Time-Memory-Data Trade-off Attack on Stream Ciphers based on Maiorana-McFarland Functions *

Khoongming Khoo
DSO National Laboratories
20 Science Park Dr, S118230, Singapore
email: kkhoongm@dso.org.sg

Guanhan Chew
DSO National Laboratories
20 Science Park Dr, S118230, Singapore
email: cguanhan@dso.org.sg

Guang Gong
Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ont. N2L 3G1, Canada
email: ggong@calliope.uwaterloo.ca

Hian-Kiat Lee
DSO National Laboratories
20 Science Park Dr, S118230, Singapore
email: lhiankia@dso.org.sg

July 9, 2008

Abstract

In this paper, we present the time-memory-data (TMD) trade-off attack on stream ciphers filter function generators and filter cominers based on Maiorana-McFarland functions. This can be considered as a generalization of the time-memory-data trade-off attack of Mihaljevic and Imai on Toyocrypt. First, we substitute the filter function in Toyocrypt (which has the same size as the LFSR) with a general Maiorana-McFarland function. This allows us to apply the attack to a wider class of stream ciphers. Second, we highlight how the choice of different Maiorana-McFarland functions can affect the effectiveness of our attack. Third, we show that the attack can be modified to apply on filter functions which are smaller than the LFSR and on filter-combiner stream ciphers. This allows us to cryptanalyze other configurations commonly found in practice. Finally, filter functions with vector output are sometimes used in stream ciphers to improve the throughput. Therefore the case when the Maiorana-McFarland functions have vector output is investigated. We found that the extra speed comes at the price of additional weaknesses which make the attacks easier.

Keywords: Time-memory-data trade-off attack, Maiorana-McFarland functions.

1 Introduction

The construction of Boolean functions with good cryptographic properties has been a well studied area of research. Some of these properties include balance, high nonlinearity, high order of resiliency, high

*This is a corrected and extended version of a paper "The Rainbow Attack on Stream Ciphers based on Maiorana-McFarland Functions" presented at the ACNS 2006 conference.

algebraic degree and high order of propagation criteria. These properties ensure the Boolean functions are resistant against various correlation attacks when used in stream ciphers [4, 22].

A well-known class of Boolean functions with good cryptographic properties is the Maiorana-McFarland class which ensures many of the above mentioned properties. For example, when n is odd, we can construct t-resilient n-bit functions with nonlinearity achieving the quadratic bound $2^{n-1} - 2^{(n-1)/2}$. By concatenating such a function with its complement, we construct $(t+1)$-resilient m-bit functions with high nonlinearity satisfying the quadratic bound $2^{m-1} - 2^{m/2}$ for even $m = n+1$. These nonlinearities are called the quadratic bounds because they are the maximum nonlinearities attainable for quadratic Boolean functions. When n is even, the Maiorana-McFarland also allows us to construct a large family of bent functions, i.e. Boolean functions with the highest nonlinearity $2^{n-1} - 2^{n/2-1}$. Finally, the saturated functions which achieve optimal order of resiliency $t = n - 1 - d$ and optimal nonlinearity $2^{n-1} - 2^{t+1}$ when the algebraic degree is d can be constructed by this method.

The Maiorana-McFarland class can be viewed as constructions based on concatenating linear functions. This is both an advantage and a weakness. It is an advantage because we can easily manipulate the distance between Maiorana-McFarland functions and linear functions to obtain resiliency and high nonlinearity. This helps to protect against correlation and fast correlation attacks [4, 22]. However, it also means the function becomes linear when we fix certain input bits. Mihaljevic and Imai were able to exploit this property to launch a search space reduction attack on Toyocrypt. Toyocrypt is a 128-bit stream cipher where a 128-bit modular linear feedback shift register (MLFSR) is filtered by a 128-bit Maiorana-McFarland function. They were able to reduce the key space from 2^{128} to 2^{96} when 32 consecutive output bits are known. The attack works because for each guess on 96 bits of the 128-bit MLFSR, they were able to form 32 linear equations based on 32 consecutive output bits. This linear system can be solved to determine the remaining 32 bits in the MLFSR. Using the time-memory-data (TMD) trade-off attack of Biryukov and Shamir, they further reduced the attack complexity to 2^{32} with 2^{80} pre-computation and 2^{64} memory.

In Section 3, we generalize the TMD trade-off attack on Toyocrypt by Mihaljevic and Imai [15] as follows:

1. We show that the search space reduction attack on Toyocrypt can be applied to a general Maiorana-McFarland function. Because linear feedback shift registers (LFSR's) are more commonly used in stream ciphers, we replace the MLFSR in Toyocrypt by an LFSR.

2. We simplify the description of the time-memory-data trade-off attack of [15] by introducing a search function $F^{(c)}(x)$.

Based on our study, we characterize the performance of the attack for different Maiorana-McFarland functions. For Maiorana-McFarland functions formed by concatenating $2^{n/2}$ linear functions of size $n/2$-bit, the search space is reduced from 2^n to $2^{3n/4}$. When we apply the TMD attack, the search space is further reduced to $2^{n/4}$ with $2^{3n/8}$ consecutive keystream bits, $2^{5n/8}$ pre-computation and $2^{n/2}$ memory. This case corresponds to:

1. Filter function in Toyocrypt with $n = 128$, $k = 64$.
2. Bent functions [5, 21].
3. Resilient functions whose nonlinearity satisfies the quadratic bound [5, 21].

For Maiorana-McFarland functions formed by concatenating a few large linear functions, we get a very effective reduction of the search space of an n-bit filter function generator from 2^n to $2^{n/2}$. In this case, the equivalent keylength is only half of what is claimed. As shown by Gong and Khoo [9], this case correspond to the:

1. Saturated functions with optimal trade-off between degree, resiliency and nonlinearity,

introduced by Sarkar and Maitra at Crypto 2000[1] [19]. Thus although this class of functions has the best trade-off among important cryptographic properties like nonlinearity, resiliency and algebraic degree, they are weak against the search space reduction attack. When we apply the TMD attack, the search space is further reduced to $2^{n/4}$ with $2^{n/2}$ consecutive keystream bits, $2^{n/2}$ pre-computation and $2^{3n/8}$ memory.

In Section 5, we extend our attack to the case where the Maiorana-McFarland function is of smaller size than the LFSR. This is a very common construction when the filtering function is implemented as a look-up-table (LUT). The LFSR may be 128-bit long and it is not possible to fit a LUT of size 2^{128} into the memory of the cipher. Thus a smaller filter function has to be used. In that case, the complexity of the search space reduction and TMD attack depends on the width of the LFSR bits which are tapped to certain input bits of the filter function. These input bits have the property that when they are known, the Maiorana-McFarland function becomes linear.

In Section 6, we extend the attack to the filter combiner model. At each clock cycle, the Boolean function will extract several bits from each of s linear feedback shift registers $LFSR_i$, $i = 0, 1, \ldots, s-1$, as input to produce a keystream bit. As analyzed by Sarkar [18], the filter combiner offers various advantages over the filter function and combinatorial generators. We show that in this case, the search space reduction and TMD attack can also be applied effectively.

[1] We note that at the time of the discovery of the saturated functions by Sarkar and Maitra, similar optimal cryptographic bounds and functions were also independently discovered by Tarannikov [23] and Zheng, Zhang [24].

In Section 7, we extend the attack to the case where the filter function is a vectorial Maiorana-McFarland function. Vector output filter function generator has higher throughput for faster communication speed but it has an additional weakness. There is an exponential decrease in the complexity of search space reduction when compared to the single output case. This gives a very efficient attack even by a direct exhaustive search. This complexity can be further reduced by applying the TMD attack.

In Sections 3 to 7, we simplified the attack scenarios to give a clearer explanation of the attack methods. In Section 8, we describe how these attacks can be easily adapted to apply to stream ciphers that use more general linear finite state machines, tap points and filter functions. Finally, we conclude the paper in Section 9.

2 Preliminaries

2.1 Notation

We adopt the following conventions for the functions used in this paper:

$f(\cdot)$	Maiorana-McFarland functions defined, for example, in Equation (1)
$g(\cdot), \phi(\cdot)$	Nonlinear functions used in the definition of $f(\cdot)$
$F^{(c)}(\cdot)$	The search function derived from $f(\cdot)$
$F(\cdot)$	Vectorial Maiorana-McFarland function defined in Equation (5)
$\tilde{f}(\cdot)$	Output from a stream cipher based on $f(\cdot)$

2.2 Maiorana-McFarland Functions

The *Hadamard Transform* of a Boolean function $f : GF(2)^n \to GF(2)$ is

$$\hat{f}(w) = \sum_{x \in GF(2)^n} (-1)^{w \cdot x + f(x)}.$$

The *nonlinearity* of a function $f : GF(2)^n \to GF(2)$ is defined as

$$N_f = 2^{n-1} - \frac{1}{2} \max_w |\hat{f}(w)|.$$

A high nonlinearity is desirable as it ensures linear approximation of f is ineffective. This offers protection against linear approximation based attacks [13, 22].

A Boolean function $f : GF(2)^n \to GF(2)$ is *t-th order correlation immune*, denoted $CI(t)$, if $\hat{f}(w) = 0$ for all $1 \leq wt(w) \leq t$ where $wt(w)$ is the number of ones in the binary representation of w. Correlation immunity ensure that f cannot be approximated by linear functions with too few terms, which offers protection against correlation attack [22]. Furthermore, if f is balanced and $CI(t)$, we say f is resilient of order t.

The Maiorana-McFarland function is defined by the equation:

$$\begin{aligned} f(x_0, \ldots, x_{n-1}) &= g(x_0, \ldots, x_{k-1}) \\ &+ (x_k, \ldots, x_{n-1}) \cdot \phi(x_0, \ldots, x_{k-1}) \end{aligned} \tag{1}$$

where $f : GF(2)^n \rightarrow GF(2)$, $g : GF(2)^k \rightarrow GF(2)$ and $\phi : GF(2)^k \rightarrow GF(2)^{n-k}$. ϕ is usually an injection or 2-to-1 map which would require $k \leq n/2$ or $k \leq n/2 + 1$ respectively.

The Maiorana-McFarland functions had been used extensively to construct Boolean functions with good cryptographic properties in the past decade (see [5] for a summary). Some notable examples are listed in the following two Propositions.

Proposition 1. *(extracted from [5, 21])*

1. *Let $f : GF(2)^n \rightarrow GF(2)$ be defined by equation (1). Let n be odd, $k = (n-1)/2$ and $\phi : GF(2)^{(n-1)/2} \rightarrow GF(2)^{(n+1)/2}$ be an injection such that*

$$\begin{aligned} wt(\phi(x_0, \ldots, x_{k-1})) &\geq t+1 \text{ and} \\ |\{z \in GF(2)^{(n+1)/2} | wt(z) &\geq t+1\}| \geq 2^{(n-1)/2}. \end{aligned}$$

   Then f is a t-resilient function with nonlinearity $2^{n-1} - 2^{(n-1)/2}$.

2. *Let $f(x_0, \ldots, x_{n-1})$, n odd, be a t-resilient function constructed as in part 1 of this proposition. Then $h : GF(2)^m \rightarrow GF(2)$, where $m = n+1$ is even and,*

$$h(x_0, \ldots, x_{m-1}) = h(x_0, \ldots, x_{n-1}, x_n) = f(x_0, \ldots, x_{n-1}) + x_n,$$

   is $t+1$-resilient and has nonlinearity $2^{m-1} - 2^{m/2}$.

3. *Let n be even, $k = n/2$ and $\phi(x_0, \ldots, x_{k-1})$ be a permutation in equation (1), then $f(x)$ is a bent function, i.e. it has the highest possible nonlinearity $2^{n-1} - 2^{n/2-1}$.*

The first construction is quite useful because a nonlinearity of $2^{n-1} - 2^{(n-1)/2}$ is considered high for functions with odd number of input bits. Furthermore, when $n \equiv 1 \pmod 4$, we can also obtain resiliency of order $(n-1)/4$ [5, page 555]. The second construction derives highly nonlinear resilient function with even number of input bits from the first construction. The third construction on bent functions is widely used in cryptography because of their high nonlinearity. Some examples include the ciphers CAST and Toyocrypt [1, 15]. The functions presented in Proposition 1 has the common property that $k \approx n/2$.

The saturated functions are functions which attain optimal trade-off between algebraic degree d, order of resiliency $t = n-d-1$ and nonlinearity $2^{n-1} - 2^{t+1}$. Such functions were constructed by Sarkar and Maitra in [19]. It was shown by Gong and Khoo in [9] that the saturated functions correspond to n-bit Maiorana-McFarland functions as follows.

Proposition 2. *(Sarkar, Maitra [19, 9]) Fix $d \geq 2$ and let $n = 2^{d-1} + d - 2$. Define $f : GF(2)^n \to GF(2)$ by equation (1) where $k = d - 1$. Let $\phi : GF(2)^{d-1} \to GF(2)^{2^{d-1}-1}$ be an injection such that $wt(\phi(x_0, \ldots, x_{k-1})) \geq 2^{d-1} - 2$. Then $deg(f) = d$, f is t-resilient having nonlinearity $2^{n-1} - 2^{t+1}$ where $t = n - 1 - d$. In that case, the order of resiliency is optimal by Siegenthaler's inequality [22] and nonlinearity is optimal by Sarkar-Maitra inequality [19].*

The function in Proposition 2 has the property that $k \approx \log_2(n) \ll n$.

Propositions 1 and 2 construct Boolean functions with optimal cryptographic properties by concatenating linear functions. But we shall show that their linear structures can be exploited to give efficient attacks on stream ciphers in Section 3.

3 The Time-Memory-Data Trade-Off Attack on Maiorana-McFarland Functions

In [15], Mihaljevic and Imai presented a time-memory-data trade-off attack on the stream cipher Toyocrypt. Toyocrypt is a filter function generator where we have an MLFSR of length 128 bit filtered by a 128-bit Boolean function of the form:

$$\begin{aligned} f(x_0, \ldots, x_{127}) &= g(x_0, \ldots, x_{63}) \\ &+ (x_{64}, \ldots, x_{127}) \cdot \pi(x_0, \ldots, x_{63}), \end{aligned}$$

where g has 3 terms in its algebraic normal form (ANF) of degree $4, 17, 63$ and π permutes the bit positions of $(x_0, \ldots, x_{63})$. Mihaljevic and Imai showed that the effective key diversity of such a generator can be reduced from 128 bits to 96 bits when 32 consecutive output bits are known. Based on this observation, they modified the Biryukov-Shamir [2] time-memory-data trade-off attack for improved cryptanalysis.

It is easy see that the filter function in Toyocrypt is a Maiorana-McFarland function with parameters $n = 128$, $k = 64$. Due to the wide usage of the Maiorana-McFarland construction, it will be useful to generalize the Mihaljevic-Imai attack to a general Maiorana-McFarland filter function generator. In this attack, we look at a stream cipher where an n-stage LFSR is filtered by a n-bit Maiorana-McFarland function defined by equation (1). We assume bit i of the LFSR is the i-th input of $f(x)$.

Suppose we know $l = \lceil \frac{n-k}{2} \rceil$ consecutive output bits $y_0, \ldots, y_{l-1}$ and let $(x_i, \ldots, x_{i+n-1})$ be the LFSR state corresponding to y_i:

$$\begin{aligned} y_i &= g(x_i, \ldots, x_{i+k-1}) \\ &+ (x_{i+k}, \ldots, x_{i+n-1}) \cdot \phi(x_i, \ldots, x_{i+k-1}). \end{aligned} \tag{2}$$

If we guess $n-l = \lfloor \frac{n+k}{2} \rfloor$ internal state bits $x_0, \ldots, x_{n-l-1}$, then for $n-l-1-(k-1)+1 = n-l-k+1 = \lfloor \frac{n+k}{2} \rfloor - k + 1 = \lfloor \frac{n-k}{2} \rfloor + 1$ clocks, the inputs to the functions g and ϕ will be known and the right hand sides of Equation (2) will be linearized. Thus the l equations we form with the l known output bits $y_0, \ldots, y_{l-1}$ will be linear because $l = \lceil \frac{n-k}{2} \rceil \leq \lfloor \frac{n-k}{2} \rfloor + 1$. Moreover, the number of unknowns is l since $x_n, \ldots, x_{2n-l-k}$ in Equation (2) can be linearly expressed in terms of $x_0, \ldots, x_{n-1}$ using the LFSR feedback relation. Therefore, we can solve for these l unknown bits by Gaussian elimination. We can check whether our guess is correct by back-substituting and comparing with a sufficiently long keystream, e.g. of length $2n$ bits. Thus we have proven that:

Theorem 1. *Consider an n-bit LFSR filtered by Equation (1) where bit i of the LFSR is the ith input of $f(x)$. The key space is reduced from 2^n to $2^{\lfloor (n+k)/2 \rfloor}$ bits when $\lceil (n-k)/2 \rceil$ consecutive output bits are known.*

Remark 1. *For ease of notation, we assume that $(n+k)/2$, $(n-k)/2$ are integers from now on. The case when they are not integers can be handled by adding the appropriate floor $\lfloor\ \rfloor$ and ceiling $\lceil\ \rceil$ operations as in Theorem 1.*

Next we improve the attack complexity of Theorem 1 by applying the TMD attack [2]. Let $f : GF(2)^n \to GF(2)$ be the filter function of an n-bit LFSR. Define $\tilde{f} : GF(2)^n \to GF(2)^n$ as:

$$\tilde{f}(\tilde{x}) = n\text{-bit output of filter function generator,}$$

when the LFSR is initialized by $\tilde{x} \in GF(2)^n$.

Let $c \in GF(2)^{(n-k)/2}$ be a fixed string. Given $x \in GF(2)^{(n+k)/2}$, we can use the proof of Theorem 1 to find $s \in GF(2)^{(n-k)/2}$ such that:

$$\begin{aligned} &\tilde{f}(x||s) \text{ restricted to first } (n-k)/2 \text{ bits } = c, \\ &\text{i.e. } \tilde{f}(x||s) = (c||y). \end{aligned}$$

where '||' is concatenation of bit strings. Based on this computation, we introduce a *search function* $F^{(c)} : GF(2)^{(n+k)/2} \to GF(2)^{(n+k)/2}$ for the description of our attack. We define $F^{(c)}(x)$ to be the right most $(n+k)/2$ bits of $\tilde{f}(x||s)$, i.e.,

$$F^{(c)}(x) = y$$

The function $F^{(c)}$ will be the search function used in our description of the TMD attack on Maiorana-McFarland functions. We note that this function can also simplify the description of the attack in [15]. Let the amount of data collected be D.

Setup:

1. Randomly choose a binary string $c \in GF(2)^{(n-k)/2}$ and define the function $F^{(c)}(x)$ as described above.

2. Define t/D distinct functions $F_1(x)$, $F_2(x)$,...,$F_{t/D}(x)$ which are slight variations of the search function $F^{(c)}(x)$ as follows. Randomly seed an $(n+k)/2$-bit LFSR (with maximum period) and generate a sequence of distinct $(n+k)/2$-bit vectors $X_1, X_2, \ldots, X_{t/D}$. Let the variant functions be defined as $F_j(x) = F^{(c)}(x) \oplus X_j$ (Please see Remark 2 for further explanation on this step).

3. Let p and t be integers defined by $pt^2 = 2^{(n+k)/2}$. Form t/D number of $p \times 2$ array as follows:

   For array j where $j = 1 \ldots t/D$, form rows $i = 1 \ldots p$ as follows: Randomly choose a start point $y_{i,0}$ and compute the chain of values $y_{i,1} = F_j(y_{i,0}), y_{i,2} = F_j(y_{i,1}), \ldots, y_{i,t} = F_j(y_{i,t-1})$. Store the start and end points $(SP_i, EP_i) = (y_{i,0}, y_{i,t})$.

Attack:

1. We look among the keystream to find D n-bit strings whose first $(n-k)/2$ bits matches the pattern c. For one such string, let the last $(n+k)/2$ bits of this string be y.

2. For each y, search among the endpoints EP_i in the j^{th} table to check if

$$EP_i = F_j^r(y \oplus X_j),\ r = 1 \ldots t,$$

   If there is a match, then $(x||s)$ is the secret initial state of the LFSR where

$$x = F_j^{t-r-1}(SP_i),$$

   and s is the $(n-k)/2$-bit string computed such that the leftmost $(n-k)/2$ bits of $\tilde{f}(x||s)$ is c. The string s can be found by solving linear equations as in the proof of Theorem 1.

3. We repeat this process for the other $D-1$ strings.

The parameters in the attack satisfy the following constraint:

Based on [2], our table only need to cover $1/D$ of the whole search space $N = 2^{(n+k)/2} = pt^2$ because we just need to break the cipher for one string out of D possible strings in the available keystream. Since

we are only storing the end points, the memory M needed is pt/D. For each of the D data, we need to look up t/D tables and compute $F_j(x)$ t times for each table. The time taken is $T = D \times t/D \times t = t^2$ function computations. Thus we derive the relation:

$$\begin{aligned} TM^2D^2 &= t^2 \times (pt/D)^2 \times D^2 \\ &\approx p^2t^4 = N^2 \Longrightarrow T = N^2/(M^2D^2). \end{aligned}$$

Let the memory be $M = 2^{mem}$ and the number of strings of the form $(c||x)$ (where $c \in GF(2)^{(n-k)/2}$ is fixed) in the collected data be $D = 2^d$. This means we need to sample $2^{\lceil (n-k)/2 \rceil + d}$ consecutive keystream bits to collect this data. This is because the string c of length $\lceil (n-k)/2 \rceil$ should occur on average once in $2^{\lceil (n-k)/2 \rceil}$ keystream bits. Thus the attack complexity is $N^2/(M^2D^2) = 2^{2\lfloor (n+k)/2 \rfloor - 2(d+mem)}$. The pre-processing time is $N/D = 2^{\lfloor (n+k)/2 \rfloor - d}$. We state the result formally as:

Theorem 2. *Consider an n-bit LFSR filtered by equation (1) where bit i of the LFSR is the ith input of $f(x)$. The LFSR initial state can be found with complexity $2^{2\lfloor (n+k)/2 \rfloor - 2(d+mem)}$ by using $2^{\lfloor (n+k)/2 \rfloor - d}$ pre-processing and 2^{mem} memory when $2^{\lceil (n-k)/2 \rceil + d}$ consecutive output bits are known.*

Remark 2. *The method we use to generate the functions $F_j(x)$ from an LFSR is by Mukhopadyay and Sarkar [16]. The method suggested in [2, 15] (and originally by Hellman in [10]) is to generate $F_j(x)$ by permuting the output bits of $F^{(c)}(x)$. But it was shown by Fiat and Naor that there exist search functions which are polynomial time indistinguishable from a random function but for which the time-memory trade-off attack fails [8], when permutation of output bits are used. The advantage of the approach of [16] is that it is not possible to construct a Fiat-Naor type example for the LFSR-based method. Moreover, LFSR sequences are very efficient to compute.*

Remark 3. *The derivation of the Rainbow attack on Maiorana-McFarland function based stream ciphers in [12, Section 3] is erroneous based on the computations in [3, Section 10]. The derivation of the complexity of time-memory-data attack in [12, Remark 2] is also erroneous. Here, we present the corrected application of the time-memory-data trade-off attack on Maiorana-McFarland based stream ciphers.*

Remark 4. *To obtain 2^{16} 128-bit ciphertext blocks where the first 32 bits is a fixed pattern c in Example 1, we need to scan through 2^{48} keystream bits. This scanning complexity is not taken into account in the attack complexity 2^{32}, which only covers the search of the TMD table. Part of the reason for not mixing the two complexities is that searching the TMD table involves computing the function $F^{(c)}$ (by solving a linear system) which is more complex than scanning for a fixed pattern from the keystream. The same remark applies to Example 3, 4 and 5 later in the paper.*

Remark 5. *Note that we need to solve a linear system of size* $(n-k)/2 \times (n-k)/2$ *for each processing and pre-processing step. For most of the applications we encounter, this linear system is quite small (*$q \times q$ *bit system where* $32 \leq q \leq 64$*). For modern* 64*-bit processors, this system can be efficiently solved with complexity* q^2*. For larger linear systems, we need to take the complexity of Gaussian elimination into account.*

3.1 Condition for the Attack to Work

Let us denote the number of guessed bits by $guess = (n+k)/2$. By the matrix stopping rule, we have $pt^2 = 2^{guess}$ and the amount of memory used is $pt/2^d = 2^{mem}$ where we use 2^d data obtained from $2^{(n-k)/2+d}$ keystream bits.

This implies that $t = 2^{guess-(mem+d)}$ and that the number of tables is $t/2^d = 2^{guess-(mem+2d)}$. Because we need at least one table ($t/2^d \geq 1$) to launch the attack, the condition $d \leq (guess - mem)/2$ must be satisfied. An identical bound holds for our subsequent attacks in Sections 5, 6 and 7 where we replace the variable "$guess$" by the corresponding number of guessed bits in the attack of those Sections.

3.2 Comparison with Direct Time-Memory-Data Trade-Off Attack

Note that if we attack the cipher of the previous remark directly using the TMD attack [2] without guessing any LFSR bits, we will get the same trade-offs when we take $D = 2^{(n-k)/2+d}$ and 2^{mem} memory, i.e. the same keystream length and memory which were used.

This is because we have $pt^2 = 2^n$ and $pt/2^{(n-k)/2+d} = 2^{mem}$. We deduce $t = 2^{(n+k)/2-(mem+d)}$, pre-processing $P = 2^n/2^{(n-k)/2+d} = 2^{(n+k)/2-d}$ and attack complexity $T = t^2 = 2^{(n+k)-2(mem+d)}$.

However, the number of tables is now given by $t/2^{(n-k)/2+d} = 2^{k-(mem+2d)}$. Thus it gives the upper bound $d \leq (k - mem)/2$ for the attack to be consistent (i.e. to have at least one table). Because we always have $k < n$, and consequently, $k < (n+k)/2 = guess$, it means that we get a smaller upper bound for d if we apply the TMD attack directly, and thus a smaller bound on the total keystream length $2^{(n-k)/2+d}$. In cases where this bound is exceeded, we will have to use a shorter keystream length in order to carry out the TMD attack of [2], which will lead to higher pre-computation and attack complexities. That is why it is preferable to use the TMD attack based on guessing LFSR bits. A similar reasoning holds for the other attacks in Sections 5, 6 and 7 of this paper.

4 On the Security of Different Maiorana-McFarland Functions against the TMD Attack

In general, the parameter k in the Maiorana-McFarland construction (equation 1) is in the range

$$1 \leq k \leq n/2.$$

4.1 The Case when k is Approximately $n/2$

Consider the extreme case $k \approx n/2$. There are many optimal functions belonging to this class as summarized in Proposition 1. In this case, $(n-k)/2 \approx n/4$ and the key diversity is reduced to $(n+k)/2 \approx 3n/4$ bits when $\approx n/4$ consecutive keystream bits are known. Suppose we collect $2^d = 2^{n/8}$ ciphertexts corresponding to a pre-computed $n/4$-bit pattern, i.e. $2^{3n/8}$ consecutive keystream bits. Then in a TMD attack with 2^{mem} memory, the complexity is $2^{5n/8}$ for pre-processing and $2^{5n/4-2mem}$ for the actual attack by Theorem 2. If n is not too big, it is reasonable to use $2^{mem} = 2^{n/2}$ memory which means the attack complexity is $2^{n/4}$. If we can obtain more keystream bits, then the pre-computation and attack complexity can be reduced further. We illustrate this case in Example 1.

Example 1. *To illustrate the cases of Sections 3.2 and 4.1 (this section), we apply Theorem 2 to Toyocrypt with the parameters* $n = 128$, $k = 64$, $d = 16$ *and* $mem = 64$.

For these parameters, the length of keystream is $\lceil \frac{n-k}{2} \rceil = \lceil \frac{128-64}{2} \rceil = 32$ *bits and the size of the search space* 2^{guess} *is* $2^{\lfloor \frac{n+k}{2} \rfloor} = 2^{\frac{128+64}{2}} = 2^{96}$. *Following the procedures described in Section 3, we randomly choose a string* $c \in GF(2)^{32}$. *This allows us to define the function* $F^{(c)}(x)$ *as in the setup phase described in Section 3. From Section 3.1, the number of tables required is* $t/D = 2^{guess-mem-2d} = 2^{96-64-2\times 16} = 2^0 = 1$. *We let* $F_1(x) = F^{(c)}(x)$ *since only 1 table is required. In the final step of the precomputation phase, we use* $F_1(x)$ *to build a* $p \times 2$ *array as in Step 3 in the Setup phase in Section 3, where* $p = 2^{guess}/t^2 = 2^{96}/(2^{16})^2 = 2^{64}$.

In the attack phase, we search amongst the keystream of length $2^{\lceil \frac{n-k}{2} \rceil + d} = 2^{32+16} = 2^{48}$ *to find* 2^{16} *substrings matching the pattern c. For each such string, we let the last* $(n+k)/2 = 96$ *bits of this string be y. Then for each y, we check if multiple applications of the search function* $F_1(y)$, *i.e.* $F_1(y)$, $F_1(F_1(y))$, *etc. match any end-points in the table. If so, then we have found the secret internal state of the corresponding data, which can be computed as in the description of the attack in Section 3.*

To summarise, the complexity of the time-memory-data trade-off attack for pre-processing and attack are

$$P = 2^{guess-d} = 2^{96-16} = 2^{80},$$

and

$$T = 2^{2(guess-(mem+d))} = 2^{2\times(96-(64+16)} = 2^{32},$$

respectively when we know 2^{48} *consecutive output bits. This attack is consistent because* $guess = (n + k)/2 = (128 + 64)/2 = 96$ *and the upper bound* $(guess - mem)/2 = (96 - 64)/2 = 16$ *on* d *is satisfied.*

We can seemingly achieve the same pre-processing and attack complexities of 2^{80} *and* 2^{32} *respectively with* 2^{48} *consecutive output bits and* 2^{64} *memory when we apply the TMD attack without any guessing, as was illustrated in Section 3.2. This is not possible in practice since the number of tables* t/D *calculated using these values is* $2^{64-(64+2\times16)} = 2^{-32}$*, and there has to be at least one table. For consistency, the inequality in Section 3.2 requires that* $d \leq (k - mem)/2 = (64 - 64)/2 = 0$*. If we have to carry out this attack, the maximum length of keystream we can use is* $2^{(n-k)/2+d} = 2^{(128-64)/2+0} = 2^{32}$*, which is much less than the* 2^{48} *we are provided with. For this keystream length, the pre-processing and attack complexities when provided with* 2^{64} *memory are* $N/D = 2^{128-32} = 2^{96}$ *and* $N^2/M^2D^2 = 2^{2\times(128-64-32)} = 2^{64}$ *respectively. This is much worse than the complexities obtained by applying the TMD attack based on guessing a part of the LFSR state bits.*

4.2 The Case when k is Much Smaller than n

The other extreme is when $k \ll n$. This scenario may occur when we use a saturated function from Proposition 2. In this case, k is much smaller than n and the key diversity $(n + k)/2$ can be taken to be approximately $n/2$. The minimum number of consecutive output bits $l = (n - k)/2$ needed is also approximately $n/2$. Suppose we collect $2^d = 1$ (where $d = 0$) ciphertext corresponding to a pre-computed $n/2$-bit pattern, i.e. we need $2^{n/2}$ consecutive keystream bits. Then in a time-memory-data trade-off attack using 2^{mem} memory, the complexity is $2^{n/2}$ for pre-processing and $2^{n-2mem-1}$ for the attack. Unlike the case $k \approx n/2$, we can use less memory here because the search space is smaller. If we use $2^{mem} = 2^{3n/8}$ memory, then the attack complexity is $2^{n/4}$.

Example 2. *To illustrate the case in this section, we use a Boolean function based on the construction in Proposition 2. We let* $d = 8$*, so that* $n = 134$ *and* $k = 7$*. Then the key diversity is* 2^{70} *and the minimum number of consecutive keystream bits needed is 64. We will, on average, encounter any particular 64-bit string once in every* 2^{64} *keystream bits. Suppose we just have one unit of data, i.e.* $d = 0$*, and* $2^{mem} = 2^{60}$ *memory. The attack is then set up the same manner as in Example 1. The complexities for pre-processing and attack are* 2^{70} *and* 2^{20} *respectively when we have* 2^{64} *keystream bits.*

Even though n *is slightly larger than that in Example 1, the memory, pre-computation and attack complexity are smaller.*

From the above discussion, we see that as k decreases, the memory, pre-computation and attack complexity decreases but the number of consecutive keystream bits needed increases. Sometimes it is not possible to obtain so many keystream bits for time-memory-data trade-off attack on equation 2. It may be more feasible to use Theorem 1 directly and perform an exhaustive search with complexity $2^{(n+k)/2}$ based on $(n-k)/2$ consecutive output bits.

5 When the LFSR and Boolean Functions have Different Sizes

As a generalization, we consider the above attack when an n-bit LFSR is filtered by a m-bit Maiorana-McFarland function $f(x)$ where $m < n$. Let the function be of the form

$$\begin{aligned} f(x_0, \ldots, x_{m-1}) &= g(x_0, \ldots, x_{r-1}) \\ &+ (x_r, \ldots, x_{m-1}) \cdot \phi(x_0, \ldots, x_{r-1}). \end{aligned} \tag{3}$$

where $f : GF(2)^m \to GF(2)$, $g : GF(2)^r \to GF(2)$ and $\phi : GF(2)^r \to GF(2)^{m-r}$.

Therefore the function $f(x)$ becomes linear when the first r input bits are fixed. Let these r input bits be tapped from amongst the leftmost k bits of the LFSR, and the remaining $m-r$ input bits of $f(x)$ be tapped from amongst the rightmost $n-k$ LFSR bits.

As before, assume $l = \lceil \frac{n-k}{2} \rceil$ consecutive output bits of $f(x)$ are known and we guess $n - l = \lfloor \frac{n+k}{2} \rfloor$ leftmost LFSR bits. Then we can form l linear equations with l unknown variables of the LFSR initial state. So knowing $\lceil (n-k)/2 \rceil$ consecutive output bits will reduce the initial state space from n bits to $n - l = \lfloor (n+k)/2 \rfloor$ bits. It is easy to see that we can apply the TMD attack as in Section 3 by using the same search function $F^{(c)}(x)$. The attack complexity for direct exhaustive search and TMD attack is the same as before but now, the parameter k depends not just on $f(x)$ but also on the tap points from the LFSR. We summarize our discussion as a theorem:

Theorem 3. *Consider an n-bit LFSR which is filtered by a m-bit Maiorana-McFarland function defined by equation (3). Suppose the first r bits of $f(x)$ is tapped from amongst the leftmost k bits of the LFSR, and the remaining $m-r$ input bits of $f(x)$ is tapped from amongst the rightmost $n-k$ LFSR bits. Then the keyspace is reduced from 2^n to $2^{\lfloor (n+k)/2 \rfloor}$ when $\lceil (n-k)/2 \rceil$ consecutive output bits are known.*

Furthermore, the LFSR initial state can be found with $2^{2\lfloor \frac{n+k}{2} \rfloor - 2(d+mem)}$ attack complexity, $2^{\lfloor (n+k)/2 \rfloor - d}$ pre-processing and 2^{mem} memory when $2^{\lceil (n-k)/2 \rceil + d}$ consecutive output bits are known.

We illustrate the attack with the following example.

Example 3. *Consider a 13-bit LFSR filtered by a 4-bit Boolean function $f(x)$.*

$$f(x_0, x_1, x_2, x_3) = g(x_0, x_1) + (x_2, x_3) \cdot \phi(x_0, x_1).$$

$f(x)$ becomes linear when we fix the first two bits x_0, x_1, i.e. $n = 13$, $m = 4$ and $r = 2$. At time i, let the output be y_i and the LFSR state be $(x_i, x_{i+1}, \ldots, x_{i+12})$. Number the positions of the LFSR by $0, \ldots, 12$ and let the tap points to (x_0, x_1, x_2, x_3) be $LFSR[0, 3, 5, 9]$. $f(x)$ is linear when the first two tap points 0 and 3 are known, so $k = 4$. Suppose we know $\lceil (n-k)/2 \rceil = 5$ consecutive output bits $y_1, \ldots, y_5$ and we guess $\lfloor (n+k)/2 \rfloor = 8$ consecutive LFSR bits $x_0, \ldots, x_7$. Form the equations:

$$y_i = f(x_i, x_{i+3}, x_{i+5}, x_{i+9}), i = 0, \ldots, 4.$$

The equations for $i = 0, 1, 2$ allows us to find x_9, x_{10}, x_{11}. The equation for $i = 3$ involves the unknown bits x_8 and x_{12}. The equation for $i = 4$ involves the unknown bit x_{13} which can be written as a linear function of the unknown bits x_8, x_{12} through the LFSR relation. Thus these two linear equations can be solved for x_8, x_{12}. Therefore we can reduce the complexity of the initial state space from 13-bit to 8-bit when 5 consecutive output bits are known.

After the search space reduction, we may apply the TMD attack but for this small example, it is easier to search directly.

6 Extending the Attack to Filter Combiner Model

In this section, we extend the search space reduction and TMD attack on the filter combiner model. For ease of explanation, we consider the case of two linear feedback shift registers $LFSR_1$ and $LFSR_2$. The attacks on more LFSR's are similar. At each clock cycle, a Boolean function will take as input several state bits from each of $LFSR_1$ and $LFSR_2$ to output a keystream bit.

Let the length of $LFSR_1$ be n_1 and that of $LFSR_2$ be n_2. Let $f : GF(2)^m \rightarrow GF(2)$ be defined by:

$$\begin{aligned} f(x_0, \ldots, x_{m-1}) &= (x_r, \ldots, x_{m-1}) \cdot \phi(x_0, \ldots, x_{r-1}) \\ &+ g(x_0, \ldots, x_{r-1}). \end{aligned} \tag{4}$$

Therefore when we fix the first r input bits, $f(x)$ becomes linear.

Let the first r input bits of $f(x)$, i.e. $(x_0, x_1, \ldots, x_{r-1})$ be tapped from amongst the leftmost k_1 and k_2 bits of $LFSR_1$ and $LFSR_2$. Let the rest of the $n - r$ input bits be tapped from amongst the rightmost $n_1 - k_1$ and $n_2 - k_2$ bits of $LFSR_1$ and $LFSR_2$.

Suppose we know l consecutive output bits $y_0, y_1, \ldots, y_{l-1}$. Let us guess the leftmost $k_1 + l$ and $k_2 + l$ bits of $LFSR_1$ and $LFSR_2$. Then at time i, $\phi(x_i, \ldots, x_{i+r-1})$, $g(x_i, \ldots, x_{i+r-1})$ are known for

all $i = 0, 1, \ldots, l-1$. This means:

$$\begin{aligned} y_i &= f(x_i, \ldots, x_{i+m-1}) \\ &= (x_{i+r}, \ldots, x_{i+m-1}) \cdot \phi(x_i, \ldots, x_{i+r-1}) \\ &+ g(x_i, \ldots, x_{i+r-1}). \end{aligned}$$

is a linear equation for $i = 0, 1, \ldots, l-1$.

We have l equations in $n_1 - (k_1 + l) + n_2 - (k_2 + l)$ variables. For this linear system to be solvable, we need

$$\begin{aligned} & n_1 - (k_1 + l) + n_2 - (k_2 + l) \leq l \\ \Longrightarrow \quad & l \geq \lceil ((n_1 - k_1) + (n_2 - k_2))/3 \rceil. \end{aligned}$$

We take $l = \lceil ((n_1 - k_1) + (n_2 - k_2))/3 \rceil$. Thus the search space is reduced from $2^{n_1+n_2}$ to:

$$2^{(k_1+l)+(k_2+l)} \approx 2^{(2(n_1+n_2)+(k_1+k_2))/3}.$$

The TMD attack can be applied for our scenario as follows. Let $l = \lceil ((n_1 - k_1) + (n_2 - k_2))/3 \rceil$, we define a function $\tilde{f} : GF(2)^{n_1} \times GF(2)^{n_2} \rightarrow GF(2)^{k_1+k_2+3l}$ to be:

$$\begin{aligned} \tilde{f}(\tilde{x_1}, \tilde{x_2}) \quad = \quad & \text{the } (k_1 + k_2 + 3l)\text{-bit} \\ & \text{output keystream} \\ & \text{when } (LFSR_1, LFSR_2) \\ & \text{are initialized by } (\tilde{x_1}, \tilde{x_2}). \end{aligned}$$

For a fixed string $c \in GF(2)^l$ and $x_i \in GF(2)^{k_i+l}$, we can find $s_i \in GF(2)^{n_i-(k_i+l)}$ such that $\tilde{f}(x_1||s_1, x_2||s_2) = (c||y)$ by the method described above. Based on this computation, we define a search function $F^{(c)} : GF(2)^{k_1+l} \times GF(2)^{k_2+l} \rightarrow GF(2)^{k_1+k_2+2l}$ to be the rightmost $k_1 + k_2 + 2l$ bits of $\tilde{f}(\tilde{x_1}, \tilde{x_2})$, i.e.

$$F^{(c)}(x_1, x_2) = y$$

Using this search function, we can perform a TMD attack as in Section 3. The search space is $N = 2^{k_1+k_2+2l}$. Assuming we have $M = 2^{mem}$ memory and we have 2^{l+d} consecutive keystream bits from which we can sample $D = 2^d$ ciphertext whose first l bits correspond to c. Then the preprocessing complexity is $N/D = 2^{k_1+k_2+2l-d}$ and attack complexity is $N^2/(M^2D^2) = 2^{2(k_1+k_2+2l-(d+mem))}$.

Example 4. *Let us consider a filter combiner generator where $LFSR_1$ and $LFSR_2$ have lengths $n_1 = 64 = n_2$. Let $f(x)$ be defined by equation (4) where $m = 64$ and $r = 32$. Let the first r bits of $f(x)$ be tapped from the leftmost k_1, k_2 bits of $LFSR_1$ and $LFSR_2$ where $k_1 = 16 = k_2$.*

The complexity of direct search without applying TMD attack is

$$2^{k_1+k_2+2l} = 2^{16+16+2\times 32} = 2^{96}.$$

where $l = \lceil((64-16)+(64-16))/3\rceil = 32$. *The complexity is less than the intended security of* $2^{n_1+n_2} = 2^{128}$.

Assuming we have $2^{mem} = 2^{64}$ *memory and* $2^{l+d} = 2^{48}$ *consecutive keystream bits where* $d = 16$. *The initial LFSR state can be recovered with* $2^{2(k_1+k_2+2l-(d+mem))} = 2^{32}$ *attack complexity and* $2^{k_1+k_2+2l-d} = 2^{80}$ *pre-processing. Thus the attack complexities are identical to Toyocrypt.*

In a similar way, the search space reduction and TMD attack of a filter combiner with s LFSR can be computed. We state this formally as:

Theorem 4. *Consider a filter combiner where equation (4) filters the content of* $LFSR_1, LFSR_2, \ldots, LFSR_s$ *of size* $n_1, n_2, \ldots, n_s$ *respectively. Let the first* r *bits of equation (4) be tapped from amongst the leftmost* k_i *bits of* $LFSR_i$. *And let the remaining* $m-r$ *bits be tapped from amongst the rightmost* $n_i - k_i$ *bits of* $LFSR_i$, $i = 1, 2, \ldots, s$.

Let $l = ((n_1-k_1)+\ldots+(n_s-k_s))/(s+1)$. *Then the key space of the filter combiner is reduced from* $2^{n_1+\ldots+n_s}$ *to* $2^{k_1+\ldots+k_s+s\times l}$ *when* l *consecutive output bits are known. Furthermore, the LFSR initial states can be found with* $2^{2(k_1+\ldots+k_s+s\times l-(d+mem))}$ *processing,* $2^{k_1+\ldots+k_s+s\times l-d}$ *pre-processing and* 2^{mem} *memory when* 2^{l+d} *consecutive keystream bits are known.*

7 Extending the Attack to Vectorial Maiorana-McFarland Functions

In this section, we consider the case where an n-bit LFSR is filtered by a vectorial Maiorana-McFarland functions $F : GF(2)^n \to GF(2)^m$ defined by:

$$\begin{aligned} &F(x_0, \ldots, x_{n-1}) \\ &= (f_0(x_0, \ldots, x_{n-1}), \quad \ldots \quad , f_{m-1}(x_0, \ldots, x_{n-1})) \end{aligned}$$

where each function $f_j : GF(2)^n \to GF(2)$ is defined by:

$$\begin{aligned} f_j(x_0, \ldots, x_{n-1}) &= (x_k, \ldots, x_{n-1}) \cdot \phi_j(x_0, \ldots, x_{k-1}) \\ &+ g_j(x_0, \ldots, x_{k-1}). \end{aligned}$$

for $j = 0, 1, \ldots, m-1$. This case may occur in practice because the encryption speed of a vector output generator is m times faster than a single bit filter function generator.

For good security, we want any linear combination of $f_j(x)$ to correspond to a t-resilient Maiorana-McFarland function with high nonlinearity. The usual method to construct $F(x)$ is to ensure that linear combinations of $f_j(x)$ correspond to concatenation of linear functions which are distinct and each linear function in the concatenation is an expression in $t+1$ or more variables. This can be achieved by using linear codes as shown in [17].

We assume bit i of the LFSR is the i-th input of $F(x)$. Suppose we know l consecutive output words, i.e. $l \times m$ consecutive output bits.

$$\begin{aligned}
&\text{word 1: } y_{0,0}, y_{0,1}, \ldots, y_{0,m-1} \\
&\text{word 2: } y_{1,0}, y_{1,1}, \ldots, y_{1,m-1} \\
&\ldots \\
&\text{word } l\text{: } y_{l,0}, y_{l-1,1}, \ldots, y_{l-1,m-1}
\end{aligned}$$

Let us guess the $k+l$ leftmost bits of the LFSR, i.e. $(x_0, x_1, \ldots, x_{k+l-1})$. Then $(x_i, x_{i+1}, \ldots, x_{i+k-1})$ is known at time $i = 0, 1, , \ldots, l-1$ and the following equations are linear.

$$\begin{aligned}
y_{i,0} &= g_0(x_i, \ldots, x_{i+k-1}) \\
&+ (x_{i+k}, \ldots, x_{i+n-1}) \cdot \phi_0(x_i, \ldots, x_{i+k-1}) \\
y_{i,1} &= g_1(x_i, \ldots, x_{i+k-1}) \\
&+ (x_{i+k}, \ldots, x_{i+n-1}) \cdot \phi_1(x_i, \ldots, x_{i+k-1}) \\
&\ldots \\
y_{i,m-1} &= g_{m-1}(x_i, \ldots, x_{i+k-1}) \\
&+ (x_{i+k}, \ldots, x_{i+n-1}) \cdot \phi_{m-1}(x_i, \ldots, x_{i+k-1})
\end{aligned}$$

We have $l \times m$ equations in $n - (k+l)$ unknowns. For this linear system to be solvable, we need:

$$\begin{aligned}
& n - (k+l) \leq l \times m \\
\Longrightarrow \quad & l \geq \lceil (n-k)/(m+1) \rceil.
\end{aligned}$$

We take $l = \lceil (n-k)/(m+1) \rceil$. Thus the search space is reduced from 2^n to $2^{k+l} = 2^{k+\lceil (n-k)/(m+1) \rceil}$.

The TMD attack can be applied for our scenario as follows. Let $l = \lceil (n-k)/(m+1) \rceil$, we define a search function $\tilde{F} : GF(2)^n \rightarrow GF(2)^{k+(m+1)l}$ to be

$$\begin{aligned}\tilde{F}(\tilde{x}) &= \text{the } (k+(m+1)l)\text{-bit output} \\ &\quad \text{keystream when the LFSR is} \\ &\quad \text{initialized by } \tilde{x} \in GF(2)^n.\end{aligned}$$

For a fixed string $c \in GF(2)^{ml}$ and $x \in GF(2)^{k+l}$, we can find $s \in GF(2)^{n-(k+l)}$ such that $\tilde{F}(x||s) = (c||y)$ by the method described above. Based on this computation, we define a search function $F^{(c)} : GF(2)^{k+l} \to GF(2)^{k+l}$ to be the rightmost $k+l$ bits of $\tilde{F}(x||s)$, i.e.,

$$F^{(c)}(x) = y$$

Using this search function, we can perform a TMD attack as in Section 3. The search space is $N = 2^{k+l}$. Assume we have $M = 2^{mem}$ memory and $m \times 2^{ml+d}$ consecutive keystream bits (from which we can sample $D = 2^d$ ciphertext whose first l m-bit words correspond to c). The slight modification is the extra factor of m, which is due to the fact that only bit strings starting at intervals of m in the keystream can be considered for "data", the output generator being a vectorial Boolean function. Thus the preprocessing complexity is $N/D = 2^{k+l-d}$ and attack complexity is $N^2/(M^2D^2) = 2^{2(k+l-(d+mem))}$.

Theorem 5. *Let $l = \lceil (n-k)/(m+1) \rceil$. Consider an n-bit LFSR filtered by equation (5) where bit i of the LFSR is the i-th input of $F(x)$. The key space is reduced from 2^n to 2^{k+l} when ml consecutive output bits are known.*

Furthermore, the LFSR initial states can be found with $2^{2(k+l-(d+mem))}$ processing, 2^{k+l-d} preprocessing and 2^{mem} memory when $m \times 2^{ml+d}$ consecutive keystream bits are known.

Remark 6. *We may also consider the case where the vector Maiorana-McFarland function has different size as the LFSR as in Section 5.*

Another extension is when the filter function in the filter combiner model is a vectorial Maiorana-McFarland function. In that case, the result is a combination of Theorem 4 and 5.

Example 5. *Consider the parameters in Toyocrypt where we have a 128-bit stream cipher filtered by a 128-bit vector Maiorana-McFarland function $F(x)$ with parameter $k = 64$ and m output bits. $F(x)$ may correspond to the vector function in Corollary 1 of [17] which is 1-resilient and has nonlinearity $2^{127} - 2^{64}$.*

Then by Theorem 5, $l = \lceil 64/(m+1) \rceil$ and the reduced search space is 2^{64+l}. Suppose we apply TMD attack with 2^d ciphertext whose first $m \times l$ bits correspond to a fixed string c. Then we need a keystream

Table 1: Reduced Search Space for $n = 128$, $k = 64$, Different Output Sizes m, and $d = 0$

Output Size m	1	2	3	4	5	6
Reduced Search Space	2^{96}	2^{86}	2^{80}	2^{77}	2^{75}	2^{74}
Consecutive Keystream Bits	32	44	48	52	55	60

Table 2: Complexities of TMD Attack for $n = 128$, $k = 64$ and Different Output Sizes m where Attack Complexity is fixed as 2^{32}.

Output Size m	1	2	3	4	5	6
Consecutive Keystream	2^{48}	2^{53}	$2^{57.6}$	2^{60}	$2^{63.3}$	$2^{68.6}$
Memory	2^{64}	2^{62}	2^{56}	2^{55}	2^{53}	2^{52}
Pre-processing Complexity	2^{80}	2^{78}	2^{72}	2^{71}	2^{69}	2^{68}
Attack Complexity	2^{32}	2^{32}	2^{32}	2^{32}	2^{32}	2^{32}
Number of Ciphertext 2^d	2^{16}	2^{8}	2^{8}	2^{6}	2^{6}	2^{6}

of length $m \times 2^{ml+d}$. If we have 2^{mem} memory, the pre-processing complexity is 2^{64+l-d} and the attack complexity is $2^{2(64+l-(d+mem))}$. These values are tabulated for different output size m in Table 1, 2.

In Table 1, we list the size of the reduced search space for different output sizes. We see that as m increases, the search space decreases.

In Table 2, we list the attack trade-offs for the time-memory-data trade-off attack. We fix the attack complexity as 2^{32} because this is considered sufficiently fast in practice. We see that as the number of output bits m increases, the memory and pre-processing complexities decrease while the amount of consecutive keystream bits needed increases.

8 Further Generalizations and Considerations

8.1 Further Generalizations of the Attacks

Some ways in which our attacks can be further generalized and prevented are as follows:

1. In Theorem 2 and 5, we have adopted the convention that the first k input bits of $f(x)$ are always tapped from the leftmost k bits of the LFSR. It is easy to see that the attacks have the same complexities if we tap any k consecutive bits of the LFSR.

2. Similarly, in Theorem 3, we can tap the first r input bits of $f(x)$ from any consecutive k bits of the LFSR. In Theorem 4, we can tap the r bits from any consecutive $k_1, \ldots, k_s$ bits of $LFSR_1, \ldots, LFSR_s$ respectively.

3. In our attacks, we have presented the TMD attack on Maiorana-McFarland functions because it is a well-known and common construction in the Boolean function literature. In that case, the function becomes linear when the leftmost k bits are known. To make the attack more general, we can look at any n-bit Boolean function which becomes linear when k (not necessarily consecutive) input bits are known.

8.2 Consequences on Design of Stream Ciphers

Consider the filter combiner in Theorem 4 that filters the contents of the s LFSRs. To prevent our attack, we can choose the k_i's such that the reduced attack complexity $2^{k_1+\ldots+k_s+s\times l}$ is not significantly better than exhaustive search complexity. For example, if $n_1 = n_2 = n_3 = n_4 = 32$ and we choose the tap points from the LFSR's to $f(x)$ such that $k_1 = 23$, $k_2 = 24$, $k_3 = 25$ and $k_4 = 26$, then $l = 6$. In this case the attack complexity is $2^{23+24+25+26+4\times 6} = 2^{122}$, which is not a significant improvement over the exhaustive search complexity of 2^{128}. Heuristically, to prevent our attack, one needs to

1. Choose the k_i's to be large enough so that the attacker has to guess almost the entire content of the LFSRs, i.e. the tap points should be spaced out wide enough.

2. Ensure the filter function $f(x)$ taps from as many of the s LFSRs as possible.

Please note that the above considerations only protect against the attacks in this paper. They have to be combined with other design criteria to protect against other cryptanalysis on stream ciphers.

9 Conclusion

We have generalized the Mihaljevic-Imai time-memory-data trade-off attack on Toyocrypt [15] to apply on any filter function generator using a Maiorana-McFarland function. We further explore different configurations when the filter function is smaller than the LFSR, when several LFSR's are used and when vector output functions are used. Further generalizations based on replacing LFSR's with other linear finite state machines, using different tap points and using more general Boolean functions are

proposed. We showed that the attack can be effectively applied in all these scenarios to significantly reduce the attack complexities. Because the Maiorana-McFarland function is a popular Boolean function construction, our attack may be a useful tool for stream cipher cryptanalysis.

There are many other stream cipher attacks which may be more effective than the TMD attack. For example, if the Boolean function has weak correlation properties, we may be able to apply correlation attack [4, 22]. When a certain multiple of the Boolean function has low degree, we can apply the algebraic attack [6]. When the stream cipher has a linear re-synchronization mechanism, we may apply the resynch-attack [7]. However, if the stream cipher is designed to protect against these attacks, and a Maiorana-McFarland function (a popular choice) is used, the TMD attack is a viable alternative.

References

[1] C. Adams, "The CAST-128 Encryption Algorithm", RFC 2144.

[2] A. Biryukov and A. Shamir, "Cryptanalytic Time/ Memory/Data Trade-offs for Stream Ciphers", LNCS 1976, *Asiacrypt 2000*, pp. 1-13, Springer-Verlag, 2000.

[3] A. Biryukov, S. Mukhopadhyay and P. Sarkar, "Improved Time-Memory Trade-Off with Multiple Data", LNCS 3897, *Selected Areas in Cryptography 2005*, pp. 110-127, Springer-Verlag, 2005.

[4] A. Canteaut and M. Trabbia, "Improved Fast Correlation Attack using Parity Check Equations of Weight 4 and 5", LNCS 1807, *Eurocrypt 2000*, pp. 573-588, Springer-Verlag, 2000.

[5] C. Carlet, "A Larger Class of Cryptographic Boolean Functions via a Study of the Maiorana-McFarland Construction", LNCS 2442, *Crypto'2002*, pp. 549-564, Springer-Verlag, 2002.

[6] N. Courtois and W. Meier, "Algebraic Attacks on Stream Ciphers with Linear Feedback", LNCS 2656, *Eurocrypt 2003*, pp. 345-359, Springer-Verlag, 2003.

[7] J. Daemon, R. Govaerts and J. Vandewalle, "Resynchronization weakness in Synchronous Stream Ciphers", LNCS 765, *Eurocrypt'93*, pp. 159-167, Springer-Verlag, 2004.

[8] A. Fiat and M. Naor, "Rigorous Time/Space Tradeoffs for Inverting Functions", *STOC 1991*, pp. 534-541, 1991.

[9] G. Gong and K. Khoo, "Additive Autocorrelation of Resilient Boolean Functions", LNCS 3006, *Selected Areas of Cryptography 2003*, pp. 275-290, Springer-Verlag, 2003.

[10] M. Hellman, "A Cryptanalytic Time-Memory Trade-Off", *IEEE Trans. on Information Theory*, vol. 26, pp. 401-406, 1980.

[11] K. Khoo and G. Gong, "New Constructions for Highly Nonlinear and Resilient Boolean Functions", LNCS 2727, *Australasian Conference on Information Security and Privacy 2003*, pp. 498-509, Springer Verlag, 2003.

[12] K. Khoo, G. Gong and H.K. Lee, " The Rainbow Attack on Stream Ciphers based on Maiorana-McFarland Functions, LNCS 3989, *Applied Cryptography and Network Security 2006*, pp. 194-209, Springer-Verlag, 2006.

[13] M. Matsui, "Linear cryptanalysis method for DES cipher", LNCS 765, *Eurocrypt'93*, pp. 386-397, 1994.

[14] W. Meier, E. Pasalic and C. Carlet, "Algebraic Attacks and Decomposition of Boolean Functions", LNCS 3027, *Eurocrypt 2004*, pp. 474-491, Springer-Verlag, 2004.

[15] M.J. Mihaljevic and H. Imai, "Cryptanalysis of Toyocrypt-HS1 Stream Cipher", IEICE Trans. Fundamentals, vol. E85-A no. 1, pp. 66-73, 2002.

[16] S. Mukhopadyay and P. Sarkar, "Application of LFSRs in Time/Memory Trade-Off Cryptanalysis", Indian Statistical Institute Technical Report No. ASD/04/9 (Revised Version), 2005.

[17] E. Pasalic and S. Maitra, "Linear Codes in Constructing Resilient Functions with High Nonlinearity", LNCS 2259, *Selected Areas in Cryptography 2001*, pp. 60-74, Springer-Verlag, 2001.

[18] P. Sarkar, "The Filter-Combiner Model for Memoryless Synchronous Stream Ciphers", LNCS 2442, *Crypto 2002*, pp. 533-548, Springer-Verlag, 2002.

[19] P. Sarkar and S. Maitra, "Nonlinearity Bounds and Constructions of Resilient Boolean Functions", LNCS 1880, *Crypto 2000*, pp. 515-532, Springer-Verlag, 2000.

[20] P. Sarkar and S. Maitra, "Construction of Boolean Functions with Important Cryptographic Properties", LNCS 1807, *Eurocrypt 2000*, pp. 485-506, Springer-Verlag, 2000.

[21] J. Seberry, X.M. Zhang and Y. Zheng, "On Constructions and Nonlinearity of Correlation Immune Functions", LNCS 765, *Eurocrypt'93*, pp. 181-199, 1994.

[22] T. Siegenthaler, "Decrypting a Class of Stream Ciphers using Ciphertexts only", *IEEE Transactions on Computers*, vol. C34, no. 1, pp. 81-85, 1985.

[23] Y. Tarannikov, "New Constructions of Resilient Boolean Functions with Maximum Nonlinearity", *Fast Software Encryption 2001*, LNCS 2355, pp. 66-77, Springer-Verlag, 2001.

[24] Y. Zheng and X.M. Zhang, "Improved Upper Bound on the Nonlinearity of High Order Correlation Immune Functions", *Selected Areas in Cryptography 2000*, pp. 262-274, LNCS 2012, Springer-Verlag, 2000.