Invertibility of multiple random functions and its application to symmetric ciphers*

Xiutao FENG[1]✉, Xiaoshan GAO[1], Zhangyi WANG[2] and Xiangyong ZENG[3]

[1] Key Laboratory of Mathematics Mechanization, Academy of Mathematics and System Science, Chinese Academy of Sciences, Beijing 100190, China
[2] School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
[3] Faculty of Mathematics and Statistics, Hubei Key Laboratory of Applied Mathematics, Hubei University, Wuhan 430062, China
fengxt@amss.ac.cn

Abstract. The invertibility of a random function (IRF, in short) is an important problem and has wide applications in cryptography. For example, searching a preimage of Hash functions, recovering a key of block ciphers under the known-plaintext-attack model, solving discrete logarithms over a prime field with large prime, and so on, can be viewed as its instances. In this work we describe the invertibility of multiple random functions (IMRF, in short), which is a generalization of the IRF. In order to solve the IMRF, we generalize the birthday theorem. Based on the generalized birthday theorem and time-memory tradeoff (TMTO, in short) method, we present an efficient TMTO method of solving an IMRF, which can be viewed as a generalization of three main TMTO attacks, that is, Hellman's attack, Biryukov and Shamir's attack with BSW sampling, and Biryukov, Mukhopadhyay and Sarkar's time-memory-key tradeoff attack. Our method is highly parallel and suitable for distributed computing environments. As a generalization of Hellman's attack, our method overcomes its shortcoming of using only one pair of known plaintext and ciphertext and first admits more than one datum in a TMTO on block ciphers at the single key scenario. As a generalization of Biryukov and Shamir's attack with BSW sampling, our method overcomes its shortcoming of using only a few data with specific prefix in stream ciphers and can utilize all data without any waste. As applications, we get two new tradeoff curves: $N^2 = TM^2D^3$, $N = PD$ and $D = \tau$ for block ciphers, and $N^2 = \tau^3TM^2D^2$, $N = \tau PD$ and $D \geq \tau$ for stream ciphers, where τ is the number of random functions, that is, the number of independent computing units available to an attacker, N is the size of key space (for block ciphers) or state (for stream ciphers) space, D the number of data captured by the attacker, and T, M, P the time/memory/precomputation cost consumed at each computing unit respectively. As examples, assume that 4096 computing units can be available for the attacker. Denote by 5-tuple (τ, T, M, D, P) the cost

* This work was supported by National Natural Science Foundation (Grant 61572491, 61972297) and National Key Research and Development Project(Grant 2018YFA0704705).

of our method. Then the cost of breaking DES, AES-128 and A5/1 is $(2^{12}, 2^{25.3}, 2^{25.3}, 2^{12}, 2^{44})$, $(2^{12}, 2^{73.3}, 2^{73.3}, 2^{12}, 2^{116})$ and $(2^{12}, 2^{22.7}, 2^{17.3}, 2^{17.3}, 2^{34.7})$ respectively.

Keywords: Random function, TMTO, block cipher, stream cipher, guess-and-determine attack

1 Introduction

Let n be a positive integer and $\{0,1\}^n$ be the set of all bit strings of length n. For a given random function f over the set $\{0,1\}^n$ and m images $y_1, y_2, \cdots, y_m$ of f, where m is a positive integer, how to find a preimage x of some y_i under f, i.e., $y_i = f(x)$, is an important and fundamental problem in cryptography [1], which is called *an invertibility of a random function* (IRF, in short) and has very wide applications. For example, searching a preimage of Hash functions, recovering a key of block ciphers under the known-plaintext-attack model, solving discrete logarithms over a prime field with large prime [2], and so on, can be viewed as its instances.

The time-memory tradeoff (TMTO, in short) is a common method of solving the IRF. It is mainly based on the following *birthday theorem*:

Lemma 1. *[3]* **(Birthday Theorem)** *Let S be a finite set, A and B be two nonempty subsets of S such that $|A| \cdot |B| \geq |S|$. Then $A \cap B$ is not empty with large probability.*

The TMTO of solving an IRF usually involves two stages: offline and online. At the one-time offline stage, one needs to precompute and store some pairs $(x, f(x))$ of preimage and image, which is viewed as the set A in Lemma 1. A common treatment is to set up a lookup table. And at the online stage, he/she will match each y_i directly in the second column of the lookup table, where y_i's are viewed as elements of the set B. If $|A| \cdot m \geq 2^n$, then he/she will obtain a preimage x of some y_i with large probability according to Lemma 1.

How to set up a more efficient lookup table is a key problem in the TMTO. Currently, many efficient setting-up-table techniques have been proposed, including Hellman's table with distinguished points [4], perfect table [5], rainbow table [6] and so on. Among them, the rainbow table is believed to be one of the most efficient methods.

The TMTO was first used in symmetric ciphers in 1980. Hellman [7] first applied it to the security evaluation of the block cipher DES, and got a general tradeoff curve $N^2 = TM^2$ on block ciphers, where N is the size of key space, and T, M are the time/memory cost respectively. A common tradeoff point is taken $T = M = N^{\frac{2}{3}}$. In the rest of this paper we will refer to it as Hellman attack. It is noticed that in Hellman attack he viewed a mapping from the key space to the ciphertext space under a fixed plaintext P_0 as a random function $c = f_{P_0}(k)$. It led to a consequence that no matter how many plaintext-ciphertext pairs are captured in reality, only one pair of plaintext and ciphertext

will be utilized in Hellman attack. In order to utilize multiple data at the online stage, some significant works were done from two aspects: a) as for block ciphers, Biryukov, Mukhopadhyay and Sarkar extended Hellman attack to a time-memory-key tradeoff at the multiple key scenario, where a fixed plaintext is encrypted by several different keys and the goal is to recover one among those keys [8], which will be referred to as BMS attack; b) as for stream ciphers, Babbage [9] and Golic [10] independently proposed a TMTO, which will be referred to as BG attack. They viewed a mapping from an internal state to a truncation of output keystream as a random function and got a tradeoff curve $N = TM$, $T = D$ and $P = M$, where N, D and P denote the size of state space, the number of required data at the online stage and the time cost at the offline stage respectively. However, a problem is open till now:

Open problem 1 *How does one utilize several pairs of known plaintext and ciphertext during Hellman attack against block ciphers at the single key scenario?*

In 2000, Biryukov and Shamir [11] combined Hellman attack and GS attack together, and further presented a time-memory-data tradeoff (TMDTO, in short) for stream ciphers, which will be referred to as BS attack. The tradeoff curve of BS attack is $N^2 = TM^2D^2$, where $T \geq D^2$, and one suggested tradeoff point is $T = N^{\frac{2}{3}}$ and $M = D = N^{\frac{1}{3}}$. In order to remove the restriction between T and D, Biryukov, Shamir and Wagner [12] improved BS attack by introducing a BSW sampling, which is indeed a simple combination of BS attack and the guess and determine attack (GDA, in short) [13,14] and reduces the time cost by increasing the amount of data. There is a shortcoming in BS attack with BSW sampling, that is, large amounts of data are filtered at the online stage and only a few data with specific prefix are used. An interesting problem in BS attack with BSW sampling is shown as below:

Open problem 2 *How does one utilize all data instead of a few data with a specific prefix in BS attack with BSW sampling?*

In this work we try to give an answer to the above two open problems. Firstly, we generalize the IRF from one dimension to high dimension, and introduce an invertibility of multiple random functions (IMRF, in short). Secondly, as for an IMRF, we generalize the birthday theorem, and propose a general algorithm of solving it. Finally, we apply it to the cryptanalysis of symmetric ciphers and get two new tradeoff curves. Let τ be the number of random functions in an IMRF, N be the size of key for block ciphers or state space for stream ciphers, and D be the number of data captured by an attacker. Here we assume that the attacker can access τ independent computing units, each containing a core and some memory and dealing with all computations related to one random function. Denote by T, M and P the time/memory/precomputation costs consumed for single computing unit respectively. Then two new tradeoff curves are $N^2 = TM^2D^3$, $N = PD$ and $D = \tau$ for block ciphers, and $N^2 = \tau^3TM^2D^2$, $N = \tau PD$ and $D \geq \tau$ for stream ciphers. An intuitive comparison of our method to existing methods is shown in Table 1.

Table 1. A comparison of our method to existing methods

Method	Tradeoff curve	Data	Scenario	Ref.
Hellman attack	$N^2 = TM^2$, $P = N$	$D = 1$	block ciphers with single key	[7]
BMS attack	$N^2 = TM^2D^2$ $N = PD$	$T \geq D^2$	block ciphers with multiple keys	[8]
BG attack	$N = TM$, $P = M$	$D \geq 1$	stream ciphers	[9,10]
BS attack	$N^2 = TM^2D^2$ $N = PD$	$T \geq D^2$	stream ciphers	[11]
BS attack with BSW sampling	$N^2 = TM^2D^2$ $N = PD$	only a few data among D are used	stream ciphers	[11]
Our method	$N^2 = TM^2D^3$ $N = PD$	$D = \tau \geq 1$	block ciphers with single key	Sec. 3.1
	$N^2 = \tau^3TM^2D_k^2$ $N = \tau PD_k$	$T \geq \tau D_k^2$	block ciphers with multiple keys	Sec. 3.2
	$N^2 = \tau^3TM^2D^2$ $N = \tau PD$	$D \geq \tau$	stream ciphers	Sec. 3.3

More precisely, our method has the following advantages:

1. It is highly parallel. The number τ of independent computing units available to an attacker is also a key parameter in our method. Each computing unit can do TMDTO attack for a target function independently. The larger τ is, the smaller the time/memory/precomputation cost consumed at each computing unit will be.
2. As for block ciphers, our method is a generalization of Hellman attack. Compared to the latter, our method admits more than one datum at the single key scenario. When a lot of plaintext/ciphertext pairs under the same key are captured, an attacker can utilize them to reduce the cost consumed at single computing unit. It is very significant for him/her to break block ciphers in the real world, especially, when he/she has access to a large amount of computing resources in a network or distributed computing environment. At the multiple key scenario, our method is a generalization of BMS attack. Compared to the latter, our method is more flexible and practical, and can utilize several data for each key.
3. As for stream ciphers, our method is a generalization of BS attack with BSW sampling. Compared to the latter, our method overcomes the shortcoming of BS attack with BSW sampling that only a few data are used at the online stage, and can utilize all data to do TMDTO attack for a target cipher. Our method can be viewed as a nice combination of TMTO attack and GDA.

As applications, we give some tradeoff points on some classical symmetric ciphers, including DES, AES-128, A5/1 [17], Grain-v1 [18], Grain-128 [19], etc.

Here we assume that 4096 computing units can be available for the attacker in Table 2, and the number of computing units are not restricted in Table 3.

Table 2. Our method with a fixed $\tau = 2^{12}$

Algorithm	Time T	Memory M	Data D	Precomp. T
DES	$2^{25.3}$	$2^{25.3}$	2^{12}	2^{44}
AES-128	$2^{73.3}$	$2^{73.3}$	2^{12}	2^{116}
A5/1	$2^{20.8}$	$2^{20.8}$	$2^{20.8}$	$2^{31.2}$
Grain-v1	$2^{65.3}$	$2^{54.7}$	$2^{54.7}$	$2^{93.3}$
Grain-128	$2^{102.7}$	$2^{93.3}$	$2^{93.3}$	$2^{150.7}$

Table 3. Our method on DES and AES

Algorithm	Comp. Unit τ	Time T	Memory M	Data D	Precomp. T
DES	$2^{18.7}$	$2^{18.7}$	$2^{18.7}$	$2^{18.7}$	$2^{37.3}$
AES	$2^{42.7}$	$2^{42.7}$	$2^{42.7}$	$2^{42.7}$	$2^{85.3}$

The rest of this paper is organized as follows. In Section 2 we first describe an IMRF, then generalize the birthday theorem, and finally provide an algorithm of solving the IMRF based on the generalized birthday theorem. As its application, Some tradeoff curves and points to block ciphers and stream ciphers are given in Sections 3.1, 3.2 and 3.3 respectively.

2 Invertibility of multiple random functions

2.1 Description

In this section we describe an invertibility of multiple random functions (IMRF, in short), which can be viewed as a generalization of an IRF.

Definition 1. *Let n and τ be two positive integers, and $f_1, f_2, \cdots, f_\tau$ be τ independent random functions from $\{0,1\}^n$ to $\{0,1\}^n$. For any given D data:*

$$\begin{aligned} &y_{1,1}, y_{1,2}, \cdots, y_{1,d_1} \in \mathrm{Img}(f_1), \\ &y_{2,1}, y_{2,2}, \cdots, y_{2,d_2} \in \mathrm{Img}(f_2), \\ &\quad\vdots \\ &y_{\tau,1}, y_{\tau,2}, \cdots, y_{\tau,d_\tau} \in \mathrm{Img}(f_\tau), \end{aligned} \tag{1}$$

where $D = \sum_{i=1}^{i=\tau} d_i$, $d_i \geq 1$, and $\mathrm{Img}(f_i)$ denotes the set of all images of f_i, $1 \leq i \leq \tau$, we call how to find a preimage x of some $y_{i,j}$ under f_i, i.e., $y_{i,j} = f_i(x)$, to be an invertibility of multiple random functions, where $1 \leq i \leq \tau$, $1 \leq j \leq d_i$.

Here it should be reminded of the following problem:

Question 1. Let n and τ be two positive integers, and $f_1, f_2, \cdots, f_\tau$ be τ independent random functions from $\{0,1\}^n$ to $\{0,1\}^n$. For any given D data $y_{i,j} \in \mathrm{Img}(f_i)$, where $1 \le i \le \tau$, $1 \le j \le d_i$, $D = \sum_{i=1}^{i=\tau} d_i$, for each f_i, how to find a preimage x_i of some $y_{i,j}$ under f_i, i.e., $y_{i,j} = f_i(x_i)$.

Though Question 1 looks very similar to an IMRF, they are two entirely different problems. The former is required to find a preimage of some $y_{i,j}$ for each f_i, that is, total τ preimages, which can be viewed to invoke an IRF simply τ times, but the latter is required to find a preimage of some $y_{i,j}$ for only one of f_i's. Obviously, an IMRF looks easier than Question 1. Thus we believe that an IMRF might have a more efficient solution than Question 1 in theory. In the next two sections we will discuss how to give a more efficient algorithm for an IMRF.

2.2 Generalized birthday theorem

In order to solve the IMRF, we will introduce a new birthday theorem, which is viewed as a generalization of the birthday theorem, that is, Lemma 1.

Theorem 1. (Generalized Birthday Theorem) *Let S be a set of size N, and $A_1, \cdots, A_\tau, B_1, \cdots, B_\tau$ be 2τ independent random subsets of S such that $|A_1| = \cdots = |A_\tau| = n$ and $|B_1| = \cdots = |B_\tau| = m$, where τ, n and m are three positive integers. If $\tau nm \ge N$, then there exists an integer i such that $A_i \cap B_i$ is nonempty with high probability, where $1 \le i \le \tau$.*

Proof. When $n+m > N$, the conclusion is trivial. Below we always assume that $n + m \le N$.

Denote by p the probability of the event that $A_i \cap B_i = \varnothing$. Then we have

$$p = \Pr[A_i \cap B_i = \varnothing] = \frac{\binom{N}{n} \cdot \binom{N-n}{m}}{\binom{N}{n} \cdot \binom{N}{m}} = \frac{(N-n)!(N-m)!}{N!(N-n-m)!}. \tag{2}$$

Let q be the probability of the event that there exists an integer i ($1 \le i \le \tau$) such that $A_i \cap B_i \ne \varnothing$. Due to the independence of all A_i and B_i, we have $q = 1 - p^\tau$. It is expected that $q > \frac{1}{2}$, that is, $p^\tau < \frac{1}{2}$.

By (2),

$$p = \frac{(N-m)(N-m-1)\cdots(N-m-n+1)}{N(N-1)\cdots(N-n+1)} = \prod_{i=0}^{n-1}(1 - \frac{m}{N-i}).$$

As a consequence, we have

$$p^\tau \le (1 - \frac{m}{N})^{n\tau} \le (1 - \frac{m}{N})^{\frac{N}{m}} \tag{3}$$

due to $n\tau \geq \frac{N}{m}$. Note that $(1-x)^{\frac{1}{x}} < \frac{1}{e}$ for any $0 < x < 1$[1], we have

$$q = 1 - p^{\tau} > 1 - \frac{1}{e} > \frac{1}{2}.$$

So the conclusion follows. ■

2.3 An efficient algorithm for an IMRF

Take $S = \{0,1\}^n$ and $N = 2^n$. For a given IMRF, a simple method of solving it is to choose one function f_i arbitrarily as a target and invoke the IRF once for it. Obviously, it is not optimal since it does not make efficient use of all known data. In this section we will provide a more efficient algorithm of solving an

[1] Consider the function $F(x) = x + \ln(1-x)$ with $x \in [0,1)$. Note that $F'(x) = 1 - \frac{1}{1-x} < 0$ for each $x \in (0,1)$, consequently it is shown that $F(x)$ is a strictly monotone decreasing function in $[0,1)$. Thus $F(x) < F(0) = 0$ for any $x \in (0,1)$, which implies that $\frac{1}{x}\ln(1-x) < -1$, that is, $(1-x)^{\frac{1}{x}} < \frac{1}{e}$.

IMRF and give its complexity evaluation, which is mainly based on the above generalized birthday theorem.

Algorithm 1: Set up lookup tables at the one-time offline stage

Input: τ random functions $f_1, f_2, \cdots, f_\tau$;
Output: lookup tables $T_{i,u}$;

choose two suitable integers m and t such that $mt^2 = N$;
for *each function* f_i*, set up* $r = t/D$ *lookup tables* $T_{i,u}$ *(*$1 \le u \le r$*),* **do**
  choose r simple permutations $\sigma_{i,u}$ and let $g_{i,u} = \sigma_{i,u} \circ f_i$, $1 \le u \le r$;
  choose m startpoints $s_{i,u,v,0}$ $(1 \le v \le m)$ randomly for each $T_{i,u}$;
  for *each startpoint* $s_{i,u,v,0}$ **do**
    compute a chain $C_{i,u,v}$ of length t

$$s_{i,u,v,0} \xrightarrow{g_{i,u}} s_{i,u,v,1} \xrightarrow{g_{i,u}} \cdots \xrightarrow{g_{i,u}} s_{i,u,v,t} \tag{4}$$

    and store the pair $(s_{i,u,v,0}, s_{i,u,v,t})$ into $T_{i,u}$;
  end
end
return *lookup tables* $T_{i,u}$;

Algorithm 2: Find a preimage x of some $y_{i,j}$ under f_i at the online stage

Input: D data $y_{i,j} \in \mathrm{Img}(f_i)$, $1 \le i \le \tau$, $1 \le j \le d_i$, $D = \sum_{i=1}^{\tau} d_i$;
Output: a preimage x of some $y_{i,j}$ under f_i or failure message;

for $i = 1, 2, \cdots, \tau$ **do**
  for $j = 1, 2, \cdots, d_i$ **do**
    for $u = 1, 2, \cdots, r$ **do**
      compute $y_{i,j,1} = \sigma_{i,u}(y_{i,j})$ and look up $y_{i,j,1}$ in the second column of $T_{i,u}$;
      if $\exists v$ *s.t.* $y_{i,j,1} = s_{i,u,v,t}$ **then**
        compute $s_{i,u,v,t-1}$ from the startpoint $s_{i,u,v,0}$ along the chain $C_{i,u,v}$ under $g_{i,u}$;
        return $x = s_{i,u,v,t-1}$ *as a preimage of* $y_{i,j}$ *under* f_i*;*
      end
      compute $y_{i,j,k} = g_{i,u}(y_{i,j,k-1})$ and look up $y_{i,j,k}$ in the second column of $T_{i,u}$ for $2 \le k \le t$;
      if $\exists v$ *s.t.* $y_{i,j,k} = s_{i,u,v,t}$ **then**
        compute $s_{i,u,v,t-k}$ from the startpoint $s_{i,u,v,0}$ along the chain $C_{i,u,v}$ under $g_{i,u}$;
        return $x = s_{i,u,v,t-k}$ *as a preimage of* $y_{i,j}$ *under* f_i*;*
      end
    end
  end
end
return *failure message that no preimage* x *is found;*

Here we explain simply why it holds that $y_{i,j} = f_i(x)$ with high probability when Algorithm 2 returns x. If x is returned at Step 7 in Algorithm 2, then we have

$$\sigma_{i,u}(y_{i,j}) = y_{i,j,1} = s_{i,u,v,t} = \sigma_{i,u}(f_i(s_{i,u,v,t-1})) = \sigma_{i,u}(f_i(x)).$$

Note that $\sigma_{i,u}$ is a permutation, it follows that $y_{i,j} = f_i(x)$. If x is returned at Step 12 in Algorithm 2, we have

$$g_{i,u}(y_{i,j,k-1}) = y_{i,j,k} = s_{i,u,v,t} = g_{i,u}(s_{i,u,v,t-1}).$$

Since $g_{i,u} = \sigma_{i,u} \circ f_i$ is also a random function, thus $y_{i,j,k-1} = s_{i,u,v,t-1}$ holds with high probability. We can approximate that $y_{i,j,k-1}$ is just equal to $s_{i,u,v,t-1}$. And so on, we further get $y_{i,j,1} = s_{i,u,v,t-k+1}$. So

$$\sigma_{i,u}(y_{i,j}) = y_{i,j,1} = s_{i,u,v,t-k+1} = \sigma_{i,u}(f_i(s_{i,u,v,t-k})) = \sigma_{i,u}(f_i(x)),$$

which implies that $y_{i,j} = f_i(x)$.

As for Algorithms 1 and 2, it should be pointed out that:

1. We assume that an attacker has access to τ computing units, each having independent core and memory. Our assumption is indeed easy to meet in reality when τ is not too large. Since the attacker has τ independent computing units, thus he/she can set up the lookup tables $T_{i,u}$ simultaneously for one function f_i on one computing unit in Algorithm 1, and match one group $y_{i,1}, y_{i,2}, \cdots, y_{i,d_i}$ in the lookup tables $T_{i,u}$ simultaneously on one computing unit in Algorithm 2, too. It shows that Algorithms 1 and 2 can be done in high parallel at the level of functions.
2. At the offline stage, a technique of Hellman attack with distinguishing points [4] can be used to set up the lookup tables $T_{i,u}$ to avoid data collision. Since mt will be far smaller than N, there are fewer collisions and merges in each $T_{i,u}$ in a practical attack.

Below we discuss the cost of Algorithms 1 and 2. Denote by T, M and P the time/memory/precomputation cost at each computing unit respectively. For simplification, we assume that $d_1 = \cdots = d_\tau = d$. Then $D = \tau d$. At the offline stage, each computing unit needs to store r lookup tables $T_{i,u}$, each $T_{i,u}$ containing m pairs of startpoint and endpoint and covering about mt data. Thus we have $M = rm = mt/D$. Since each pair of startpoint and endpoint stored in $T_{i,u}$ is obtained by means of a chain $C_{i,u,v}$ of length t, thus the precomputation needs to invoke $g_{i,u}$ total rmt times, that is, $P = rmt = mt^2/D$. At the online stage, for each $y_{i,j}$, it needs t queries in a lookup table $T_{i,u}$, and t calls for $g_{i,u}$. Thus each computing unit needs at most rtd queries for a group of images $y_{i,1}, y_{i,2}, \cdots, y_{i,d}$ of f_i and total rtd calls for all $g_{i,u}$. In a practical attack, we can speed up one query in the lookup table $T_{i,u}$ by means of sorting or hash mapping when m is not too large. Here we approximate $T = rtd = t^2/\tau$, which implies T queries and T calls. In order to find a collision with high probability, by Theorem 1, it is expected that $\tau \times \frac{mt^2}{D} \times d = N$. So we get the following conclusion:

Theorem 2. *For a given IMRF defined as in Definition 1, let T, M and P the time/memory/precomputation cost at each computing unit in Algorithms 1 and 2 respectively. Then we have*

$$N^2 = \tau T M^2 D^2 \text{ and } PD = N, \tag{5}$$

where $T \geq D^2/\tau$ and $D \geq \tau$.

Proof. The conclusion follows directly from $M = mt/D$, $P = mt^2/D$, $T = t^2/\tau$, $mt^2 = N$ and $r = t/D \geq 1$.

3 Application

3.1 Block ciphers at the single key scenario

Block cipher is one of classical symmetric ciphers and has been widely used in information processing to protect the confidentiality of message. A typical block cipher contains three main parameters: key, plaintext and ciphertext. Plaintexts are encrypted to ciphertexts under the control of keys in a block cipher. Due to the recovery of ciphertexts, the block cipher must be a permutation on the plaintext space. Therefore the plaintext space and the cipher space are the same in block ciphers.

Let $\mathcal{K}$ and $\mathcal{C}$ be the key space and the plaintext space of a block cipher respectively. Here we consider the scenario of single key analysis of block ciphers. In Hellman attack, a fixed plaintext P_0 is chosen and the ciphertext C is viewed as a function $f_{P_0}(K)$ on the key K, where $C = f_{P_0}(K) = E_K(P_0)$, and E_K denotes the encryption function of the block cipher. Since it is required to recover the specific unknown key K, Hellman attack uses exactly one datum at the online stage though an attacker may capture many plaintext/ciphertext data easily. Below we will provide a new TMDTO method, which overcomes the disadvantage of Hellman attack and can use more than one plaintext/ciphertext datum got under the same key. To the best of our knowledge, this is the first multi-data TMTO attack against block ciphers at the single key scenario.

Suppose that τ computing units are available for us. We first choose τ fixed plaintexts $P_1, P_2, \cdots, P_\tau$ and τ functions f_i from the key space $\mathcal{K}$ to the ciphertext space $\mathcal{C}$, where $C_i = f_i(K) = E_K(P_i)$, $1 \leq i \leq \tau$. If the size of $\mathcal{K}$ is not equal to that of $\mathcal{C}$, for example, DES, a reduction function R is required. At this time we let $f_i(K) = R(E_K(P_i))$. It is noticed that each f_i in a block cipher is viewed as a random function, and they are mutually independent for different plaintexts. At the online stage, the attacker has known τ plaintext/ciphertext pairs (P_i, C_i) of a block cipher under an unknown key K, where $C_i = E_K(P_i)$, $1 \leq i \leq \tau$. So the attacker attempts to recover K by Algorithms 1 and 2 in Section 2.3. Note that there is only one datum used for each f_i, that is, $d = 1$, thus $D = \tau$. Then we get the following tradeoff curve:

$$N^2 = T M^2 D^3, D = \tau, PD = N, \tag{6}$$

where $T \geq D$. Let n be the bit length of the key, i.e., $N = 2^n$. Set $\tau = 2^l$. For a given τ such that $l \leq \frac{n}{3}$, a common tradeoff point is $T = M = 2^{\frac{1}{3}(2n-3l)}$ and $P = 2^{n-l}$. If τ is not fixed, then the curve $T = M = D = 2^{\frac{n}{3}}$ and $P = 2^{\frac{2n}{3}}$ is suggested.

It is shown in (6) that the number τ of available computing units is also a key parameter. Obviously, the larger τ is, the smaller the time/memory cost T and M at each computing unit will be. This is a very important property in the real world, which will help us to execute some practical attacks for some block ciphers in a distributed computing environment. Due to the restriction of practical computing resource, l is usually very small, for example, $l \leq 20$.

Here we provide a simple comparison with Hellman attack. Let T_{total}, M_{total} and P_{total} be the total time/memory/precomputation cost of all computing units, that is, $T_{\text{total}} = \tau T$, $M_{\text{total}} = \tau M$ and $P_{\text{total}} = \tau P$. Note that $D = \tau$, by Formula (6), we have

$$N^2 = T_{\text{total}} M_{\text{total}}^2 \text{ and } P_{\text{total}} = N,$$

which is the same as that of Hellman attack. Thus we have the following conclusion:

Theorem 3. *For a block cipher, the time/memory/precomputation costs T, M and P at each computing unit are reduced linearly when τ or D is increased.*

Finally, as examples, we apply the above method to the well-known block ciphers DES and AES with 128-bit key (AES-128, in short). The results for a fixed $\tau = 2^{12}$ are listed in Table 4, and for a varied τ in Table 5.

Table 4. Our method on DES and AES-128 for a fixed $\tau = 2^{12}$

	Time cost T	Memory cost M	Data D	Precomp. cost P
DES	$2^{25.3}$	$2^{25.3}$	2^{12}	2^{44}
AES-128	$2^{73.3}$	$2^{73.3}$	2^{12}	2^{116}

Table 5. Our method on DES and AES-128

	Comp. Unit τ	Time cost T	Memory cost M	Data D	Precomp. cost P
DES	$2^{18.7}$	$2^{18.7}$	$2^{18.7}$	$2^{18.7}$	$2^{37.3}$
AES-128	$2^{42.7}$	$2^{42.7}$	$2^{42.7}$	$2^{42.7}$	$2^{85.3}$

3.2 Block ciphers at the multiple key scenario

In [8] Biryukov, Mukhopadhyay and Sarkar described a multiple key scenario on block ciphers, where a fixed plaintext was encrypted repeatedly by several

different keys. At such a scenario they naturally extended Hellman attack to a time-memory-key tradeoff on block ciphers, and got the same tradeoff curve as that of BS attack on stream ciphers:

$$N^2 = TM^2D_k^2,$$

where D_k is the number of possible keys on the online stage. It is referred to as BMS attack. When we pay our eyes on an IMRF, it is found that our method is more suitable for the multiple key scenario on block ciphers, where the arbitrary number of plaintexts can be encrypted repeatedly by several different keys. Let τ be the number of known plaintexts, denoted by $P_1, P_2, \cdots, P_\tau$, and each plaintext P_i be encrypted repeatedly by D_k different keys $k_1, k_2, \cdots, k_{D_k}$. So we get total τD_k data $y_{i,j} = f_i(k_j) = E_{k_j}(P_i)$ at the online stage, where $1 \leq i \leq \tau$ and $1 \leq j \leq D_k$. We take each f_i as a random function and view them as an instance of the IMRF. By Theorem 1, we get a new tradeoff curve for block ciphers at the multiple key scenario:

$$N^2 = \tau^3 TM^2 D_k^2 \quad \text{and} \quad \tau P D_k = N, \tag{7}$$

where $T \geq \tau D_k^2$. A common tradeoff point is that $P = T = N^{\frac{3}{5}}$, $M = D_k = \tau = N^{\frac{1}{5}}$. A comparison of our method to BMS attack on AES-128 is shown as in Table 6.

Table 6. A comparison of our method to BMS attack on AES-128

	Plaintexts (τ)	Time (T)	Memory (M)	Key (D_k)	Precomp. (P)
BMS attack	1	2^{80}	2^{56}	2^{32}	2^{96}
Our method	2^{20}	2^{60}	2^{48}	2^{20}	2^{80}

3.3 Stream ciphers

Stream ciphers are another of classical symmetric ciphers and are mainly used in network communication. Stream ciphers have a very different behavior from block ciphers. A typical stream cipher usually contains a number of internal registers, a seed key and an initial vector (optional), and mainly consists of a state function and a filter function. In a stream cipher, the state function involves in the update of states of the internal registers, and the filter function derives a key sequence from states of the internal registers, which is used to encrypt plaintexts to ciphertexts.

TMTO can be applied to stream ciphers by several methods, for example, BG/BS attack, HS attack [15], DK attack [16], and so on. Compared with the TMTO for block ciphers, the TMTO for stream ciphers has the advantage that it can utilize many data at the online stage. Here we focus on BS attack with BSW sampling. In BS attack, a mapping $y = f(x)$ from the state x to the prefix

y of the output keystream is viewed as a random function on the state space of size N, and its goal is to recover some state x of the internal registers from a piece of output keystream bits. The BSW sampling is a technique of combining TMTO attack and GD attack together, and is used to improve BS attack. For a stream cipher, let n be the bit length of its states. Suppose that an attacker can determine the rest l-bit value x_0 of a state x from y_0 by guessing the $(n-l)$-bit value x' of x, where $y = y_0 \parallel y' = f(x) = f(x_0 \parallel x')$, x_0 and y_0 have l bits, and x' and y' have $(n-l)$ bits. Then a new function $y' = f_{y_0}(x')$ on a $(n-l)$-bit subspace of the state space can be derived from $y = f(x)$, which is dependent on the value of y_0 and shown in Fig. 1.

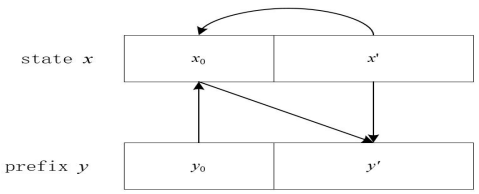

Fig. 1 Diagram of deriving f_{y_0} from f

In BS attack with BSW sampling, the prefix y_0 is chosen to a fixed value, denoted by a, and the function $y' = f_a(x')$ is viewed as a random function on the space of dimension $n-l$. BS attack with BSW sampling reduces the target space from n bits to $n-l$ bits, but leads to a disadvantage: though an attacker has captured D prefixes y, only the prefixes y whose first l-bit y_0 matches a are valid, that is to say, only $2^{-l}D$ data can be used at the online stage, and most data (about $(1-2^{-l})D$) are useless indeed. Below we give a new TMDTO method, which overcomes the disadvantage of BS attack with BSW sampling and utilizes all D prefixes y at the online stage. Our method can be viewed as a nice combination of TMTO attack and GD attack.

Similarly, here we still assume that at least τ computing units are available for an attacker. For a given stream cipher, we first analyze its security by means of GD attack. Suppose that the state x can be recovered from a prefix y by guessing the $(n-l)$-bit value of x. Then we analyze its security again by means of TMTO attack. We choose τ fixed distinct l-bit prefixes $a_1, a_2, \cdots, a_\tau$ and τ functions $f_{a_1}, f_{a_2}, \cdots, f_{a_\tau}$, and run Algorithms 1 and 2 as described in Section 2.3 . Let D be the number of prefixes y, which can be extracted from a $(n+D-1)$-bit successive keystream. For each f_a, on average, about $2^{-l}D$ prefixes y belong to the image of f_a, that is, $d = 2^{-l}D$. Note that τ must be no more than 2^l and the preimage space of f_a has size $2^{-l}N$ for any a, thus we have

$$N^2 = \tau^3 T M^2 D^2 \text{ and } \tau P D = N, \tag{8}$$

where $\tau \le 2^l \le D$ and $T \ge 2^{-2l}\tau D^2$.

For a given stream cipher, let n and k be the bit lengths of the state and seed key respectively. Since $\log_2 \tau$ is usually very small due to the limit of computing resource, it is easy to meet the condition $l \ge \log_2 \tau$ for a maximal l got in GD

attack. Thus we usually take $l = \log_2 \tau$. In this case all D data are used at the online stage without any waste. Below we take $l = \log_2 \tau$ such that $l \leq \frac{n}{4}$ and give several common tradeoff points:

- $T = 2^{\frac{1}{3}(2n-5l)}$, $M = D = 2^{\frac{1}{3}(n-l)}$ and $P = 2^{\frac{2}{3}(n-l)}$;
- $T = M = 2^{k-l}$, $D = 2^{\frac{1}{2}k}$ and $P = 2^{\frac{3}{2}k-l}$ if $n = 2k$.

It should be pointed out that it is also meaningful when $l > \log_2 \tau$. Though not all data are used in this case, the time/memory/precomputation cost at each computing unit can also be reduced. Let L be the maximal value of l got by GD attack for a stream cipher. If $L > \log_2 \tau$, then the following tradeoff point is taken:

- $T = M = D = (\tau^{-3}N^2)^{\frac{1}{5}}$, $P = (\tau^{-2}N^3)^{\frac{1}{5}}$ and $l = \lceil \frac{1}{5}\log_2(\tau N) \rceil$ when $L \geq \frac{1}{5}\log_2(\tau N)$;
- $T = (\tau^{-1}2^{-4l}N^2)^{\frac{1}{3}}$, $M = D = (2^l\tau^{-2}N)^{\frac{1}{3}}$, $P = (2^{-l}\tau^{-1}N^2)^{\frac{1}{3}}$ and $l = L$ when $\log_2 \tau < L < \frac{1}{2}\log_2(\tau^{-2}N)$.

Below we provide a simple comparison with BS attack with BSW sampling. Let T_{total}, M_{total} and P_{total} be the total time/memory/precomputation cost of all computing units, that is, $T_{\text{total}} = \tau T$, $M_{\text{total}} = \tau M$ and $P_{\text{total}} = \tau P$. By Formula (8), we have

$$N^2 = T_{\text{total}} M_{\text{total}}^2 D^2 \text{ and } P_{\text{total}} D = N,$$

which is the same as that of BS attack. Thus we have the following conclusion:

Theorem 4. *For a stream cipher, the time/memory/precomputation costs T, M and P at each computing unit are reduced linearly when τ is increased.*

Finally, as examples, our method is applied to the stream ciphers A5/1 [17], Grain-v1 [18] and Grain-128 [19] respectively. Note that l is taken at most 16, 28 and 48 in A5/1, Grain-v1 and Grain-128 respectively [20], the results of our analysis for a fixed $\tau = 2^{12}$ are listed in Table 7.

Table 7. Our method on A5/1, Grain-v1 and Grain-128 for a fixed $\tau = 2^{12}$

	Units τ	GD param. l	Time T	Mem. M	Data D	Prep. P
A5/1	2^{12}	12	$2^{20.8}$	$2^{20.8}$	$2^{20.8}$	$2^{31.2}$
Grain-v1	2^{12}	28	$2^{65.3}$	$2^{54.7}$	$2^{54.7}$	$2^{93.3}$
Grain-128	2^{12}	48	$2^{102.7}$	$2^{93.3}$	$2^{93.3}$	$2^{150.7}$

References

1. Goldreich O.: Foundations of cryptography volume I basic tools. Cambridge University Press (2008)

2. Pohlig S.C. and Hellman M.E.: An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Transaction on Information Theory, vol. 24, pp.106-110 (1978)
3. Feller W.: An introduction to probability theory and its applications. 3rd ed. New York: Wiley (1968)
4. Denning D.E.: Attributed to rivest in cryptography and data security. Addison-Wesley, page 100 (1982)
5. Avoine G., Junod P. and Oechslin P.: Characterization and improvement of time-memory trade-off based on perfect tables. ACM Transaction Information System Security, 11(4), 17:117:22 (2008).
6. Oechslin P.: Making a faster cryptanalytic time-memory trade-off. Crypto 2003, LNCS 2729, pp.617-630 (2003)
7. Hellman M.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory 26(4), pp.401-406 (1980)
8. Biryukov A., Mukhopadhyay S. and Sarkar P.: Improved time-memory trade-offs with multiple data. SAC 2005, LNCS 3897, pp.110-127 (2006)
9. Babbage S.: Improved exhaustive search attacks on stream ciphers. European Convention on Security and Detection 1995. IEE Conference Publication, pp.161-166 (1995)
10. Golic J.D.: Cryptanalysis of alleged A5 stream cipher. EUROCRYPT 1997, LNCS 1233, pp.239-255 (1997)
11. Biryukov A. and Shamir A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. ASIACRYPT 2000, LNCS 1976, pp.1-13 (2000)
12. Biryukov A., Shamir A. and Wagner D.: Real time cryptanalysis of A5/1 on a PC. FSE 2000, LNCS 1978, pp.1-18 (2001)
13. Hawkes P. and Rose G.: Guess and determine attacks on SNOW. SAC 2002, LNCS 2595, pp.37-46 (2003)
14. Feng X., Liu J., Zhou Z., Wu C. and Feng D.: A byte-based guess and determine attack on SOSEMANUK. ASIACRYPT 2010, LNCS 6477, pp.146-157 (2010)
15. Hong J. and Sarkar P.: New applications of time memory data tradeoffs. ASIACRYPT 2005, LNCS 3788, pp.353-372 (2005)
16. Dunkelman O. and Keller N.: Treatment of the initial value in time-memory-data trade-off attacks on stream ciphers. Information Processing Letters, 107(5), pp.133-137 (2008)
17. Anderson R.: A5. Newsgroup Communication (1994)
18. Hell M., Johansson T., Maximov A. and Meier W.: The Grain family of stream ciphers. New stream cipher designs, LNCS 4986, pp.179-190 (2008)
19. Hell M., Johansson T., Maximov A. and Meier W.: A stream cipher proposal: Grain-128. IEEE International Symposium on Information Theory, ISIT 2006 (2006)
20. Wei Y., Pasalic E., Zhang F. and Wu W.: Key recovery attacks on Grain family using BSW sampling and certain weaknesses of the filtering function. http://eprint.iacr.org/2014/971.pdf (2014)