A GENERAL FRAMEWORK FOR GUESS-AND-DETERMINE AND TIME-MEMORY-DATA TRADE-OFF ATTACKS ON STREAM CIPHERS

Guanhan Chew and Khoongming Khoo
DSO National Laboratories, 20 Science Park Drive, 118230, Singapore

Keywords: Guess-and-Determine Attack, Time-Memory-Data Trade-Off Attack, Stream Ciphers.

Abstract: In this paper, we present a framework for guess-and-determine attack on stream ciphers that relies on guessing part of the internal state and solving for the remaining unknown cipher state based on known keystream bits. We show that this basic attack can always be extended to a Time-Memory-Data (TMD) Trade-Off attack. This allows us to easily extend any guess-and-determine attack to a guess-and-determine TMD attack, which improves the online attack at the expense of memory, pre-processing time, and data requirement. Lastly, we illustrate three applications of the attack framework.

1 INTRODUCTION

In this paper, we study the guess-and-determine attack on stream ciphers. We will consider ciphers which satisfy the following attack criteria.

Attack Criteria. Assume we have a cipher with an n-bit internal state which maps its internal state to a fixed number of output bits at every clock. We will consider ciphers where we can guess k bits of the secret state and solve for the remaining $n-k$ secret state bits based on $l \geq n-k$ keystream bits (sometimes $l = n-k$ keystream bits will suffice).

We shall show how this basic attack can be extended to a guess-and-determine time-memory-data (GD-TMD) trade-off attack.

The time-memory-data (TMD) trade-off attack (Biryukov and Shamir, 2000) is an extension of the time-memory-trade-off (TMTO) attack of (Hellman, 1980). In the attack of (Biryukov and Shamir, 2000), the authors show that when the adversary has more data to work with, he can improve on (lessen) the online attack complexity and more importantly, lessen the pre-computation complexity (which is usually the hardest part of TMTO attacks).

Our GD-TMD attack is similar in principle to the usual TMD attack which aims to invert the mapping from internal state to output bits. However, the difference is that each evaluation step of a Hellman chain in GD-TMD consists of a mini guess-and-determine attack, as compared to a cipher evaluation in the normal TMD attack. We show that although our GD-TMD attack has the same complexity trade-offs as a TMD attack on the cipher with comparable memory and keystream resources, our attack allows much more optimal complexities to be achieved in practice when more data is available to the adversary. Our framework allows an adversary to convert any guess-and determine attack to a GD-TMD attack easily. This allows for improved TMD attacks with better complexities as opposed to applying the normal TMD attack of (Biryukov and Shamir, 2000) on the cipher directly.

Finally, we show how our framework can be applied in practice. As shown in Section 3, we convert the guess-and-determine attack on Toyocrypt (Mihaljevic and Imai, 2002) to a GD-TMD attack. In this case, the framework allows us to do the conversion easily without going through the substantial explanation given in (Mihaljevic and Imai, 2002). Second, the authors of (Khoo et. al., 2007) showed that when a Maiorana-McFarland function is used as a filter function in stream ciphers, guess-and-determine attacks and GD-TMD attacks can be performed on them. Two scenarios which are not considered in (Khoo et. al., 2007) are attacks on the vectorial filter combiner and combinatorial generator. We shall show how a guess-and-determine attack can be performed on them and how this can be easily converted to a GD-TMD attack for better online attack complexity.

Chew G. and Khoo K. (2008).
A GENERAL FRAMEWORK FOR GUESS-AND-DETERMINE AND TIME-MEMORY-DATA TRADE-OFF ATTACKS ON STREAM CIPHERS.
In *Proceedings of the International Conference on Security and Cryptography*, pages 300-305
DOI: 10.5220/0001918703000305

2 GUESS-AND-DETERMINE ATTACK

Assuming the cipher satisfies our attack criteria, let the minimum number of state bits that needs to be guessed, given l known keystream bits, be k_{min}. This minimum value can depend on n, l and properties of the cipher. We can, of course, choose to guess more than this minimum number. If we let k, such that $k_{min} \leq k \leq n$, be the number of internal state bits guessed in the attack, then the attack complexity T is

$$T = 2^k \times \mathcal{G}[n,l,k], \tag{1}$$

where function $\mathcal{G}[\cdot]$ is the complexity of solving the system of equations for each guess. The exact form of this function will differ from cipher to cipher.

3 GUESS-AND-DETERMINE TIME-MEMORY-DATA TRADE-OFF ATTACK

The attack time complexity in (1) can be improved upon when we apply the Time-Memory-Data Trade-Off attack of (Biryukov and Shamir, 2000), which is based on Hellman's original time-memory trade-off attack in (Hellman, 1980). Oeschlin's rainbow attack (Oeschlin, 2003) is not discussed here since the time-memory-data trade-off curve of the rainbow method has been shown in (Biryukov et. al., 2006) to be inferior to that for Hellman's method.

Our procedure is similar to that described in Theorem 2 of (Khoo et. al., 2007). Defining n, k and l as in the previous section, we choose a fixed string c such that $c \in GF(2)^l$. We define the function $f : GF(2)^n \rightarrow GF(2)^{k+l}$ so that

$f(\tilde{x}) = (k+l)$-bit output of the keystream generator when the internal state is initialized to $\tilde{x} \in GF(2)^n$.

Given $x \in GF(2)^k$, and c, we can solve for $u \in GF(2)^{n-k}$ such that the first l output keystream bits of the stream cipher when initialized to the state $(x||u)$ is the string c, where '$||$' denotes concatenation of bit strings. With this initial state, we can further generate k additional keystream bits y, so that $y \in GF(2)^k$. We define $(c||y)$ to be the keystream bits when the cipher is initialized to the state $(x||u)$:

$$f(x||u) = (c||y). \tag{2}$$

Let $D = 2^d$ be the amount of data. For our case, D is the number of occurences of the string c in our keystream. We define the search functions $F_i^{(c)} : GF(2)^k \rightarrow GF(2)^k$ for $i = 1, \ldots, t/D$ as follows:

$$F_i^{(c)}(x) = y \oplus X_i$$

where x and y are related through (2), and the X_i's are distinct randomly generated bit strings in $GF(2)^k$. This search function will be used to compute the TMD tables, each with length t, which will be used to conduct an attack similar to that described in (Khoo et. al., 2007).

We define m to be the total number of rows needed in a table if we were to carry out a Time-Memory (TM) Trade-Off attack as described in (Hellman, 1980). The entire state space N is equal to mt^2 by the matrix stopping rule (Hellman, 1980). Since we are performing a TMD attack, this search space can be reduced to mt^2/D according to the amount of data D that we have. According to (Biryukov and Shamir, 2000), we shall use t/D tables each of size $m \times t$. Since we are only storing the start points and end points, each table will only require m units of memory. Thus the amount of memory M required for the t/D tables is $M = mt/D$. For every one of the D data points, computing all t iterations of the t/D search functions would require approximately $t \times t/D \times \mathcal{G}[n,l,k]$ units of processing time. Since we have D units of data, the total online attack time T is $t^2 \times \mathcal{G}[n,l,k]$. Substituting these parameters into the trade-off equation gives:

$$\begin{aligned} TM^2D^2 &= t^2 \times \mathcal{G}[n,l,k] \times (mt/D)^2 \times D^2 \\ &= m^2t^4 \mathcal{G}[n,l,k] \\ &= N^2 \mathcal{G}[n,l,k] \\ &= 2^{2k} \mathcal{G}[n,l,k], \end{aligned}$$

where we have made use of $N = mt^2$ to eliminate the m's and t's and derive the trade-off curve equation $TM^2D^2 = N^2 \mathcal{G}[n,l,k]$.

During pre-processing, t evaluations of functions $F_i^{(c)}$ need to be made for every row, with a complexity of $\mathcal{G}[n,l,k]$ for each evaluation. Thus the complexity in building t/D tables of m rows is $t \times \mathcal{G}[n,l,k] \times t/D \times m = mt^2/D \times \mathcal{G}[n,l,k] = N/D \times \mathcal{G}[n,l,k]$.

We can summarise the previous discussion with the following results:

$$\begin{aligned} \text{Memory} &= M = 2^{mem} \\ \text{Data} &= D = 2^d \\ \text{Pre-processing Complexity} &= N/D \times \mathcal{G}[n,l,k] \\ &= 2^{k-d} \times \mathcal{G}[n,l,k] \\ \text{Length of constant string } c &= l \\ \text{Online Attack Complexity} &= T \\ &= 2^{2(k-(d+mem))} \\ &\times \mathcal{G}[n,l,k] \end{aligned} \tag{3}$$

In deriving the above result, we need to ensure that conditions implicit in the derivation of TMD Trade-Off curve are satisfied, i.e. the amount of memory

M and the data available D should not be arbitrarily chosen even when they appear to satisfy $TM^2D^2 = N^2$. For example, suppose $N = 2^{96}$ and in our attack, we choose $M = 2^{80}$ and $D = 2^{10}$. Although the pre-processing complexity and online attack complexity of 2^{86} and 2^{12} seem valid, the number of tables t/D, as defined previously, is less than 1. Using $M = mt/D$ and $N = mt^2$, we get $t/D = N/(MD^2) = 2^{k-mem-2d} = 2^{96-80-20} = 2^{-4} < 1$.

For stream ciphers whose internal state maps to one keystream bit after every clock, the amount of consecutive keystream bits corresponding to the above parameters is 2^{l+d}. This is because the fixed string c of length l will occur, on average, once in 2^l consecutive keystream bits. If there are 2^d data, this implies that the total keystream length is 2^{l+d}. Using a similar reasoning, for stream ciphers whose internal state maps to μ keystream bits after every clock, the amount of consecutive keystream bits is $\mu \times 2^{l+d}$.

4 ADVANTAGES OF GD-TMD OVER NORMAL TMD ATTACK

In this section, we compare the guess-and-determine TMD attack from the last section with a normal TMD attack (Biryukov and Shamir, 2000) based on the same amount of memory and 2^{l+d} consecutive keystream bits, where $l \approx n-k$. We consider a stream cipher whose internal state maps to one keystream bit after every clock. In this case, the amount of data is equal to the length of keystream for the normal TMD attack.

For Normal TMD Attack:

$$mt^2 = 2^n \text{ and } mt = 2^{mem} \times 2^{d+n-k},$$

which implies:

$$t = 2^{k-(mem+d)} \text{ and } m = 2^{2(mem+d-k)+n}.$$

In the pre-computation phase, $t/2^{d+n-k} = 2^{k-mem-(n-k)-2d}$ tables of size $m = 2^{2(mem+d-k)+n}$ are stored. For at least one table to be computed, we need:

$$\begin{aligned} & 2k - mem - n - 2d \geq 0 \\ \Longrightarrow\ & d \leq (2k - mem - n)/2 \\ =\ & (k - mem - (n-k))/2. \end{aligned} \tag{4}$$

The attack complexity of the Normal TMD attack is $2^{2(k-(d+mem))}$ encryptions.

For Guess-and-Determine TMD Attack:

$$mt^2 = 2^k \text{ and } mt = 2^{mem} \times 2^d,$$

which implies:

$$t = 2^{k-(mem+d)} \text{ and } m = 2^{2(mem+d)-k}.$$

In the pre-computation phase, $t/2^d = 2^{k-mem-2d}$ tables of size $m = 2^{2(mem+d)-k}$ are stored. For at least one table to be computed, we need:

$$k - mem - 2d \geq 0 \Longrightarrow d \leq (k - mem)/2. \tag{5}$$

The attack complexity for the GD-TMD attack is $2^{2(k-(d+mem))} \times \mathcal{G}[n,l,k]$. We see that even though this attack complexity may be slightly higher than that for the normal TMD attacks, this attack has two advantages over the normal TMD attack:

1. **Larger Range of Values for d.** Based on the inequalities in (4) and (5), the GD-TMD attack has a larger upper bound for d. This allows us to utilize a larger amount of data, should it become available, and in so doing, achieve a lower overall attack complexity.

2. **Smaller Memory Requirement.** Another advantage of the GD-TMD attack is that we only need to store k-bit words as compared to storing n-bit words in the normal TMD attack. Thus the GD-TMD attack only uses k/n of the memory required in the normal TMD attack.

5 APPLICATIONS

5.1 Attack on the Toyocrypt Cipher

The Toyocrypt stream cipher is Cryptrec submission for the Japanese government initiative to find cryptographic standards. It is based on a 128-bit modular feedback shift register (MLFSR) filtered by a balanced 128-bit boolean function of degree 64 with high nonlinearity. The MLFSR is a linear state machine where the feedback function correspond to a triangular matrix. In (Mihaljevic and Imai, 2002), the authors described a basic guess-and-determine attack where they make use of the property that if 64 input bits of the filter function $f(x)$ is known, then $f(x)$ becomes a linear function. They guess 96 of the input bits and obtain 32 linear equations from 32 keystream bits. This linear system can be solved to obtain the 32 remaining unknown bits.

By applying the Guess-and-determine TMD attack (GD-TMD) of Section 3, we easily get the following trade-offs:

$$\begin{array}{rcl}
\text{Memory} & = & M = 2^{mem} \\
\text{Data} & = & D = 2^{d} \\
\text{Pre-processing Complexity} & = & N/D \\
 & \times & \mathcal{G}[n,l,k] \\
 & = & 2^{96-d} \\
 & \times & \mathcal{G}[128,32,96] \\
\text{Length of constant string } c & = & 32 \\
\text{Online Attack Complexity} & = & T \\
 & = & 2^{2(96-d-mem)} \\
 & \times & \mathcal{G}[128,32,96],
\end{array}$$

where $t/2^d = 2^{96-mem-2d}$ tables of size $m = 2^{2(mem+d)-96}$ are stored. The amount of keystream required is 2^{l+d} since the cipher maps the internal state to one keystream bit at every clock. In comparison, a substantial part of (Mihaljevic and Imai, 2002) is devoted to explaining how the guess-and-determine attack can be converted into a TMD attack.

If we choose $d = 16$ and $mem = 64$, then we have the following attack complexities:

$$\begin{array}{rcl}
\text{Memory} & = & M = 2^{64} \\
\text{Data} & = & D = 2^{16} \\
\text{Pre-processing Complexity} & = & N/D \times \mathcal{G}[n,l,k] \\
 & = & 2^{80} \times \mathcal{G}[128,32,96] \\
\text{Length of constant string } c & = & 32 \\
\text{Online Attack Complexity} & = & T \\
 & = & 2^{2(96-16-64)} \\
 & \times & \mathcal{G}[128,32,96] \\
 & = & 2^{32} \times \mathcal{G}[128,32,96]
\end{array}$$

where $2^{96-64-2\times16} = 1$ table of size $2^{2(64+16)-96} = 2^{64}$ is stored and the amount of keystream required is $2^{32+16} = 2^{48}$. The same attack complexity is obtained in (Mihaljevic and Imai, 2002). Note that the authors assumed that the complexity $\mathcal{G}[128,32,96]$, which is that of solving a 32×32 linear system is negligible. In this case, $\mathcal{G}[128,32,96]$ can be taken as $32^2/64 = 2^4$ where we consider Gaussian elimination on a 64-bit processor. This is because we can consider the XOR of two 64-bit words as 1 operation.

The preprocessing complexity is 2^{84} while the overall online attack complexity is 2^{36}.

5.1.1 Comparison with Normal TMD Attack

For comparison, we consider the scenario in Section 4 where we apply the normal TMD with 2^{32+d} data, on the Toyocrypt cipher with 128-bit key.

$$\begin{array}{rcl}
\text{Memory} & = & M = 2^{mem} \\
\text{Data} & = & D = 2^{32+d} \\
\text{Pre-processing Complexity} & = & N/D \\
 & = & 2^{128-(32+d)} \\
 & = & 2^{96-d} \\
\text{Online Attack Complexity} & = & T \\
 & = & 2^{2(128-(32+d)-mem)} \\
 & = & 2^{2(96-d-mem)},
\end{array}$$

where $t/2^{32+d} = 2^{64-mem-2d}$ tables of size $m = 2^{2(mem+d)-64}$ are stored.

We see that the trade-off for memory, preprocessing and attack complexities are exactly the same for both the normal TMD and GD-TMD attacks. The only difference is the number and size of the stored tables. For at least a table to be formed in the normal TMD attack, we need $64 - mem - 2d \geq 0$ which implies $d \leq (64 - mem)/2$.

Assume we have 2^{48} keystream bits and $M = 2^{mem} = 2^{64}$ memory at our disposal. For at least one table to be formed, we need $d \leq (64 - mem)/2 = 0$, which implies we can only use at most $D = 2^{32+d} = 2^{32+0} = 2^{32}$ keystream bits. In that case, the attack complexity is $2^{2(96-d-mem)} = 2^{2(96-0-64)} = 2^{64}$. This is still worse than the attack complexity of 2^{36} which we obtained by the GD-TMD attack for the same scenario. Furthermore, since $d = 0$, the preprocessing complexity is 2^{96} which is also much larger than that of the GD-TMD attack.

5.2 Attack on the Filter Combiner based on Vectorial Maiorana-McFarland Functions

The guess-and-determine attack on the filter combiner based on Maiorana-McFarland equations has been studied in (Khoo et. al., 2007). In this Section, we generalize this attack to one on the filter combiner with vectorial output. Using our framework, we easily convert it to a TMD attack for better online attack complexity.

We consider the case where 2 LFSRs — LFSR$_1$ and LFSR$_2$ — of lengths n_1 and n_2 respectively (so that $n = n_1 + n_2$), are filtered by a vectorial Maiorana-McFarland function $F : GF(2)^n \to GF(2)^\mu$ defined by:

$$\begin{array}{rl}
 & F(x_0,\ldots,x_{n-1}) \\
= & (f_0(x_0,\ldots,x_{n-1}),\ldots,f_{\mu-1}(x_0,\ldots,x_{n-1}))
\end{array}$$

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

$$\begin{aligned} & f_j(x_0,\ldots,x_{n-1}) \\ = \; & (x_r,\ldots,x_{n-1})\cdot\phi_j(x_0,\ldots,x_{r-1}) \\ + \; & g_j(x_0,\ldots,x_{r-1}), \end{aligned} \tag{6}$$

for $j=0,\ldots,\mu-1$. The functions ϕ_j and g_j are defined by $\phi_j : GF(2)^r \rightarrow GF(2)^{n-r}$ and $g_j : GF(2)^r \rightarrow GF(2)$. The symbol '$\cdot$' denotes the usual dot product. The inputs to these functions $x_0,\ldots,x_{r-1}$, which are not necessarily adjacent nor indexed in the order in which they appear on the LFSRs, are tapped from amongst the bits of both LFSRs. Vectorial Maiorana-McFarland functions are used because they have good cryptographic properties (see (Pasalic and Maitra, 2001; Carlet, 2002))

Assuming we guess enough bits in LFSRs so that the inputs to the ϕ_j and g_j functions for p consecutive clocks are completely determined, and that the keystream bits corresponding to these clocks are known, then (6) is linearized and can be solved via Gaussian elimination, if enough keystream bits are available to form a full-ranked system of linear equations. If the number of bits guessed in LFSR$_1$ and LFSR$_2$ are k_1 and k_2 respectively, and the number of known keystream bits is $l = p\times\mu$, then the time complexity of this attack as described by Equation 1, is $2^{k_1+k_2}\times\mathcal{G}[n,p\times\mu,k_1+k_2]$, where the function $\mathcal{G}[\cdot]$ represents the complexity of Gaussian elimination for each guess. For the resulting system of linear equations to be solvable, we require that the number of equations be at least as many as the number of unknowns, i.e. $p\times\mu \geq n_1+n_2-(k_1+k_2)$, which is equivalently $k_1+k_2 \geq n_1+n_2-\mu\times p$. The parameters k_1 and k_2 also need to satisfy some minimum bound so that the inputs to the functions ϕ_j and g_j are determined, i.e. $k_1 \geq k_{1,min}$ and $k_2 \geq k_{2,min}$ for some $k_{1,min} \geq r$ and $k_{2,min} \geq r$ that depend on the positions of the tap points.

Suppose that $n_1 = 33$, $n_2 = 31$, $\mu = 3$ and $r = 5$. Let these 5 bits that are to be guessed be tapped from amongst the leftmost 10 bits and 7 bits of LFSR$_1$ and LFSR$_2$ respectively. Both LFSRs clock in the left direction. Hence the minimum number of bits $(k_{1,min}, k_{2,min})$ we can guess for this attack to be effective are 10 from LFSR$_1$ and 7 from LFSR$_2$. Depending on the number of keystream bits we have and the attack complexity we would like to achieve, the total number of guessed bits k_1 and k_2 can vary according to the conditions stipulated previously. Suppose that we guess the leftmost 19 and 16 bits of LFSR$_1$ and LFSR$_2$ respectively. Then the total number of unknown bits is $33-19+31-16 = 29$. Furthermore, the leftmost 10 and 7 bits of LFSR$_1$ and LFSR$_2$ are determined for $p = 10$ consecutive clocks. If we know the $10\times 3 = 30$ keystream bits corresponding to these clocks, we can solve for the un-guessed bits since we have more linear equations than unknowns.

Furthermore, if we have more keystream bits, the attack can be extended to a TMD Trade-Off Attack. The constant string length l used in the TMD Trade-Off Attack needs to be fixed at the length of keystream for which the Guess-and-Determine attack was conducted. For this attack, we need to find at least one occurrence of the chosen string, which can only be taken at intervals of μ keystream bits. This implies that we need at least about $\mu\times 2^l$ known keystream bits. The time complexity and other parameters are as derived in Equation (3).

Suppose we choose $M = 2^{15}$ and $D = 2^5$ for the above scenario, then the TMD attack complexity is $2^{33.72}$ with a preprocessing time of $2^{33.72}$. The factor $\mathcal{G}[64,30,35]$ can be taken to be $29^2/64 \approx 2^{3.72}$ if we consider Gaussian elimination of 29 unknown variables on a 64-bit processor. The amount of keystream bits required is $3\times 2^{\mu\times p+d} = 3\times 2^{3\times 10+5} = 3\times 2^{35}$. To summarise,

$$\begin{aligned} \text{Memory} &= M = 2^{15} \\ \text{Data} &= D = 2^5 \\ \text{Pre-processing Complexity} &= N/D\times\mathcal{G}[n,l,k] \\ &= 2^{35-5} \\ &\times \mathcal{G}[64,30,35] \\ &= 2^{33.72} \\ \text{Online Attack Complexity} &= T \\ &= 2^{2(35-15-5)} \\ &\times \mathcal{G}[n,l,k] \\ &= 2^{33.72}. \end{aligned}$$

In this case, $t/2^d = 2^{35-15-2\times 5} = 2^{10}$ tables of size $m = 2^{2(15+5)-35} = 2^5$ are stored.

5.3 Attack on the Combinatorial Generator based on Maiorana-McFarland Functions

We consider a combinatorial generator where n LFSRs, each of length approximately[1] L, are filtered by the following Maiorana-McFarland function to produce 1 keystream bit at each clock.

$$\begin{aligned} & f(x_0,\ldots,x_{n-1}) \\ = \; & (x_r,\ldots,x_{n-1})\cdot\phi(x_0,\ldots,x_{r-1}) \\ + \; & g(x_0,\ldots,x_{r-1}). \end{aligned} \tag{7}$$

[1]We choose the lengths of the LFSRs to be approximately equal but mutually coprime to obtain a large combined period. The derived trade-off values are approximate.

In this setup, one bit from each of the n LFSRs is tapped into the input of the function f. The functions ϕ and g are defined by $\phi : GF(2)^r \to GF(2)^{n-r}$ and $g : GF(2)^r \to GF(2)$. Each of the r bits in these functions are tapped from the first r LFSRs.

We see that if we guess the content of the first r LFSRs, then equation (7) becomes linear and the content of the $n-r$ remaining LFSR's can be obtained by linear algebra based on $(n-r)L$ keystream bits. The complexity of this guess-and-determine attack is $2^{rL} \times \mathcal{G}[nL,(n-r)L,rL]$ where $\mathcal{G}[nL,(n-r)L,rL] = ((n-r)L)^2/64$ (assuming we do Gaussian elimination on a 64-bit machine).

By applying the Guess-and-determine TMD attack (GD-TMD) of Section 3, we easily get the following trade-offs:

$$\begin{aligned}
\text{Memory} &= M = 2^{mem} \\
\text{Data} &= D = 2^d \\
\text{Pre-processing Complexity} &= 2^{nL}/D \\
&\times \mathcal{G}[nL,(n-r)L,rL] \\
&= 2^{nL-d} \\
&\times ((n-r)L)^2/64 \\
\text{Length of constant string } c &= rL \\
\text{Online Attack Complexity} &= T \\
&= 2^{2(rL-d-mem)} \\
&\times ((n-r)L)^2/64,
\end{aligned}$$

where $t/2^d = 2^{rL-mem-2d}$ tables of size $m = 2^{2(mem+d)-rL}$ are stored. To form at least one table, we need $rL - mem - 2d \geq 0$ which implies $d \leq (rL-mem)/2$. Since this cipher maps the internal state to one keystream bit at each clock, the amount of keystream bits required is 2^{rL+d}.

6 CONCLUSIONS

We have presented the general framework for Guess-and-Determine attacks on stream ciphers that satisfy certain attack criteria. This approach has been extended to that of a Time-Memory-Data Trade-Off attack. Results for the complexities and other attack parameters have been derived. We conclude by illustrating this framework with attacks on the Toyocrypt cipher, the vectorial filter combiner and the combinatorial generator.

We note that in all our examples, our attack framework is applied on stream ciphers involving a filter function which becomes linear when certain input bits are guessed. Other stream ciphers on which our attack framework may be applied include clock-controlled stream ciphers. In some of these ciphers, guessing the key bits which control the irregular clocking may cause the keystream equations to be linear. Then the remaining key bits can be solved by Gaussian elimination. By applying the result of Section 3, we can easily convert these attacks to a GD-TMD attack on clock controlled stream ciphers.

REFERENCES

A. Biryukov, S. Mukhopadhyay and P. Sarkar (2006). Improved time-memory trade-offs with multiple data. In *LNCS 3897,* Selected Areas in Cryptography 2005, *pp. 110-127*. Springer-Verlag.

Biryukov, A. and Shamir, A. (2000). Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *LNCS 1976,* ASIACRYPT 2000, *pp. 1-13*. Springer-Verlag.

Carlet, C. (2002). A larger class of cryptographic boolean functions via a study of the Maiorana-McFarland construction. In *LNCS 2442,* Crypto'2002, *pp. 549-564*. Springer-Verlag.

Hellman, M. (1980). A cryptanalytic time-memory trade-off. In *IEEE Trans. on Information Theory, vol. 26, pp.401-406*.

K. Khoo, G. Gong. and H.K. Lee. (2006). The rainbow attack on stream ciphers based on Maiorana-McFarland functions. In *LNCS 3989,* Applied Cryptography and Network Security 2006, *pp. 194-206 (Corrected version of this paper can be found in (Khoo et. al., 2007))*. Springer-Verlag.

K. Khoo, G. Gong, H.K. Lee and G. Chew (2007). The time-memory-data trade-off attack on stream ciphers based on Maiorana-McFarland functions. In *Cryptology ePrint Archive Report 2007/242, found at 'http://eprint.iacr.org/2007/242'. (Corrected version of (Khoo et. al., 2006))*.

Mihaljevic, M. and Imai, H. (2002). Cryptanalysis of Toyocrypt-HS1 stream cipher. In *IEICE Trans. Fundamentals, vol. E85-A no. 1, pp. 66-73*.

Oeschlin, P. (2003). Making a faster cryptanalytic time-memory trade-off. In *LNCS 2729,* CRYPTO 2003, *pp. 617-630*. Springer-Verlag.

Pasalic, E. and Maitra, S. (2001). Linear codes in constructing resilient functions with high nonlinearity. In *LNCS 2259,* Selected Areas in Cryptography 2001, *pp. 60-74*. Springer-Verlag.