Improved Slender-set Linear Cryptanalysis

Guo-Qiang Liu[1], Chen-Hui Jin[1], and Chuan-Da Qi[2]

[1] Information Science and Technology Institute, Zhengzhou, 450000, Henan, China
[2] Xinyang Normal University, Xinyang, 464000, Henan, China
`liuguoqiang87@hotmail.com`
`jinchenhui@126.com`
`qichuanda@sina.com`

Abstract. In 2013, Borghoff *et al.* introduced a slender-set linear cryptanalysis on PRESENT-like ciphers with key-dependent secret S-boxes. In this paper, we propose an improved slender-set linear attack to PRESENT-like ciphers with secret S-boxes. We investigate three new cryptanalytic techniques, and use them to recover the secret S-boxes efficiently. Our first new idea is that we propose a new technique to support consistency of partitions of the input to the secret S-boxes. Our second new technique is that we present a more efficient method to recover the coordinate functions of secret S-boxes based on more information than that of Borghoff's attack. The third new technique is that we propose a method of constructing all correct coordinate function of secret S-boxes by pruning search algorithm. In particular, we implemented a successful linear attack on the full round Maya in practice. In our experiments, the correct S-box can be recovered with 2^{36} known plaintexts, $2^{18.9}$ time complexity and negligible memory complexity at a success rate of 87.5% based on 200 independent trials. Our attack is the improvement and sequel of Borghoff's work on PRESENT-like cipher with secret S-boxes.

Keywords: block cipher, linear cryptanalysis, PRESENT-like, secret S-box.

1 Introduction

Block ciphers are a class of cryptographic algorithms which are widely used. After the establishment of Advanced Encryption Standard (AES), there has been a need for cryptographic solutions which can provide low cost security for small device, such as RFID tags or sensor networks. This opens up the study field of lightweight block cipher.

In recent years, many lightweight block ciphers have been developed, such as mCrypton [14], HIGHT [11], SEA [20], DESXL [13], KATAN [8], MIBS [12]. PRESENT [4] is the most remarkable representative lightweight block cipher. It is proposed by Bogdanov *et al.* in CHES 2007. PRESENT cipher is an iterated 31 rounds SPN block cipher with a 64-bit block size. PRESENT has two variants of key, one with an 80-bit and the other with a 128-bit. Each round of PRESENT cipher has three layers. The first layer is a substitution layer, which consists of 16 parallel applications of the same 4-bit S-box. The second layer is a permutation layer, which consists of a bit-wise permutation of 64-bit. The last layer is a key addition layer, where a round key is exclusive-ored to the text. Due to the small 4-bit S-box, bit-wise permutation and Xor addition, PRESENT has a fast and compact hardware implementation. In recent years, the cryptanalysis on PRESENT has been actively performed so far. The best known cryptanalysis attack on PRESENT is a linear attack on 26 of the 31 rounds presented by Cho [9] in CT-RSA 2010. This attack could break the 26 rounds PRESENT with 2^{64} data complexity, 2^{32} memory accesses complexity and 2^{72} time complexity. Although this attack is hardly to perform in practice, the number of rounds used should not be dramatically reduced.

In order to strengthen the PRESENT cipher and reduce the number of rounds, Gomathisankaran and Lee [15] presented a PRESENT-like cipher with secret S-boxes which is named Maya. The Maya cipher is a 16 rounds SPN cipher similar to PRESENT. The difference between PRESENT and Maya is that the substitution layer of Maya consists of 16 different S-boxes which are key-dependent and unknown.

Linear cryptanalysis [16, 17] is a classical tool for analyzing block ciphers. It was introduced by Matsui in 1993. Since for a random permutation, the probability of any linear relation between the plaintext and corresponding ciphertext bits should be balanced at 1/2. But this is not always the same in the case of block ciphers. The attacker can construct a distinguisher for a key-recovery attack based on this information leakage.

Nowadays, there are only a few papers investigating the security of PRESENT-like cipher with secret S-boxes. In 2011, Borghoff *et al.* [6] proposed a slender-set differential cryptanalysis of PRESENT-like cipher with randomly chosen secret S-boxes. In 2013, Borghoff *et al.* [7] outline how to attack this type of cipher with a linear attack by using slender-set. However, the slender-set linear cryptanalysis proposed in [7] was simply described and has something to be improved. In this paper, we present an improved slender-set linear cryptanalysis on PRESENT-like cipher.

Our Results In this paper, we investigate an improved slender-set linear cryptanalysis of PRESENT-like cipher with secret S-boxes. Our contributions are threefold. First, we present a new technique to support consistency of partitions of the input to the secret S-box of the first S-box layer. This modification makes it possible to transform the original vectors into binary vectors more reasonably. Our second new idea is that we propose a new method to get the coordinate functions of secret S-boxes efficiently based on more information. This method considers all the information that comes from the candidate coordinate functions together instead of the three longest candidate coordinate functions. The third new technique is that we present a method of constructing the correct coordinate functions by pruning search algorithm, which can help us to recover the secret S-boxes more efficiently. Finally, we focus on the settings of PRESENT-like cipher where the secret S-boxes are key-dependent and are repeated for the first and last rounds. We propose an effective method of determining the correct S-box from the equivalent S-boxes with low time complexity. In particular, we implemented an improved slender-set linear attack to the full round Maya successfully. In our experiments, the correct S-box can be recovered with 2^{36} data complexity, $2^{18.9}$ time complexity and negligible memory complexity at a success rate of 87.5%. The correct S-box is usually found in less than a few minutes on a standard PC. Our attack is the improvement and sequel of Borghoff's work on PRESENT-like cipher with secret S-boxes.

Related Work There are lots of design and analysis to ciphers with secret S-boxes. In 1994, Gilbert and Chauvaud [10] presented a differential attack on the cipher Khufu which used secret S-boxes. Biham and Biryukov [1] described methods of strengthening DES, without slowing encryption speed, in which key-dependency is added by simply XORing key material before and after the S-boxes calculation. They studied the strengthened version of DES against exhaustive search, differential, and linear attacks that can be used existing hardware. In 1996, Vaudenay [23] provided a cryptanalysis of reduced-round variants of Blowfish, in which the S-boxes were randomly generated from the private key. In 1998, Schneier and Kelsey *et al.* [19] studied a new encryption algorithm with key-dependent S-boxes. In 2001, Biryukov and Shamir [2, 3] investigated the security of iterated ciphers, in which the substitutions and permutations are all secret and key-dependent. In 2009, Borghoff et al. [5] researched on the cipher C2, which has a secret S-box. In 2011, Szaban and Seredynski [22] proposed cryptanalysis of a methodology to design dynamic cellular automata (CA)-based S-boxes. Stoianov [21] presented and analyzed an approach to change the S-boxes used in the algorithm AES. In 2013, Peng and Jin [18] proposed a cryptanalysis of a new method to design key-dependent S-boxes by using hyperchaotic Chen system.

Organization The paper is organized as follows. Section 2 introduces the structure of PRESENT-like cipher. Section 3 outlines the slender-set linear attack on PRESENT-like cipher described in [7]. Section 4 presents our improved slender-set linear cryptanalysis on PRESENT-like cipher with secret S-boxes. Section 5 gives experimental results of our attack on Maya cipher. Finally, Section 6 concludes the paper.

2 The PRESENT-like Cipher

The PRESENT-like cipher is an n-bit SPN block cipher. The round function consists of round key, S-boxes and permutations. Assuming that the number of rounds is N.

(1) Round key K: n-bit round key is Xored to the text.

(2) S-box S: n/m key dependent and different m-bit secret S-boxes.

(3) P-box P: The bit-wise permutation between the S-box layers is public or secret.

The PRESENT-like cipher can be described in Algorithm 1.

Algorithm 1 N-rounds PRESENT-like cipher

```
Require:   $n$-bit plaintext $X$; main key $K$
Ensure:   $n$-bit ciphertext $C = E_K(X)$
 1: Derive $n/m$ $m$-bit S-boxes $S_i$ and round keys $K_i$ $(1 \le i \le n/m)$ from $K$
 2: $STATE = X$
 3: for $i = 1$ to $N$ do
 4:   Parse $STATE$ as $STATE_1||STATE_2||\cdots||STATE_{n/m}$
 5:   for $j = 1$ to $n/m$ do
 6:     $STATE_j = S_j(STATE_j)$
 7:     Apply bit permutation to $STATE$
 8:     Add round key $K_i$ to $STATE$
 9:   end for
10: end for
11: return
```

Maya is a typical example of the PRESENT-like cipher described in Algorithm 1 with $n = 64$ and $m = 4$

3 Borghoff's Slender-set Linear Attack

In this section, we review the Borghoff's slender-set linear attack of recovering the secret S-boxes described in [7]. We call the attack in [7] as *Borghoff's linear attack* for short. First, we introduce some notations and definitions used in this paper.

3.1 Linear Cryptanalysis

We follow the notations used in [7]. The canonical inner product on F_2^n is denoted by $\langle \cdot, \cdot \rangle$, that is

$$\langle (a_0, a_1, \cdots, a_{n-1}), (b_0, b_1, \cdots, b_{n-1}) \rangle = \sum_{i=0}^{n-1} a_i b_i$$

Given a function $H : F_2^n \to F_2^m$, for an m -bit output mask β and an n-bit input mask α , the bias ε of the linear approximation $\langle \beta, H(x) \rangle + \langle \alpha, x \rangle$ is defined by

$$p(\langle \beta, H(x) \rangle + \langle \alpha, x \rangle = 0) = \frac{1}{2} + \varepsilon$$

where the probability p is taken over all choices of inputs x. The Walsh or Fourier-transform of H at the pair $(\alpha, \beta) \in F_2^n \times F_2^m$ is defined by

$$\hat{H}(\alpha, \beta) = \sum_{x \in F_2^n} (-1)^{\langle \beta, H(x) \rangle + \langle \alpha, x \rangle}$$

The relation between the Fourier transform of H and the bias of a linear approximation is given by

$$\varepsilon = \frac{\hat{H}(\alpha,\beta)}{2^{n+1}}$$

Thus to study the bias $\langle\beta, H(x)\rangle + \langle\alpha, x\rangle$ is equivalent to study the Fourier-transform of H at the pair (α, β).

3.2 Brief Description of Borghoff's Linear Attack

In this section, we introduce the basic principle of Borghoff's linear Attack. In [7], Borghoff *et al.* started with a differential-style attack on PRESENT-like cipher. They supposed that the characteristics with single-bit differences at the output of the S-box layer in the first round have a stronger correlation with single-bit differences at the input to the S-box layer in the last round. They named this differential attack as the slender-set differential cryptanalysis. Using a similar hypothesis for linear characteristics, they could get useful information about the secret S-boxes with the single-bit mask at the output of the S-box layer in the first round and the low-weight mask at the input of the S-box layer in the last round. Thus we can name this linear attack as slender-set linear cryptanalysis. In the following, we outline Borghoff's linear attack.

Without loss of generality, Borghoff's linear attack focused on the leftmost S-box, which is denoted simply by S . All other S-boxes can be processed similarly. We denote that

$$F : F_2^4 \times F_2^{60} \rightarrow F_2^{64} \quad and \quad F(x, y) = c$$

where the function F is the encryption function that starts after the first layer of S-boxes, x is the output of the leftmost S-box of the first S-box layer, and y is concatenation of the outputs of the remaining S-boxes of the first S-box layer. Thus the Fourier-transform of F at the pair $((\alpha_1, \alpha_2), \beta) \in F_2^{64} \times F_2^{64}$ is denoted by

$$\hat{F}(\alpha,\beta) = \hat{F}((\alpha_1,\alpha_2),\beta) = \sum_{x\in F_2^4}\sum_{y\in F_2^{60}}(-1)^{\langle\beta,F(x,y)\rangle+\langle\alpha_1,x\rangle+\langle\alpha_2,y\rangle}$$

In [12], Borghoff *et al.* used of the bias of the value $\langle\beta, F(x, y)\rangle$ for a fixed values of x . They denote the corresponding function by

$$T_x : F_2^{60} \rightarrow F_2^{64} \quad and \quad T_x(y) = F(x, y)$$

and we look at

$$\hat{T}_x(0,\beta) = \sum_{y\in F_2^{60}}(-1)^{\langle\beta,T_x(y)\rangle} = \sum_{y\in F_2^{60}}(-1)^{\langle\beta,F(x,y)\rangle}$$

We list two useful lemmas in [7] as follows.

Lemma 1 *[7] With the notation from above, it holds that*

$$2^4 \hat{T}_\lambda(0,\beta) = \sum_{\alpha_1 \in F_2^4} (-1)^{\langle \alpha_1, \lambda \rangle} \hat{F}((\alpha_1, 0), \beta)$$

Denote the whole encryption function by E. They split its input as before and consider

$$E : F_2^4 \times F_2^{60} \rightarrow F_2^{64} \quad \text{and} \quad E(x,y) = c$$

where x is now the input to the first S-box, and y is the input to all other S-boxes. They define the function corresponding to fixing x as T'_x , that is

$$T'_x : F_2^{60} \rightarrow F_2^{64} \quad \text{and} \quad T'_x(y) = E(x,y)$$

For a selection of masks β and each possible value of x , they estimate the value of $T'_x(0,\beta)$. This is simply done by encrypting a number of known plaintexts with the first four bits fixed to x and counting how often the value of

$$\langle \beta, T'_x(y) \rangle = \langle \beta, E(x,y) \rangle$$

is zero (resp., one). The next lemma is the key for deducing information about the secret S-box.

Lemma 2 *[7] With the notation from above, the bias of $\langle \beta, T'_x(y) \rangle$ is equal to the bias of $\langle \beta, T_{S(x)}(y) \rangle$. That is*

$$\hat{T}'_x(0,\beta) = \hat{T}_{S(x)}(0,\beta)$$

In [7], they assume that, for a given mask β , there is exactly one mask α such that $\hat{F}((\alpha,0),\beta)$ is high while for any $\xi \neq \alpha$ the value $\hat{F}((\xi,0),\beta)$ is close to zero. According to Lemma 1 and Lemma 2, we have

$$\begin{aligned} \hat{T}'_x(0,\beta) = \hat{T}_{S(x)}(0,\beta) &= 2^{-4} \sum_{\xi \in F_2^4} (-1)^{\langle \xi, S(x) \rangle} \hat{F}((\xi,0),\beta) \\ &\approx 2^{-4} (-1)^{\langle \alpha, S(x) \rangle} \hat{F}((\alpha,0),\beta) \end{aligned} \tag{1}$$

In this way, we can partition the values of x into two equally-sized sets V_0 and V_1 depending on the sign of $\hat{T}'_x(0,\beta)$, where $V_\gamma = \{x | \langle \alpha, S(x) \rangle = \gamma\}, \gamma = 0,1.$ If we get all four linearly independent coordinate functions of secret S-box, such as $(\langle 2^i, S(0) \rangle, \langle 2^i, S(1) \rangle, \cdots, \langle 2^i, S(15) \rangle), 0 \leq i \leq 3$, we can recover the secret S-box. Borghoff's linear attack is summarized as the following steps:

Step1. Let the output mask $\beta = 0^{4j} || b || 0^{60-4j}$, $0 \leq j \leq 15$. For every leftmost input $0 \leq x \leq 15$ and for every $1 \leq b \leq 15$, we estimate the value of the counter $\hat{T}'_x(0,\beta)$ by Eq. (1) after encrypting enough plaintexts.

Step2. After 15 vectors $W_\beta = (\hat{T}'_0(0,\beta), \hat{T}'_1(0,\beta), \cdots, \hat{T}'_{15}(0,\beta))$ being retrieved, we identify the three longest vectors using the Euclidean norm as a metric, as Borghoff *et al.* assume that these vectors contain the most reliable information. We transform each of these vectors into a binary vector such that the eight highest counter values correspond to '1'-bits and the remaining correspond to '0'-bits.

Step3. We take a majority vote among these three binary vectors to find the correct coordinate functions of secret S-box.

Step4. We recover the 4-bit secret S-boxes based on four linearly independent coordinate functions of secret S-boxes.

Through the above steps, we may partition the value of x into two sets $V_\gamma = \{x|\langle\alpha, S(x)\rangle = \gamma\}, \gamma = 0, 1$. However, the value of α and γ are unknown. Borghoff *et al.* pointed out that they might reveal one or more of the sets $\{x|\langle 2^i, S(x)\rangle = 0\}, 0 \leq i \leq 3$ by repeating the steps described above for other value of $\beta = 0^{4j}||b||0^{60-4j}$ with different $0 \leq j \leq 15$.

However, the linear cryptanalysis proposed in [7] was in less details and has something to be improved.

First, according to Eq. (1), we can only partition the value of x into two sets $V_\gamma = \{x|\langle\alpha, S(x)\rangle = \gamma\}, \gamma = 0, 1$ by Borghoff's linear attack. However, for a fixed α , the value of γ is unknown, which would cause problems: for a fixed α, we only know which values of x belonging to the same set, but we still do not know the values of x belonging to which set. In other words, if one coordinate of the binary vector is equal to "0", it dose not mean the corresponding value of coordinate function is essentially equal to "0". Furthermore, the sign of $\hat{F}((\alpha, 0), \beta)$ might be opposite for the same α and the different β. According to Eq. (1), the opposite sign of $\hat{F}((\alpha, 0), \beta)$ will cause opposite sign of $\hat{T}'_x(0, \beta)$. This makes it possible to partition the x into different set with the same α and the different β. We did the same work as Borghoff *et al.* did. As an example, we carried out this attack using 2^{25} known plaintexts on the 10 rounds PRESENT-like cipher. The three longest vectors were these:

$(-3138, -2218, -3156, 3146, -2486, 1784, -2974, -3452, 1392, 1602, 2850, 3198, -3100, 2796, -3458, 1708)$

$(-2558, -1768, -2022, 2798, -1754, 2538, -1808, -2440, 2784, 2694, 2424, 3378, -2576, 2378, -2658, 2424)$

$(3046, 1842, 1730, -2982, 1952, -1600, 2116, 2930, -2426, -2742, -2036, -2440, 2918, -1764, 3112, -1670)$

After transforming these vectors into binary vectors as described, one gets

$$(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1)$$

$$(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1)$$

$$(1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0)$$

We can see that the partition of x in the third binary vector is opposite comparing with that in the first and second binary vectors. This problem will bring some noise when we recover the correct coordinate functions of secret S-box with majority vote. To overcome this, we develop a new idea to support consistency of the partition of x.

Second, due to that Borghoff *et al.* just used the information from the three longest vectors, which will lose the information from the candidate binary vectors. Furthermore, the majority vote for getting "true" vector in [7] is not reasonable and to be perfected. In order to improve the information collection phase, we present a new method of making full use of information from the 240 output low-weight masks $\beta = 0^{4j}||b||0^{60-4j}, 0 \leq j \leq 15, 1 \leq b \leq 15$ of the S-box layer in the last round instead of three longest vectors. And we introduce an improved voting method for constructing the correct candidate coordinate functions.

Third, for PRESENT-like cipher, the experiment shows that we can only get one correct set $\{x|\langle 2^i, S(x)\rangle = 0\}, 0 \leq i \leq 3$ in most case by using Borghoff's linear attack. As an example, we carried out Borghoff's attack by using 2^{22} to 2^{27} known plaintexts on the 10 rounds PRESENT-like cipher. However, we can only get one or two correct coordinate functions of secret S-box(see Table 3). This makes it possible to recover 1 or 2 out of 4 bit information of secret S-box in most case, but not the 4-bit information about secret S-box. The question is: How to get more correct coordinate functions of secret S-box as many as possible with lower data complexity? To overcome this problem, we proposed method of constructing all correct coordinate functions of secret S-box by using pruning search algorithm.

4 Improved Slender-set Linear Cryptanalysis

In this section, we explain the approach of our improved slender-set linear cryptanalysis of recovering the secret S-box. Comparing with the Borghoff's linear attack, there are three main improvements in this paper. First, we present a new technique to avoid the complementary vectors in the candidate coordinate functions of secret S-boxes. Second, we propose an efficiently method of getting full use of all the information coming from the active S-box synthetically. In other words, we consider all the $15 \times 16 = 240$ vectors together instead of the three longest vectors. Third, we introduce an effective filter to construct correct coordinate functions of secret S-boxes by using pruning search algorithm. The pruning search algorithm can help us to search the correct coordinate functions as many as possible with lower complexity. Finally, we focus on the settings where the S-boxes are key-dependent and are repeated for the first and last rounds. We investigate an efficient method of recovering the secret S-box uniquely with a low time complexity.

Notation. Throughout this section, assuming that α is a binary vectors, the Hamming weight of α is denoted by $wt(\alpha)$. The complementary vector of α is denoted by $\overline{\alpha}$. The vector $W_\beta = (\hat{T}'_0(0,\beta), \hat{T}'_1(0,\beta), \cdots, \hat{T}'_{15}(0,\beta))$ described in Section 3.2 is called as the *original vector.*

4.1 Recovering the Correct Coordinate Functions

We start with the first new technique of our improved slender-set linear cryptanalysis. In Borghoff's linear attack, the value of corresponding coordinate in different binary vectors may be different. But the partition of x is valid in one binary vector. In this case, for a fixed k, we consider to partition x by the distance between $\hat{T}'_i(0,\beta)$ and $\hat{T}'_k(0,\beta)$, where $0 \le i \le 15$. Without loss of generality, we let $k = 0$ in our paper. We propose the notion of *relative distance* to support consistency of the partition of x as follows.

Specifically, for every $0 \le j \le 15, 1 \le b \le 15$ we consider the output masks $\beta = 0^{4j}||b||0^{60-4j}$, and we get $15 \times 16 = 240$ *original vectors* $W_\beta = (\hat{T}'_0(0,\beta), \hat{T}'_1(0,\beta), \cdots, \hat{T}'_{15}(0,\beta))$ after encrypting enough plaintexts. For every β and every $0 \le i \le 15$, we compute the *relative distance*

$$\mathbb{R}^{(i)}_\beta = \hat{T}'_0(0,\beta) - \hat{T}'_i(0,\beta)$$

between $\hat{T}'_0(0,\beta)$ and $\hat{T}'_i(0,\beta)$. We transform each of these *relative distance* vectors $(\mathbb{R}^{(0)}_\beta, \mathbb{R}^{(1)}_\beta, \cdots, \mathbb{R}^{(15)}_\beta)$ into binary vectors $(\mathbb{B}^{(0)}_\beta, \mathbb{B}^{(1)}_\beta, \cdots, \mathbb{B}^{(15)}_\beta)$ such that the eight highest values correspond to "1"-bits and the remaining values correspond to "0"-bits. If $\mathbb{B}^{(0)}_\beta$ is equal to "1", then we transform the vector $(\mathbb{B}^{(0)}_\beta, \mathbb{B}^{(1)}_\beta, \cdots, \mathbb{B}^{(15)}_\beta)$ into the complementary vector $B_i = (\overline{\mathbb{B}^{(0)}_\beta}, \overline{\mathbb{B}^{(1)}_\beta}, \cdots, \overline{\mathbb{B}^{(15)}_\beta})$. If $\mathbb{B}^{(0)}_\beta$ is equal to "0", then we let $B_i = (\mathbb{B}^{(0)}_\beta, \mathbb{B}^{(1)}_\beta, \cdots, \mathbb{B}^{(15)}_\beta)$.

As an example, we obtained the vectors described in Section 3.2. The *relative distance* vector are these for the three longest vectors:

$(0, 920, -18, 6284, 652, 4922, 164, -314, 4530, 4740, 5988, 6336, 38, 5934, -320, 4846)$

$(0, 790, 536, 5356, 804, 5096, 750, 118, 5342, 5252, 4982, 5936, -18, 4936, -100, 4982)$

$(0, -1204, -1316, -6028, -1094, -4646, -930, -116, -5472, -5788, -5082, -5486, -128, -4810, 66, -4716)$

We transform the *relative distance* vector into binary vectors as follows:

$$(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1)$$

$$(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1)$$

$$(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1)$$

The benefit of this technique is that the value of the first coordinate of binary vector B_i is identically equal to "0". By using this method, we can support consistency of the partition of input x based on the *original vector* $W_\beta = (\hat{T}'_0(0,\beta), \hat{T}'_1(0,\beta) \cdots, \hat{T}'_{15}(0,\beta))$.

After we get the 240 binary vectors $B_i, 1 \leq i \leq 240$, we introduce the second technique of our improved attack as follows.

We define the distances between two binary vectors B_i and B_j by

$$\mathbb{D}_{B_i,B_j} = wt(\overline{B_i \oplus B_j})$$

where $1 \leq i, j \leq 240$ and $wt(\overline{B_i \oplus B_j})$ is the *Hamming weight* of $\overline{B_i \oplus B_j}$. Due to the first coordinate of the binary vectors being identically equal to "0", it holds that $\mathbb{D}_{B_i,B_j} \neq 0$. Thus we can see that $\mathbb{D}_{B_i,B_j} = 2, 4, 6, 8, 10, 12, 14, 16$ and the number of $\mathbb{D}_{B_i,B_j}$ is equal to $C_{240}^2 = 28680$. For two random vectors α and β, we present the probability distribution of $wt(\overline{\alpha \oplus \beta})$ as following lemma.

Lemma 3. *With the notation above, let α, β be random vectors and $wt(\alpha) = wt(\beta) = 8$ with $\alpha = (0, a_1, a_2, \cdots, a_{15})$, $\beta = (0, b_1, b_2, \cdots, b_{15})$, $a_i, b_i \in \{0, 1\}, 1 \leq i \leq 15$. Then the probability of $wt(\overline{\alpha \oplus \beta}) = k$ is equal to*

$$p(wt(\overline{\alpha \oplus \beta}) = k) = \frac{C_8^{(16-k)/2} C_7^{(16-k)/2}}{C_{15}^7}$$

where $k = 2, 4, 6, 8, 10, 12, 14, 16$.

We now compute the probability distribution of $wt(\overline{\alpha \oplus \beta})$ for two vectors α, β based on the notation in Lemma 3 (see Table 1).

Table 1. The probability distribution of $wt(\overline{\alpha \oplus \beta})$ for randomly chosen vectors α, β

k	Probability	k	Probability	k	Probability	k	Probability
2	0.1243%	6	18.2751%	10	30.4584%	14	0.8702%
4	3.0458%	8	38.0730%	12	9.1375%	16	0.0155%

Due to the method of *relative distance*, there have no complementary vectors in the binary vectors. According to Table 1, the expectation of $wt(\overline{\alpha \oplus \beta})$ is equal to 8.533 for two random vectors. For two binary vectors, they will be similar to each other when $wt(\overline{B_i \oplus B_j})$ approximate to 16. Thus we believe the vectors B_i and B_j would be closer to each other when $wt(\overline{B_i \oplus B_j}) > 8$.

Now we explain our approach for getting the information about secret S-box from all 240 binary vectors B_β. We assume that the binary vectors corresponding the same α are similar to each other. We start with the method of partition B_β by using the notion of *similarity degree* as following definition.

Definition 1. Given 240 binary vectors $B_i, 1 \leq i \leq 240$. Let $\xi, \tau > 0$ and $\mathbb{D}_{B_i,B_j} = wt(\overline{B_i \oplus B_j})$ be the distances between binary vectors B_i and B_j , where $1 \leq j \leq 240$. The *similarity degree* of B_i and B_j is defined by

$$\mathbb{S}_{B_i,B_j} = g(B_i, B_j) + \sum_{\substack{1 \leq k \leq 240 \\ k \neq i, k \neq j}} (f(B_i, B_k) + f(B_j, B_k))$$

where the function $g(B_i, B_j) = \begin{cases} \xi, & if\ wt(\overline{B_i \oplus B_j}) \geq t; \\ 0, & others. \end{cases}$ and $f(B_i, B_j) = \begin{cases} \tau, & if\ wt(\overline{B_i \oplus B_j}) \geq t; \\ 0, & others. \end{cases}$

According to Definition 1, the *similarity degree* of B_i and B_j considers not only the relationship between B_i and B_j but also the relationship from other B_k, which can help us to collect all the correlation between 240 candidate binary vectors synthetically with suitable value of t, ξ, τ. Fig 1 shows the form of *similarity degree*:

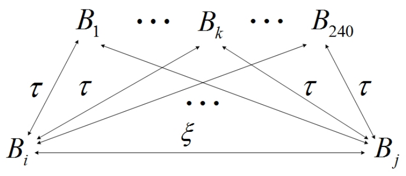

Fig. 1. The form of similarity degree

For 240 candidate binary vectors, we can get $C_{240}^2 = 28680$ *similarity degrees* between the binary vectors B_i and B_j. The higher value of *similarity degree*, the stronger correlation between two binary vectors.

In order to recover four coordinate functions of secret S-box, we start with approach of how to partition the 240 binary vectors into four parts as follows.

Given 240 binary vectors $B_i, 1 \leq i \leq 240$. As can be seen in Table 1, the probability of $\mathbb{D}_{B_i,B_j} = 16$ is equal to 0.0155%. Such a small probability means that: If $\mathbb{D}_{B_i,B_j} = 16$, there must be a very strong correlation between two binary vectors B_i and B_j . In other words, the binary vectors B_i and B_j should be very close to the same coordinate function of secret S-box in case of $\mathbb{D}_{B_i,B_j} = 16$. Therefore, we treat the binary vectors which hold $\mathbb{D}_{B_i,B_j} = 16$ as the priority vectors. Assuming that we construct r priority sets $\Omega_k, 1 \leq k \leq r$, for every $k_i, k_j \in \Omega_k$, it must hold that $\mathbb{D}_{B_{k_i},B_{k_j}} = 16$. According to the definition of priority sets, the higher value of $|\Omega_k|$, the more information about one coordinate function of secret S-box. Therefore, we start with the priority set which the value of $|\Omega_k|$ is maximal. Let $k_i \in \Omega_k$. We choose a vector B_{k_i} from the 240 binary vectors and mark B_{k_i}. Then we sort 240 binary vectors in descending order up to the value of $\mathbb{S}_{B_{k_i},B_j}$ from $j = 1$ to $j = 240$. We check if the vector B_j is marked for every $1 \leq j \leq 240$. If B_j is unmarked, We add B_j into the first set Φ_1 until $|\Phi_1| = 60$. We stop repeating this method when we have identified four parts $\Phi_1, \Phi_2, \Phi_3, \Phi_4$.

Next, we propose the improved voting method of constructing the candidate coordinate functions of secret S-boxes based on these four sets $\Phi_1, \Phi_2, \Phi_3, \Phi_4$. In [7], Borghoff *et al.* proposed a majority voting method to get "true" vector from three longest vectors. It means that the weighting factor of each vector is considered equal. Compared to Borghoff's attack, our attack treats the similarity degree as the weighting factor in voting method as follows.

Let $\beta = (\beta_0, \beta_1, \cdots, \beta_{15})$. For a given set Φ_l, $1 \leq l \leq 4$, for each $0 \leq x \leq 15$, we compute

$$v_{l,x} = \sum_{\alpha \in \Phi_l} \left(\sum_{\beta \in \Phi_l, \beta_x = 0} \mathbb{S}_{\alpha,\beta} - \sum_{\beta \in \Phi_l, \beta_x = 1} \mathbb{S}_{\alpha,\beta} \right)$$

and transform the vectors $(v_{l,0}, v_{l,1}, \cdots, v_{l,15})$ into a binary vector which is denoted by V_l, such that the eight highest value correspond to "0"-bits and the remaining correspond to "1"-bits. Our method of finding four candidate coordinate functions of secret S-boxes is described in Algorithm 2.

Algorithm 2 Finding four candidate coordinate functions of secret S-boxes

```
Require:
    The 240 binary vectors $B_i, 1 \leq i \leq 240$;
    The value of $t, \xi, \tau$;
Ensure:    Four candidate coordinate functions $V_1, V_2, V_3, V_4$
 1: According to the value of $t, \xi, \tau$ , for every $1 \leq i, j \leq 240$, we compute the similarity
    degrees $\mathbb{S}_{B_i,B_j}$ .
 2: Construct $r$ priority sets $\Omega_k, 1 \leq k \leq r$.
 3: Choose binary vector $w \in \Omega_k$, where the value of $|\Omega_k|$ is maximal. We mark $w$ and
    sort 240 binary vectors in descending order up to the value of $\mathbb{S}_{w,B_j}$, $1 \leq j \leq 240$.
 4: $l = 1$
 5: while $l \leq 4$ do
 6:    $n = 1$.
 7:    for $j = 1$ to 240 do
 8:       if $B_j$ is unmarked then
 9:          Add $B_j$ into the set $\Phi_l$ and $n \leftarrow n + 1$.
10:       else if $n \leq 60$ then
11:          Continue.
12:       else
13:          Break.
14:       end if
15:    end for
16:    For each $0 \leq x \leq 15$ , we compute

       $$v_{l,x} = \sum_{\alpha \in \Phi_l} \left( \sum_{\beta \in \Phi_l, \beta_x = 0} \mathbb{S}_{\alpha,\beta} - \sum_{\beta \in \Phi_l, \beta_x = 1} \mathbb{S}_{\alpha,\beta} \right)$$

       where $\beta = (\beta_0, \beta_1, \cdots, \beta_{15})$. Then we transform the vectors $(v_{l,0}, v_{l,1}, \cdots, v_{l,15})$
       into a binary vector $V_l$ such that the eight highest value correspond to "0"-bits
       and the remaining correspond to "1"-bits.
17:    $l \leftarrow l + 1$
18: end while
19: return  Four candidate coordinate functions $V_1, V_2, V_3, V_4$.
```

However, in our experiment, the candidate vectors found by Algorithm 2 might not be complete the correct coordinate functions of secret S-box. In order to get all information about secret S-box, we propose a method of constructing all correct coordinate functions of secret S-box by using pruning search algorithm. The efficient filtering method in pruning search algorithm is presented as the following proposition.

Proposition 1. *Let $r \leq m$, S-box $S(x) = (s_0(x), s_1(x), \cdots, s_{m-1}(x)) : \{0,1\}^m \rightarrow \{0,1\}^m$ be a bijective mapping, and $t_0, t_1, \cdots, t_{m-1}$ be a permutation of $0, 1, \cdots, m-1$. Then it holds that the function $h(x) = (s_{t_0}(x), s_{t_1}(x), \cdots, s_{t_{r-1}}(x)) : \{0,1\}^m \rightarrow \{0,1\}^r$ is balanced.*

In [7], the authors pointed out that the intersections of two correct sets (with different masks) contain exactly four values. our Proposition 1 extended the conclusion of their checking correctness.

For every correct vectors $V_i = (v_{i,0}, v_{i,1}, \cdots, v_{i,15}), 1 \leq i \leq r \leq 4$, it holds that

$$h(x) = \begin{pmatrix} 0 & 1 & \cdots & 15 \\ v_{1,0}||\cdots||v_{r,0} & v_{1,1}||\cdots||v_{r,1} & \cdots & v_{1,15}||\cdots||v_{r,15} \end{pmatrix}$$

is balanced. We call this filter the *balance filter*. In order to measure the effectiveness of *balance filter*, we present the probability of random vectors passing this filter.

Without loss of generality, let $a_i, b_i, c_i, d_i \in \{0,1\}, 1 \leq i \leq 15$,$A = (0, a_1, a_2, \cdots, a_{15})$, $B = (0, b_1, b_2, \cdots, b_{15})$,$C = (0, c_1, c_2, \cdots, c_{15})$,$D = (0, d_1, d_2, \cdots, d_{15})$ and $wt(A) = wt(B) = wt(C) = wt(D) = 8$. Then the probability of A, B, C, D passing the *balance filter* is equal to

$$\frac{15!}{(C_{15}^{7})^4} \approx 2^{-10.36}$$

Such a small probability means the *balance filter* is a strong filter, and many wrong candidate coordinate functions may be found by this filter. The pruning search algorithm of constructing correct coordinate functions of secret S-boxes by using *balance filter* is described in Algorithm 3.

In conclusion, our attack for recovering the correct coordinate functions of secret S-boxes can be described as following steps:

Step1. We get 240 *original vectors* $W_\beta = (\hat{T}'_0(0,\beta), \hat{T}'_1(0,\beta), \cdots, \hat{T}'_{15}(0,\beta))$ after encrypting enough plaintexts. For every β, we compute the *relative distance* $\mathbb{R}^i_\beta$ between $\hat{T}'_0(0,\beta)$ and $\hat{T}'_i(0,\beta)$.

Step2. Since we have obtained 240 *relative distance* vectors $(\mathbb{R}^{(0)}_\beta, \mathbb{R}^{(1)}_\beta, \cdots, \mathbb{R}^{(15)}_\beta)$ in step 1, we transform these vectors into binary vectors B_β.

Step3. In this step we find four candidate coordinate functions by using Algorithm 2 based on 240 binary vectors B_β .

Step4. We search all four correct candidate coordinate functions of secret S-boxes by using Algorithm 3.

After the steps above, though we get four correct sets $V_{\gamma_i} = \{x | \langle \alpha_i, S(x) \rangle = \gamma_i\}, \gamma_i = 0, 1$, $0 \leq i \leq 3$, we still do not know the value of α_i, γ_i. Therefore, we

Algorithm 3 Constructing the sets of four candidate correct coordinate functions

```
Require:
    Four candidate coordinate functions $V_1, V_2, V_3, V_4$ searched by Algorithm 2;
    The sets $\Theta_t^{(k)}$, $t = 1,2,3,4$, $k = 2,4,6,8,10,12,14,16$. The vector $V_{t,r}^{(k)} \in \Theta_t^{(k)}$, is
    the one of the possible vectors with changing $k$ bits compared with the vector $V_t$
    and with $wt(V_{t,r}^{(k)}) = 8$, where $1 \le r \le |\Theta_t^{(k)}|$;
    $j = 0$;
Ensure: The set $\Psi$ of four correct candidate coordinate functions
    The Function $F(\Theta, j, V)$
 1: $j \leftarrow j + 1$
 2: for $k$ is even, $k = 2$ to $16$ do
 3:     for $i = 1$ to $|\Theta_t^{(k)}|$ do
 4:         $U_j = V_{j,i}^{(k)}$.
 5:         if $2 \le j < 4$ then
 6:             Check if the vectors $U_1, \cdots, U_j$ pass the balance filter.
 7:             if pass the balance filter then
 8:                 do $F(\Theta, j, V)$.
 9:             else
10:                 Return.
11:             end if
12:         end if
13:         if $j = 4$ then
14:             Add $\{U_1, \cdots, U_j\}$ into the set $\Psi$.
15:         end if
16:     end for
17: end for
18: return The set $\Psi$ of four correct candidate coordinate functions
```

just recover the equivalent S-boxes of the secret S-boxes. In [7], Borghoff *et al.* pointed out that the single-bit mask at the output of the S-box layer in the first round has a stronger correlation with the low-weight mask at the input of the S-box layer in the last round. It was reasonable to assume that α_i was of weight one. According to this assumption, in order to determine the correct secret S-box, we should consider $2^4 \times 4! \approx 2^{8.6}$ possible equivalent S-boxes for each 4-bit S-box in the cipher Maya. Therefore, we need $(2^{8.6})^{16} \approx 2^{137.6}$ time complexity to recover all 16 secret S-boxes by exhaustive search. It is so large that the complexity is infeasible. In this paper, we focus on the settings of PRESENT-like cipher where the S-boxes are key-dependent and are repeated for the first and last rounds. We present a method of determining all 16 secret S-boxes with lower time complexity.

4.2 Determining the Secret S-box

In this section, we propose a technique to determine the secret S-boxes from the equivalent ones. Let $S_1, S_2, \cdots, S_N$ be secret S-boxes and $S_1^{-1}, S_2^{-1}, \cdots, S_N^{-1}$ be the inverse of these S-boxes. Without the loss of generality, we explain how to determine the leftmost S-box S_1 . Assuming that we get m correct coordinate functions of secret S-box $U_j = (u_{j,0}, u_{j,1}, \cdots, u_{j,2^m-1}), V_j = (v_{j,0}, v_{j,1}, \cdots, v_{j,2^m-1}), 1 \le$

$j \leq m$, for m-bit secret S-boxes S_1 and S_1^{-1} , we denote $S_1(x) = P_1(f_1(x) \oplus k_1)$ and $S_1^{-1}(x) = Q_1^{-1}(g_1^{-1}(x) \oplus t_1)$, where $0 \leq k_1, t_1 \leq 2^m - 1$, P_1, Q_1 are m-bit permutations and f_1, g_1 are known functions based on U_j and V_j:

$$f_1(x) = \begin{pmatrix} 0 & 1 & \cdots & 2^m - 1 \\ u_{1,0}||\cdots||u_{m,0} & u_{1,1}||\cdots||u_{m,1} & \cdots & u_{1,2^m-1}||\cdots||u_{m,2^m-1} \end{pmatrix}$$

$$g_1(x) = \begin{pmatrix} 0 & 1 & \cdots & 2^m - 1 \\ v_{1,0}||\cdots||v_{m,0} & v_{1,1}||\cdots||v_{m,1} & \cdots & v_{1,2^m-1}||\cdots||v_{m,2^m-1} \end{pmatrix}$$

With the notations above, if we determine the value of k_1, P_1 or t_1, Q_1, we can recover the secret S-box S_1 (or S_1^{-1}) uniquely. We propose an efficient filtering method of determining the value of t_1, P_1, Q_1 as follows.

For every $0 \leq x \leq 2^m - 1$, it must hold that $P_1(f_1(x) \oplus k_1) = g_1(Q_1(x) \oplus t_1)$. Then we have

$$P_1 f_1(x) \oplus P_1 k_1 = g_1(Q_1(x) \oplus t_1) \quad \textit{and} \quad P_1 f_1(0) \oplus P_1 k_1 = g_1(Q_1(0) \oplus t_1)$$

then we have

$$P_1 f_1(0) \oplus P_1 f_1(x) \oplus g_1(Q_1(0) \oplus t_1) \oplus g_1(Q_1(x) \oplus t_1) = 0$$

and then

$$P_1(f_1(0) \oplus f_1(x)) = g_1(Q_1(0) \oplus t_1) \oplus g_1(Q_1(x) \oplus t_1)$$

where f_1, g_1 are known functions, t_1 is an unknown m-bit constant and P_1, Q_1 are unknown m-bit permutation. In order to determine t_1, P_1, Q_1 , we guess t_1, Q_1 by exhaustive search first. If the value of t_1 and Q_1 are correct, it must hold that P_1 is an m-bit permutation. By using this guess and determine method, the time complexity of determining one secret S-box can be reduced to $2^m \times m!$ from $(2^m)^2 \times (m!)^2$. To the cipher Maya, the complexity to determine one correct S-box is equal to $2^4 \times 4! \approx 2^{8.58}$.

In order to check the correctness of t_1 and Q_1, we introduce two effective filters as following definition.

Definition 2. Let $0 \leq e_1, e_2 \leq 2^m - 1$, $e_1 \neq e_2$. Assuming that $P_1(f_1(0) \oplus f_1(x)) = g_1(Q_1(0) \oplus t_1) \oplus g_1(Q_1(x) \oplus t_1)$, where f_1, g_1 are known functions, t_1 is an unknown m-bit constant and Q_1, P_1 are unknown m-bit permutation. If the value of t_1, Q_1 are correct, it holds that

$$\begin{aligned} & g_1(Q_1(f_1^{-1}(e_1 \oplus e_2 \oplus f_1(0))) \oplus t_1) \oplus g_1(Q_1(0) \oplus t_1) \\ & = g_1(Q_1(f_1^{-1}(e_1 \oplus f_1(0))) \oplus t_1) \oplus g_1(Q_1(f_1^{-1}(e_2 \oplus f_1(0))) \oplus t_1) \end{aligned}$$

we name this filter *linear filter*. Let $0 \leq e_i \leq 2^m - 1$ and $wt(e_i) = 1, 1 \leq i \leq r$, it holds that

$$wt\left(g_1(Q_1(f_1^{-1}(e_i \oplus f_1(0))) \oplus t_1) \oplus g_1(Q_1(0) \oplus t_1)\right) = 1$$

we name this filter *weight filter*.

The linear and weight filter can generalize to more than two elements e_1, e_2 . In general, given r linearly independent $e_1, e_2, \cdots, e_r$, where $r \leq m$. For *linear filter*, assuming that $a_i \in \{0,1\}$ and at least two of $a_1, a_2, \cdots, a_r$ are nonzero, then the linear combination $a_1e_1 \oplus a_2e_2 \oplus \cdots \oplus a_re_r$ can obtain one equation for checking the correctness of t_1, Q_1. There are $2^r - r - 1$ linear combinations based on $e_1, e_2, \cdots, e_r$, thus we have $2^r - r - 1$ equations for checking the correctness of t_1, Q_1. For *weight filter*, each $e_i, wt(e_i) = 1, 1 \leq i \leq r$ obtains one equation for checking the correctness of t_1, Q_1. Thus we have r equations for checking the correctness of t_1, Q_1. In fact, the *linear filter* is equivalent to P being linear, and *weight filter* is equivalent to $wt(P(e_i)) = 1$ with $wt(e_i) = 1$. In order to measure the effectiveness of *linear filter* and *weight filter*, we start with the necessary and sufficient condition of t_1, Q_1 passing these two filters as following theorem.

Theorem 1. *Let P be an m-bit bijective mapping. The necessary and sufficient condition of P being a m-bit permutation is that P is linear and for every e_i with $wt(e_i) = 1$, we have $wt(P(e_i)) = 1$.*

According to theorem 1, the case of P being a m-bit permutation is equivalent to that of t_1, Q_1 passing the *linear filter* and *weight filter*. For an m-bit bijective mapping P , the probability of P being a bit-wise permutation is equal to $4!/16! \approx 2^{-39.67}$. Thus the probability of t_1, Q_1 passing the *linear filter* and *weight filter* is equal to $2^{-39.67}$. Moreover, the number of candidate t_1, Q_1 is equal to $2^4 \times 4! \approx 2^{8.58}$. It means that we can determine the correct t_1, Q_1 uniquely. Since we have known the correct t_1, Q_1 , we can calculate P_1 uniquely. By using this method, we can determine the correct S-boxes one by one. The time complexity for recovering all 16 S-boxes can be reduced to $16 \times 2^4 \times 4! \approx 2^{12.58}$ from $2^{137.6}$.

5 Experiments

In this section, we apply our attack on PRESENT-like cipher in practice and estimate the success rate and complexity. Our experiment is based on 200 independent trials. In our experiment, let $\tau = 1, \xi = 2, t = 10$ (see Definition 1 and Algorithm 2).

In the sequel, we measure the data complexity in units which are equivalent to a known plaintext. The dominative time consumption of our attack is the time cost in the pruning search algorithm (Algorithm 3). Therefore, we measure the time complexity of our attack in units which are equivalent to an operation of checking if the vectors passing the *balance filter*.

In [7], Borghoff *et al.* pointed out that they carried out the attack using 2^{25} data complexity to 10 rounds Maya. However, the number of correct coordinate functions obtained and success rate were not presented in [7]. We did the same experiment as Borghoff *et al.* did on the 10 rounds Maya with 2^{22} to 2^{27} data complexity. Table 2 shows the success rate of finding at least one correct coordinate function of the secret S-box by using Algorithm 2 in this paper and using the method in [7]. Table 3 shows more details about the number of correct coordinate functions and the success rate by Borghoff's method.

Table 2. The success rate of finding at least one correct coordinate function of the secret S-box on 10 rounds Maya with different data complexity by using Algorithm 2 and Borghoff's method

Data complexity	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}
Algorithm 2	100.0%	100.0%	99.5%	94.0%	55.0%	18.5%
Borghoff's method	91.5%	88.0%	71.0%	40.5%	23.0%	3.5%

Table 3. The number of correct coordinate functions and success rate with different data complexity on 10 rounds Maya in [7]

Number of correct coordinate functions	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}
1	37.0%	55.0%	63.0%	36.0%	22.0%	3.5%
2	49.5%	29.0%	6.5%	4.0%	1.0%	0.0%
3	5.0%	4.0%	1.5%	0.5%	0.0%	0.0%
4	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

According to Table 2, the success rate is higher in our improved slender-set linear attack. It means that we can get more information about the secret S-boxes than Borghoff's attack. As can be seen from Table 3, more than two correct coordinate functions would be recovered by Borghoff's linear attack in very rare cases with 2^{22} data complexity and even though with 2^{27} data complexity. In order to recover a 4-bit secret S-box, it requires four linearly independent coordinate functions. It means that Borghoff's attack can not get enough information to recover the secret S-box.

However, after getting one or more correct coordinate functions, being different from Borghoff's attack, our attack can construct all four correct coordinate functions of secret S-box by using Algorithm 3. Assuming that we get one correct coordinate function of the secret S-box, the maximal time complexity of Algorithm 3 is equal to $(C_{15}^{7})^3 \approx 2^{37.95}$. Table 4 shows the time consumption of Algorithm 3 based on 1 to 3 correct coordinate functions. Our experiment is based on 200 independent trials.

From Table 4, we can find all four correct coordinate functions of secret S-box in less than a few minutes on a standard PC (at AMD Athlon 7750 Processor 2.7 GHz) after getting one or more correct coordinate functions. To 10 rounds Maya, we can recover all four correct coordinate functions of secret S-box with 2^{24} data and $2^{17.78}$ time complexity at success rate 94% (see Table 2 and Table

Table 4. The average time complexity of finding all four correct coordinate functions of secret S-box by using Algorithm 3 on 10 rounds Maya

Number of known correct coordinate functions	Time complexity	Cost on stander PC
1	$2^{17.78}$	$\approx$ 7 Min. 12 Sec.
2	$2^{14.74}$	$\approx$ 53 Sec.
3	$2^{9.35}$	$\leq$ 1 Sec.

4). We apply our attack to 6 to 16 rounds Maya and estimate the data and time complexity and success rate (see Table 5).

Table 5. The data and time complexity and success rate of recovering all four correct coordinate functions on 6 to 16 rounds Maya cipher in this paper

Rounds	6	7	8	9	10	11	12	13	14	15	16
Data complexity	$2^{16.6}$	$2^{18.1}$	$2^{20.0}$	$2^{22.1}$	$2^{24.0}$	$2^{26.3}$	$2^{27.9}$	$2^{29.5}$	2^{31}	$2^{34.2}$	2^{36}
Time complexity	$2^{18.4}$	$2^{18.9}$	$2^{17.7}$	$2^{18.7}$	$2^{17.8}$	$2^{18.7}$	$2^{15.2}$	$2^{16.7}$	$2^{18.9}$	$2^{18.7}$	$2^{18.9}$
Success rate	90.5%	87.5%	88.0%	91.5%	94.0%	89.5%	90.0%	93.0%	91.5%	88.5%	87.5%

From Table 5, we can see that our attack can break 16 rounds Maya with 2^{36} known plaintexts and $2^{18.9}$ time complexity at success rate 87.5%. Our experiments suggest that 30-rounds Maya cipher can be break with approximately 2^{64} known plaintexts by using our attack.

6 Conclusion

In this paper, we present an improved slender-set linear attack to PRESENT-like cipher with secret S-boxes. We present three improvements comparing with Borghoff's attack. First, we use a new technique to to support consistency of partitions of the input x to the secret S-boxes of the first S-box layer. Second, a new technique is proposed by making full use of the information from all the 240 original vectors together instead of the three longest vectors. Third, an effective filter for constructing correct coordinate functions of secret S-boxes by using pruning search algorithm is presented. These three techniques can help us to get the all four correct coordinate functions more efficiently. Finally, we focus on the settings where the secret S-boxes are key-dependent and are repeated for the first and last rounds. We propose a filter to determine the correct S-box from equivalent S-boxes with lower time complexity. We describe the practical attack to full round Maya, as detailed in Table 5. The experiments show that the correct S-box can be recovered with 2^{36} known plaintexts, $2^{18.9}$ time complexity and negligible memory complexity at a success rate of 87.5% based on 200 independent trials. Our attack is the improvement and sequel of Borghoff's work, which is the best linear attack on PRESENT-like cipher with secret S-boxes up to now.

An interesting open question is to find a more efficient method to recover the correct coordinate functions of secret S-boxes. In fact, the approximation by Eq. (1) in section 3.2 could bring too much noise. Furthermore, the theoretical model for complexity of recovering coordinate functions would be a possible direction of future work.

Acknowledgments

I wish to thank the anonymous reviewers for very useful comments that help to improve the paper. This work was supported by National Natural Science Foundation of China (Grant No. 61272488, No. 61272465).

References

1. Biham, E., Biryukov, A.: How to strengthen DES using existing hardware. In: Pieprzyk, J., Safavi-Naini, R. (eds.) Advances in Cryptology - ASIACRYPT'94, vol. 917. Lecture Notes in Computer Science, pp. 398-412. Springer Berlin Heidelberg, (1995)
2. Biryukov, A., Shamir, A.: Structural Cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) Advances in Cryptology-EUROCRYPT 2001, vol. 2045. Lecture Notes in Computer Science, pp. 395-405. Springer Berlin Heidelberg, (2001)
3. Biryukov, A., Shamir, A.: Structural Cryptanalysis of SASAS. J. Cryptology 23(4), 505-518 (2010). doi:10.1007/s00145-010-9062-1
4. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, vol. 4727. Lecture Notes in Computer Science, pp. 450-466. Springer Berlin Heidelberg, (2007)
5. Borghoff, J., Knudsen, L., Leander, G., Matusiewicz, K.: Cryptanalysis of C2. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009, vol. 5677. Lecture Notes in Computer Science, pp. 250-266. Springer Berlin Heidelberg, (2009)
6. Borghoff, J., Knudsen, L., Leander, G., Thomsen, S.: Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes. In: Joux, A. (ed.) Fast Software Encryption, vol. 6733. Lecture Notes in Computer Science, pp. 270-289. Springer Berlin Heidelberg, (2011)
7. Borghoff, J., Knudsen, L., Leander, G., Thomsen, S.: Slender-Set Differential Cryptanalysis. J. Cryptology 26(1), 11-38 (2013). doi:10.1007/s00145-011-9111-4
8. Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2009, vol. 5747. Lecture Notes in Computer Science, pp. 272-288. Springer Berlin Heidelberg, (2009)
9. Cho, J.: Linear Cryptanalysis of Reduced-Round PRESENT. In: Pieprzyk, J. (ed.) Topics in Cryptology - CT-RSA 2010, vol. 5985. Lecture Notes in Computer Science, pp. 302-317. Springer Berlin Heidelberg, (2010)

10. Gilbert, H., Chauvaud, P.: A Chosen Plaintext Attack of the 16-round Khufu Cryptosystem. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, vol. 839. Lecture Notes in Computer Science, pp. 359-368. Springer Berlin Heidelberg, (1994)
11. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.-S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, vol. 4249. Lecture Notes in Computer Science, pp. 46-59. Springer Berlin Heidelberg, (2006)
12. Izadi, M., Sadeghiyan, B., Sadeghian, S., Khanooki, H.: MIBS: A New Lightweight Block Cipher. In: Garay, J., Miyaji, A., Otsuka, A. (eds.) Cryptology and Network Security, vol. 5888. Lecture Notes in Computer Science, pp. 334-348. Springer Berlin Heidelberg, (2009)
13. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) Fast Software Encryption, vol. 4593. Lecture Notes in Computer Science, pp. 196-210. Springer Berlin Heidelberg, (2007)
14. Lim, C., Korkishko, T.: mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In: Song, J.-S., Kwon, T., Yung, M. (eds.) Information Security Applications, vol. 3786. Lecture Notes in Computer Science, pp. 243-258. Springer Berlin Heidelberg, (2006)
15. Mahadevan, G., Ruby, B. L.: Maya: A Novel Block Encryption Function. International Workshop on Coding and Cryptography, `http://palms.princeton.edu/system/files/maya.pdf`
16. Matsui, M.: The First Experimental Cryptanalysis of the Data Encryption Standard. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, vol. 839. Lecture Notes in Computer Science, pp. 1-11. Springer Berlin Heidelberg, (1994)
17. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, vol. 765. Lecture Notes in Computer Science, pp. 386-397. Springer Berlin Heidelberg, (1994)
18. Peng, J., Jin, S.: Designing Key-Dependent S-Boxes Using Hyperchaotic Chen System. In: Zhong, Z. (ed.) Proceedings of the International Conference on Information Engineering and Applications (IEA) 2012, vol. 216. Lecture Notes in Electrical Engineering, pp. 733-740. Springer London, (2013)
19. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: a 128-bit Block Cipher. In: AES proposal, 1998.
20. Standaert, F.-X., Piret, G., Gershenfeld, N., Quisquater, J.-J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) Smart Card Research and Advanced Applications, vol. 3928. Lecture Notes in Computer Science, pp. 222-236. Springer Berlin Heidelberg, (2006)
21. Stoianov, N.: One Approach of Using Key-Dependent S-BOXes in AES. In: Dziech, A., Czyżewski, A. (eds.) Multimedia Communications, Services and Security, vol. 149. Communications in Computer and Information Science, pp. 317-323. Springer Berlin Heidelberg, (2011)
22. Szaban, M., Seredynski, F.: Dynamic Cellular Automata-Based S-Boxes. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) Computer Aided Systems Theory - EUROCAST 2011, vol. 6927. Lecture Notes in Computer Science, pp. 184-191. Springer Berlin Heidelberg, (2012)
23. Vaudenay, S.: On the weak keys of blowfish. In: Gollmann, D. (ed.) Fast Software Encryption, vol. 1039. Lecture Notes in Computer Science, pp. 27-32. Springer Berlin Heidelberg, (1996)