycle T-functions and a stream cipher proposal. Presented at *SASC – The State of the Art of Stream Ciphers*, ECRYPT Workshop, October 14-15, Brugge, Belgium, 2004.
4. J. Hong, D. Lee, Y. Yeom, and D. Han. A new class of single cycle T-functions. To appear in H. Gilbert and H. Handschuh, editors, *Fast Software Encryption 2005, 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005. Revised Papers*, volume 3557 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
5. P. Junod. On the optimality of linear, differential and sequential distinguishers. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003. Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, 2003.
6. A. Klimov. *Applications of T-functions in cryptography.* PhD thesis, Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot (Israel), 2004.
7. A. Klimov and A. Shamir. A new class of invertible mappings. In B. Kaliski, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002: 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002. Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 470–483. Springer-Verlag, 2002.
8. A. Klimov and A. Shamir. Cryptographic applications of T-functions. In *Selected Areas in Cryptography: 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 2003. Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, 2004.
9. A. Klimov and A. Shamir. New cryptographic primitives based on multiword T-functions. In B. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2004.

10. A. Klimov and A. Shamir. The TFi family of stream ciphers. Technical note, 2004.
11. I. Mantin and A. Shamir. A practical attack on broadcast RC4. In M. Matsui, editor, *Fast Software Encryption: 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001. Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 152–164. Springer-Verlag, 2002.
12. M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 1993. Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1993.
13. J. Mitra and P. Sarkar. Time-memory trade-off attacks on multiplications and T-functions. In P. Lee, editor, *Advances in Cryptology – ASIACRYPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 468–482. Springer-Verlag, 2004.
14. T. Peyrin and F. Muller. Personal communication, 2005.

A Optimal Distinguishers

In a recent paper, Baignères *et al.* [1] have analysed optimal algorithms (in terms of number of samples) aiming at distinguishing two random sources whose probability distributions are completely known to a cryptanalyst. We briefly recall the framework of Baignères *et al.*

Let D_0 and D_1 be two probability distributions sharing the same support $\mathcal{X}$. We consider the problem of distinguishing these two distributions using ν iid samples. A (possibly computationally unbounded) algorithm $\boldsymbol{\delta}^\nu$ which takes as input a sequence of ν realizations $\boldsymbol{z}^\nu$ distributed according to D where either $\mathsf{D} = \mathsf{D}_0$ or $\mathsf{D} = \mathsf{D}_1$, and outputs 0 or 1 according to its decision, is called a *distinguisher*. It can be fully determined by an acceptance region $\mathcal{A} \subset \mathcal{X}$ such that $\boldsymbol{\delta}^\nu(\boldsymbol{z}^\nu) = 1$ iff $\boldsymbol{z}^\nu \in \mathcal{A}$. The ability to distinguish a distribution from another is usually measured in terms of the *advantage* of the distinguisher and is defined by

$$\mathrm{Adv}_{\boldsymbol{\delta}^\nu} = \left| \Pr_{\mathsf{D}_0^\nu}[\boldsymbol{\delta}^\nu(\boldsymbol{Z}^\nu) = 0] - \Pr_{\mathsf{D}_1^\nu}[\boldsymbol{\delta}^\nu(\boldsymbol{Z}^\nu) = 0] \right| .$$

Hence, the distinguisher can make two types of errors: it can either output 0 when $\mathsf{D} = \mathsf{D}_1$ or 1 when $\mathsf{D} = \mathsf{D}_0$; we will denote these respective error probabilities by α and β, respectively, and the overall error probability is defined as $\pi_\mathrm{e} = \frac{1}{2}(\alpha + \beta)$.

In [5] it is shown that it is easy to define explicitly an optimal distinguisher in this precise statistical setting. Indeed, given a fixed overall probability of error, it is sufficient for an optimal distinguisher to count the number $\nu_x(\boldsymbol{z}^n)$ of occurrences of all possible symbols $x \in \mathcal{X}$ in the sample $\boldsymbol{z}^n$, to compute the log-likelihood ratio

$$\mathsf{llr}(\boldsymbol{z}^\nu) = \sum_{x \in \mathcal{X}} \nu_x(\boldsymbol{z}^\nu) \log \frac{\Pr_{\mathsf{D}_0}[x]}{\Pr_{\mathsf{D}_1}[x]} \tag{8}$$

and to output 0 as decision iff $\mathsf{llr}(\boldsymbol{z}^\nu) > 0$. If we assume that the distributions D_0 and D_1 are close to each other, i.e. $\Pr_{\mathsf{D}_0}[x] = \pi_x$ and $\Pr_{\mathsf{D}_1}[x] = \pi_x + \varepsilon_x$ with $|\varepsilon_x| \ll \pi_x$ for all $x \in \mathcal{X}$, then the following result gives a very accurate estimation of the necessary number of samples.

Theorem 1 (Baignères *et al.* [1]). *Let $X_1, \ldots, X_\nu$ be iid random variables defined over $\mathcal{X}$ with probability distribution D, let D_0 and D_1 be two distributions sharing the*

same support which are close to each other, where $\pi_x = \Pr_{\mathsf{D}_0}[x]$ *and* $\pi_x + \varepsilon_x = \Pr_{\mathsf{D}_1}[x]$. *Let* d *be a real number defined by*

$$d = \nu \sum_{x \in \mathcal{X}} \frac{\varepsilon_x^2}{\pi_x} \ .$$

Then, the overall probability of error of an optimal distinguisher between D_0 *and* D_1 *is approximately*

$$\pi_{\mathrm{e}} \approx \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\sqrt{d}}{2}} e^{-\frac{t^2}{2}} \, dt \ .$$

Baignères *et al.*, based on this result, introduced then what seems to be a natural "measure", named *squared Euclidean imbalance* and denoted $\Delta(\mathsf{D}_0, \mathsf{D}_1)$, between a distribution D_0 and a close distribution D_1 defined by

$$\Delta(\mathsf{D}_0, \mathsf{D}_1) = \sum_{x \in \mathcal{X}} \frac{\varepsilon_x^2}{\pi_x} \ , \tag{9}$$

since $\Delta(\mathsf{D}_0, \mathsf{D}_1)$ is directly linked to the number of samples needed to distinguish both probability distributions with a good success probability.