Improving the Algorithm 2 in Multidimensional Linear Cryptanalysis

Phuong Ha Nguyen, Hongjun Wu, and Huaxiong Wang

Division of Mathematical Sciences,
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
ng0007ha@e.ntu.edu.sg, {wuhj,hxwang}@ntu.edu.sg

Abstract. In FSE'09 Hermelin *et al.* introduced the Algorithm 2 of multidimensional linear cryptanalysis. If this algorithm is m-dimensional and reveals l bits of the last round key with N plaintext-ciphertext pairs, then its time complexity is $\mathcal{O}(mN2^l)$. In this paper, we show that by applying the *Fast Fourier Transform* and *Fast Walsh Hadamard Transform* to the Algorithm 2 of multidimensional linear cryptanalysis, we can reduce the time complexity of the attack to $\mathcal{O}(N + \lambda 2^{m+l})$, where λ is $3(m+l)$ or $4m+3l$. The resulting attacks are the best known key recovery attacks on 11-round and 12-round *Serpent*. The data, time, and memory complexity of the previously best known attack on 12-round *Serpent* are reduced by factor of $2^{7.5}$, $2^{11.7}$, and $2^{7.5}$, respectively. This paper also simulates the experiments of the improved Algorithm 2 in multidimensional linear cryptanalysis on 5-round *Serpent*.

Keywords: Multidimensional linear cryptanalysis, Linear Cryptanalysis, Serpent, Fast Fourier Transform, Fast Walsh Hadamard Transform.

1 Introduction

In 1993, Matsui [13] introduced the linear cryptanalysis and two algorithms, Algorithm 1 and Algorithm 2. These two algorithms exploit block cipher's linear approximation between the plaintext P, the ciphertext C and the secret key with a certain probability. Algorithm 2 is the modified version of Algorithm 1 by relaxing the first round and/or the last round of the linear approximation of Algorithm 1. Algorithm 2 can recover multiple secret key bits instead of one bit in Algorithm 1 with the same number of samples N required by the attack. However, the time complexity of the distillation phase of Algorithm 2 is much higher than that of Algorithm 1. Without loss of generality, the Substitution Permutation Network (SPN) n-round block cipher is considered, the last round of cipher is relaxed with l-bit subkey K_n of the last round key involved in the attack and N samples given. The time complexity for recovering l-bit K_n is $\mathcal{O}(2^l N)$. Matsui [15] suggested a method to reduce the time complexity to $\mathcal{O}(N + 2^{2l})$ by using one pre-computed table T. In 2007, Collard [7] *et al.* realized that table

T is a *circulant* matrix and applied the *Fast Fourier Transform* (FFT) to this table to reduce the time complexity to $\mathcal{O}(N + 3l2^l)$.

There are several important papers [12,2,9] which extended Matsui's algorithms, i.e., m linear approximations are used instead of one linear approximation in the attack. By combining m linear approximations together in the attack, the number of samples N is expected to be reduced. Two of the most outstanding models are the full Biryukov's model [11] and Hermelin's model [10] which is known as multidimensional linear cryptanalysis. The time complexity of the extended Algorithm 2 of these models is $\mathcal{O}(mN2^l)$. Even if we apply the algorithm in [7] to the full Biryukov's model, the time complexity of the extended Algorithm 2 is $\mathcal{O}(2^m N + 3l2^{l+m})$, which is still too high. Due to the high time complexity, the extended Algorithm 2 of these models can not be used to attack as many rounds as Algorithm 2 of Matsui or the extended Algorithm 1 does.

The extended Algorithm 2 in [10] computes the information for the attack by using a very natural algorithm which, however, it has a high computation cost, i.e., $\mathcal{O}(mN2^l)$. Our main contribution is to show that by applying the FFT and *Fast Walsh Hadamard Transform* ($FWHT$) to the extended Algorithm 2, the time complexity of the attack is reduced to $\mathcal{O}(N + \lambda 2^{m+l})$, where λ is $3(m + l)$ or $4m + 3l$. Firstly, we work with N pairs of samples (P, C) to extract the information for the attack only once. Secondly, we use two assistant tables to compute the information extracted from N samples. Finally, we apply FFT, $FWHT$ to these tables to derive the correct key from 2^l guessed keys. Based on the steps above, we introduce two methods, *method 1* and *method 2*, for two cases of the extended Algorithm 2 to reduce the time complexity to $\mathcal{O}(N + \lambda 2^{m+l})$, where λ is $3(m+l)$ or $4m+3l$. Applying these methods to the extended Algorithm 2, we attack 11-round *Serpent* with 2^{116} data complexity, $2^{107.5}$ time complexity, 2^{104} memory complexity. The previously best known result on 11-round *Serpent* is given in [7] with 2^{118} data complexity, $2^{114.3}$ time complexity, 2^{108} memory complexity. Therefore, the attack in this paper is the best on 11 rounds. For 12-round *Serpent*, we have two attacks. The first reduces the data and time complexity in [17] by factors of $2^{5.5}$ and $2^{19.4}$, respectively, but the memory requirements are higher by a large factor of 2^{100}. In comparison with [17], the second attack reduces the data, time, and memory complexity by $2^{7.5}$, $2^{11.7}$, and $2^{7.5}$, respectively. Hence our second attack on 12 rounds is much better than [17].

This paper is organized as follows. Section 2 introduces the background as well as some notations and definitions. Next, Section 3 gives the brief description of Algorithm 2 of the full Biryukov model and of the Hermelin model. In this section, the algorithm with time complexity $\mathcal{O}(mN2^l)$, which is used to compute the information for determining the correct key, is also explained. Then, Section 4 describes two efficient computing methods for two cases of the extended Algorithm 2. The experimental results on 5-round *Serpent* and the best cryptanalytic results against 11-round and 12-round *Serpent* are reported in Section 5. Lastly, Section 6 concludes this paper.

2 Notations and Background

The notations and definitions used in the rest of paper are introduced in this section. For the sake of convenience we follow the notations in [9].

Let $V_m = GF(2)^m$ denote the space of m-dimensional binary vectors. If $a = (a_1, \ldots, a_m), b = (b_1, \ldots, b_m)$ are two vectors in V_m, its inner product $a \cdot b$ is defined as follows: $a \cdot b = \bigoplus_{i=1}^{m} a_i b_i$. The function $f : V_m \rightarrow V_1$ is called a Boolean function. The function $f = (f_1, \ldots, f_m) : V_l \rightarrow V_m$ is called a vectorial Boolean function, where $f_i : V_l \rightarrow V_1$ is a Boolean function for $i = 1, \cdots, m$.

Let X be a random variable in V_m and $p_\eta = Pr(X = \eta)$, where $\eta \in V_m$. Then $p = (p_0, p_1, \ldots, p_{2^m-1})$ is the probability distribution of random variable X. If we associate with a vectorial Boolean function $f : V_l \rightarrow V_m$ a random variable $Y := f(X)$, where X is uniformly distributed in V_l, then the probability distribution of Y is $p(f) := (p_0(f), \ldots, p_{2^m-1}(f))$ where $p_\eta(f) = Pr(f(X) = \eta)$, for all $\eta \in V_m$.

The correlation between a binary random variable X and 0 is $\rho = Pr(X = 0) - Pr(X = 1)$. The value $\epsilon = \rho/2$ is called a bias of the random variable X. Let $g : V_m \rightarrow V_1$ be a Boolean function. Its correlation with 0 is defined as

$$\begin{aligned}\rho &= 2^{-m}(\#\{\eta \in V_m | g(\eta) = 0\}) - \#\{\eta \in V_m | g(\eta) = 1\}) \\ &= 2Pr(g(X) = 0) - 1,\end{aligned}$$

where X is uniformly distributed in V_m.

We recall important results on the *Fast Fourier Transform* [8], *Fast Walsh-Hadamard Transform* [16] and *Parseval's* theorem. Given a k-dimensional vector $\mathbf{E} = (E_1, \ldots, E_k)$ and a matrix $\mathbf{F}^{k \times k}$, we have k-dimensional vector $\mathbf{D} = \mathbf{F}\mathbf{E}^T$, where $\mathbf{E}^T$ is the transpose of $\mathbf{E}$. The matrix $\mathbf{F}$ is a *Hadamard* matrix if $\mathbf{F}(i,j) = (-1)^{ij}$ for $i, j = 0, \cdots, k-1$. The matrix $\mathbf{F}$ is a *Fourier* matrix if $\mathbf{F}(i,j) = (e)^{2\pi\sqrt{-1}ij/k}$, for $i, j = 0, \cdots, k-1$. If matrix $\mathbf{F}$ is either *Fourier* or *Hadamard*, then vector $\mathbf{D}$ can be computed with complexity $\mathcal{O}(k \log k)$ instead of $\mathcal{O}(k^2)$ by *FFT* or *FWHT*, respectively.

If $\mathbf{F}$ is a matrix then we denote $\mathbf{F}[\cdot, j], \mathbf{F}[i, \cdot]$ the j-th column and the i-th row of matrix $\mathbf{F}$, respectively.

3 Algorithm 2 of Multidimensional Linear Cryptanalysis

The m-dimensional linear cryptanalysis based on m linear approximations is introduced by Hermelin *et.al* [9]. In the attack, 2^m linear approximations which are the combinations of m linear approximation are exploited to reduce the data complexity. At first, we briefly review the multidimensional linear cryptanalysis as well as the results of [9], which are needed for our efficient computing methods. Then, we present the extended Algorithm 2 of multidimensional linear cryptanalysis of the full Biryukov's model and the Hermelin's model and we explain the high time complexity of the extended Algorithm 2 afterwards.

3.1 Construction of Multidimensional Probability Distribution

Let $f : V_l \rightarrow V_n$ be a vectorial Boolean function and binary vectors $w_i \in V_n, u_i \in V_l$ $(i = 1, \ldots, m)$ be selection patterns such that pairs (u_i, w_i) are linearly independent. Define the functions g_i as

$$g_i(\eta) := w_i \cdot f(\eta) \oplus u_i \cdot \eta, \quad \forall \eta \in V_l$$

and g_i has correlation ρ_i $(i = 1, \cdots, m)$. Then $\rho_1, \ldots, \rho_m$ are called the base-correlations and $g_1, \ldots, g_m$ are the base approximations of f. Let $g = (g_1, \ldots, g_m)$ be an m-dimensional vectorial Boolean function and denote $p = (p_0, \ldots, p_{2^m-1})$ probability distribution of g.

Lemma 1. *[9] Let $g = (g_1, \ldots, g_m) : V_l \rightarrow V_m$ be a vectorial Boolean function and $p = (p_0, \ldots, p_{2^m-1})$ its probability distribution. Then*

$$2^l p_\eta = 2^{-m} \sum_{a \in V_m} \sum_{b \in V_l} (-1)^{a \cdot (g(b) \oplus \eta)}, \quad \eta \in V_m.$$

Define

$$\rho(a) = 2^{-l} \sum_{b \in V_l} (-1)^{a \cdot g(b)} = Pr(a \cdot g(X) = 0) - Pr(a \cdot g(X) = 1),$$

where X is an random variable uniformly distributed in V_l. Thus, the combined approximation $a \cdot g$ has the correlation $\rho(a)$ for all $a \in V_m$.

Corollary 1. *Let $g : V_l \rightarrow V_m$ be a vectorial Boolean function with probability distribution p and correlations $\rho(a)$ of the combined approximations $a \cdot g$, for all $a \in V_m$. Then for $\eta \in V_m$,*

$$p_\eta = 2^{-m} \sum_{a \in V_m} (-1)^{a \cdot \eta} \rho(a). \tag{1}$$

3.2 Brief Analysis on Algorithm 2 of the Full Biryukov's Model and Hermelin's Model

Algorithm 2 of the multidimensional linear cryptanalysis is the extension of the Matsui's Algorithm 2 [13] by using 2^m linear approximations constructed from m base linear approximations. Let us consider the SPN block cipher with n rounds. Let $u \cdot P \oplus v \cdot C_{n-1} \oplus w \cdot \bar{K}$ be the $(n-1)$-round linear approximation of the block cipher, where u, v, w are the selection patterns of the plaintext P, the output C_{n-1} of $(n-1)$-th round, and the inner key $\bar{K}$, respectively. In the Matsui's Algorithm 2, this linear approximation is extended to n rounds by replacing $v \cdot C_{n-1} = z \cdot S^{-1}(K_n \oplus C_l)$, where z is a selection pattern corresponding to v on the input of the last round, S^{-1} is the inverse of the last S-box layer, K_n is the l bits out of the last round key and C_l is the l bits out of the ciphertext C involved in the attack. Let us denote $z \cdot S^{-1}(K_n \oplus C_l)$ function $f(K_n \oplus C_l)$, then we have the following linear approximation which is used in Matsui's Algorithm 2

$$u \cdot P \oplus f(K_n \oplus C_l) \oplus w \cdot \bar{K}.$$

Since the above linear approximation has correlation, the linear approximation $u \cdot P \oplus f(K_n \oplus C_l)$ also has correlation and then Matsui's Algorithm 2 [13] will determine the right key K_n based on this linear approximation. Instead of using one linear approximation, the extended Algorithm 2 uses many linear approximations. Let us assume that the extended Algorithm 2 is m-dimensional and recovers l-bit subkey K_n of the last round key. The m linear approximations used in the extended Algorithm 2 as base linear approximations are $g_i := u_i \cdot P \oplus f(K_n \oplus C_l)$ for $i = 1, \ldots, m$, where u_i are the selection patterns of plaintext. Define the vectorial boolean function $g = (g_1, \ldots, g_m)$. Then, $a \cdot g := a \cdot (u_1 \cdot P, \ldots, u_m \cdot P) \oplus f(K_n \oplus C_l)$ are combined approximations with the correlation $\rho(a)$ for $a \in V_m$.

The aim of the extend Algorithm 2 is to exploit 2^m combined approximations $a \cdot g$ in order to reduce the the data complexity N, i.e, [10]$N \sim \mathcal{O}(1/c)$, where $c = \sum_{\forall a \in V_m, a \neq 0} \rho^2(a)$ is called a capacity of the system. In practice, the correlation $\rho(a)$ of $a \cdot g$ is computed by *Piling-Up Lemma* [9]. The probability distribution $p = (p_0, \ldots, p_{2^m-1})$ of g is not uniformly distributed. The distillation phase [2,3] working on N samples (P, C) helps us to derive the correct K_n from 2^l guessed keys K. The time complexity of this phase is the major factor in the total time complexity of the attack. Let $\mathbf{p}^{2^l \times 2^m}$ be the matrix in which each row is empirical probability distributions of g for each candidate key K, i.e., $\mathbf{p}[K, \cdot] = (\mathbf{p}[K, 0], \ldots, \mathbf{p}[K, 2^{m-1}])$ $(\forall K \in V_l)$. In [11], Hermelin *et al.* proved that the full Biryukov's model is equal to the convolution model and the distillation phase of the convolution model uses the same algorithm with the Hermelin's model of Algorithm 2 to compute the empirical m-dimensional probability distribution of g, i.e., $\mathbf{p}$. The algorithm in [11] is as follows:

```
Input: $N$ pairs of plaintext-ciphertexts $\{(P_1, C_1), \ldots, (P_N, C_N)\}$,
       $g = (g_1, \ldots, g_m)$
Output: the matrix $\mathbf{p}$ of empirical probability distribution for all
        candidate $K_n \in V_l$
foreach $K \in V_l$, $\eta \in V_m$ do
    set $\mathbf{p}[K, \eta] = 0$.
end
foreach $K \in V_l$, $t$:=0,..., $N$ do
    Calculate $\eta = g(P_t, C_t, K) = (g_1(P_t, C_t, K), \ldots, g_m(P_t, C_t, K))$;
    $\mathbf{p}[K, \eta]$++;
end
foreach $K \in V_l$, $\eta \in V_m$ do
    $\mathbf{p}[K, \eta] = \mathbf{p}[K, \eta]/N$.
end
```

Algorithm 1. The Hermelin's Algorithm to compute the $\mathbf{p}$ with given N samples plaintext-ciphertext

In Hermelin's Algorithm, we use only m linear approximations $g_1, \ldots, g_m$ and the distillation phase computes all values of $\mathbf{p}[K, \cdot]$ $(\forall K \in V_l)$ with $mN2^l$ time complexity. Based on the *Wrong Key Hypothesis* [10], if K is the wrong key, its $\mathbf{p}[K,\cdot]$ is uniformly distributed and the right key K's $\mathbf{p}[K,\cdot]$ is non uniformly distributed. The χ^2 or LLR statistic [10] is applied to determine what kind of probability distribution for all $\mathbf{p}[K,\cdot]$ $(\forall K \in V_l)$. Then, the correct key K is determined by the information received from the chosen statistic.

Although the algorithm used to compute all $\mathbf{p}[K,\cdot]$ $(\forall K \in V_l)$ directly from N samples (P, C) is very elegant and can be easily implemented, its complexity is much higher in comparison to the extended Algorithm 1 [11,14] or to Algorithm 2 of Matsui. In cryptanalysis, it is the bottleneck in terms of time complexity. Therefore the extended Algorithm 2 is not comparable to Algorithm 2 of Matsui in terms of the number of rounds attacked or time complexity.

4 Efficient Computation of Distillation Phase of Extended Algorithm 2 of Matsui

In this section, we describe two efficient computing methods, *method 1* and *method 2*, for two cases in multidimensional linear cryptanalysis. These methods use Lemma 1, Corollary 1, the FFT, $FWHT$ and two pre-computed tables T, E to avoid repeatedly working on N samples and 2^l guessed keys K_n in order to reduce the time complexity from $\mathcal{O}(mN2^l)$ to $\mathcal{O}(N + \lambda 2^{m+l})$, where λ is a positive number which is specified for each case.

4.1 Case 1 and *Method 1*

In the first case, we have N pairs of samples (P, C) and m base linear approximations $g_i := u_i \cdot P \oplus f(K_n \oplus C_l)$ for $i = 1, \ldots, m$, where C_l is l bits out of ciphertext C involved in attack. All m base approximation g_i share one function $f(\cdot)$ and m selection patterns of plaintext $u_1, \ldots, u_m$ are linearly independent. Suppose $g = (g_1, \ldots, g_m)$, we then need to compute the empirical probability distribution matrix $\mathbf{p}$ of g.

Lemma 2. *For the first case, there exists a method (method 1) which computes the empirical probability distribution* $\mathbf{p}$ *of* g *with* $\mathcal{O}(N + (4m + 3l)2^{m+l})$ *time complexity.*

Due to the word count limit of this paper, the proof and description of *method 1* are presented in Appendix A.

4.2 Case 2 and *Method 2*

In this case, the first and the last round keys of cipher are considered in the attack. We have N pairs of samples (P, C) and m base linear approximations $g_i := f_1^i(K_1 \oplus P_{l_1}) \oplus f_2(K_n \oplus C_{l_2})$ for $i = 1, \ldots, m$, where K_1 is l_1 bits of the first round key, P_{l_1} is l_1 bits of the plaintext P, K_n is l_2 bits of the last round key, C_{l_2} is l_2 bits of the ciphertext C, f_1^i, f_2 $(i = 1, \ldots, m)$ are the boolean functions in [13,7]. The functions $f_1^i(\cdot)$ are constructed in the same way of $f_2(\cdot)$ for $i = 1, \ldots, m$. Suppose $g = (g_1, \ldots, g_m)$, we then need to compute the empirical probability distribution matrix $\mathbf{p}$ of g.

Lemma 3. *For the second case, there exists a method (method 2) which computes the empirical probability distribution* $\mathbf{p}$ *of* g *with* $\mathcal{O}(N + 3(l_1 + l_2 + m)2^{l_1+l_2+m})$ *time complexity.*

Due to the word count limit of this paper, the proof and description of *method 2* are presented in Appendix B.

5 Results on Cryptanalysis of Serpent

In this section, the experimental results of 5-round *Serpent* are reported first. *Serpent* [1] is the SPN block cipher which is one of the five AES finalists. *Serpent* has 128-bit block size and supports 128-bit, 192-bit and 256-bit keys. Based on the experimental results, we manage to confirm some facts claimed in [7] for the extended Algorithm 2. Then, we describe our attacks against 11-round and 12-round *Serpent*. The cryptanalytic results show that the multidimensional linear cryptanalysis can reduce the number of samples N and the time complexity in comparison to other methods, i.e., the linear cryptanalysis of Matsui and the differential-linear cryptanalysis on *Serpent* [17].

5.1 Experimental Results on 5-Round Serpent

In this subsection, the experiment on 5-round Serpent aims to confirm the results in [3,10] by using the extended Algorithm 2 with *method 1*. We call the extended Algorithm 2 with *method 1* or *method 2* improved extended Algorithm 2. For simplicity, the χ^2 statistic is applied to 4-, 7-, 10-dimensional linear cryptanalysis with $N = 2^{20}, 2^{21}, \ldots, 2^{26}$ and $l = 12$. The capacities are $2^{-23}, 2^{-22}, 2^{-21}$ for 4-, 7-, 10-dimensional linear cryptanalysis, respectively. If the correct l-bit key K_n has rank r out of 2^l possible guessed keys, then the attack obtains an $(l - \log_2 r)$-bit advantage over exhaustive search [3]. We test fifty keys and compute the advantage [2,3] for each key. Then, we construct the comparison table of average advantages for 4-, 7-, and 10-dimensional linear cryptanalysis for each set of N pairs. Table 1 displays the average of advantages for each set of N samples and 4-, 7-, 10-dimensional linear cryptanalysis.

The results confirm the facts that the extended Algorithm 2 with χ_2 statistic does not work as well as expected and when $m = 4$ the average of advantages is the highest. Furthermore, the experiments tell us that, given capacity c and if $N = 2^2/c$,then the advantage ≥ 1 of certain key has more than 50% success

Table 1. The average of advantages on 5-round *Serpent*

$\log_2(N)$	20	21	22	23	24	25	26
m=4	1.30	1.45	1.60	1.64	1.65	1.65	1.55
m=7	1.46	1.50	1.39	1.19	1.29	1.35	1.37
m=10	1.21	1.19	1.30	1.25	1.38	1.55	1.52

probability. According to the experimental results in [3], if the combination of LLR statistic and maximum KL is used instead of χ^2 statistic, then the multidimensional linear cryptanalysis works better, i.e., the average of advantages is much higher for each set of N samples.

5.2 Cryptanalysis of 11-Round and 12-Round Reduced Serpent

Before going to the cryptanalysis of 11-round and 12-round *Serpent*, we recall the fact that both methods do not require any decryption or encryption operation as that in Algorithm 2 of Matsui or in the differential-linear cryptanalysis [17]. We only need to extract the subset of bits involved in the plaintext P and the ciphertext C to compute the table E. The linear cryptanalysis can reach 11-round reduced *Serpent* by Algorithm 2 of Matsui based on the 9-round linear approximation [1,7,5]. The improved extended Algorithm 2 uses the 9-round linear approximation in [7,6,4] which is called *9-round linear approximation of Collard*. The 10-round linear approximation is constructed by adding one more round before the above 9-round linear approximation. The 11-round linear approximation, which is constructed from this 10-round linear approximation with the last round S_4 relaxed, is as follows:

$$\underbrace{S_2 \longrightarrow \underbrace{S_3 \longrightarrow \ldots \longrightarrow S_3}_{\text{9-round linear approximation of Collard}}}_{\text{10-round}} \longrightarrow \underbrace{S_4}_{\text{round relaxed}} .$$

In S_2 at the first round, there are 15 active S-boxes. The output masks of these active S-boxes described in hexadecimal number are:

S-box	0	3	5	6	9	10	11	13	15	16	17	24	28	29	30
output-mask	e	d	7	c	3	6	4	b	4	b	2	1	1	e	8

Table 2. Comparison of the attacks against reduced-round *Serpent*

Round		complexity		
		data	time	memory
11	Lin.cryptanalysis [5]	2^{118} KP	2^{166} En	2^{121}
	Lin.cryptanalysis [5]	2^{118} KP	$2^{173.5}$ En	2^{97}
	Lin.cryptanalysis [7]	2^{118} KP	$2^{114.3}$ En	2^{108}
	MultiDim.cryptanalysis [this paper](*method 1*)	2^{116} KP	$2^{107.5}$ En	2^{104}
	MultiDim.cryptanalysis [this paper](*method 1*)	2^{118} KP	$2^{109.5}$ En	2^{100}
12	Differential-Lin.cryptanalysis [17]	$2^{123.5}$ CP	$2^{249.4}$ En	$2^{128.5}$
	MultiDim.cryptanalysis [this paper](*method 2*)	2^{118} KP	$2^{228.8}$ En	2^{228}
	MultiDim.cryptanalysis [this paper](*method 1*)	2^{116} KP	$2^{237.5}$ En	2^{121}

En - Encryptions, KP - Known Plaintexts, CP - Chosen Plaintexts.

Based on the observation on active S-boxes of S_2, we choose 56 linearly independent selection patterns of plaintext $u_1, \dots, u_{56}$, i.e., we have 56-dimensional linear cryptanalysis attack.

If S_4 at the 11-th round has 12 active S-boxes, then in the 11-th round, the number of key bits l is 48 and the capacity c of 56-dimensional system is computed as follows:

$$\begin{aligned} c &= \sum_{a \neq 0, a \in V_{56}} \rho^2(a) \\ &= (2^{-57})^2 \sum_{m=0}^{11} C_m^{11} 8^m 2^{11-m} 4^4 (2^{-4m} 2^{-2(11-m)} 2^{-8}) \\ &= 2^{-114}. \end{aligned} \tag{2}$$

Let the number of samples required N be 2^{116}. If *method 1* is used, then the total time complexity is $(N/352 + 2^{111.6}/352)$ which is equal to $2^{107.5}$ 11-round *Serpent* encryptions and the memory will be 2^{104}.

If S_4 at the 11-round has 11 active S-boxes, then in the 11-th round, l is 44 and the capacity is 2^{-116}. Let N be 2^{118}, then the total time complexity is $(N/352 + 2^{107.5}/352)$ which is equal to $2^{109.5}$ 11-round *Serpent* encryptions and the memory is 2^{100}.

We attack the 12-round *Serpent* by using the above 11-round linear approximation and relaxing 12-th round S_4, which has 11 active S-boxes.

$$\underbrace{S_1 \longrightarrow S_2 \longrightarrow \underbrace{S_3 \longrightarrow \dots \longrightarrow S_3}_{\text{9-round linear approximation of Collard}}}_{\text{11-round}} \longrightarrow \underbrace{S_4}_{\text{round relaxed}} .$$

The capacity is 2^{-116}. Let $N = 2^{118}$, $l_1 = 128$ bits and $l_2 = 44$ bits. Based on *method 2*, the total time complexity is $(N/384 + 2^{237.4}/384)$ which is equal to $2^{228.8}$ 12-round *Serpent* encryptions and the memory is 2^{228}.

We can attack 12-round *Serpent* by *method 1* by relaxing the first and the 12-th rounds. We search for all 2^{128} possible keys in S_1 at the first round and use *method 1* for the other 11 rounds similarly to the case $N = 2^{116}$. Then the time and memory complexity are $2^{237.5}, 2^{121}$. Table 2 shows that the improved extended Algorithm 2 is better than all the previously known algorithms, i.e., linear cryptanalysis and differential-linear cryptanalysis, on 11-round and 12-round *Serpent*.

6 Conclusion

We studied two methods to reduce the time complexity in distillation phase of extended Algorithm 2 from $\mathcal{O}(mN2^l)$ to $\mathcal{O}(N + \lambda 2^{m+l})$, where m is the number of dimension of the attack, N is the number of samples needed, l is the number of key bits in the first and/or in the last rounds. These methods are introduced when we combine the Corollary 1, the Lemma 1, exploiting the repeated structure of data and key and using 2 assistant pre-computed tables T, E. Applying $FFT, FWHT$ to the tables T, E, we have two efficient computing methods to determine the correct key out of 2^l guessed keys .

We have simulated the experiments on 5-round *Serpent* to check the improved extended Algorithm 2 and to confirm claims in [3,10]. Based on the results of the experiments, we develop attack on 11-round, 12-round *Serpent* by using the improved extended Algorithm 2. These results are the best among those reported so far. On 12-round *Serpent*, we can reduce the data complexity, time and memory complexity of the previously best known attack by factors of $2^{7.5}$, $2^{11.7}$ and $2^{7.5}$, respectively. The result of 11-round *Serpent* is even better than the best currently reported, i.e., data complexity, time complexity, and memory complexity are reduced by factor of $2^2, 2^7$, and 2^4, respectively. The improved extended Algorithm 2 is competitive to the extended Algorithm 1 and Algorithm 2 of Matsui in terms of data complexity, time complexity and the number of rounds attacked. The extended Algorithm 2 usually involves many active S-boxes in the outer round(s) because it uses many linear approximations. Intuitively, it is easy to reach the limit of complexity in terms of time complexity and memory complexity. It implies that the bound of number of rounds attacked in improved extended Algorithm 2 is close to that in Algorithm 2 of Matsui somehow.

Acknowledgements

This work was supported in part by the Singapore National Research Foundation under Research Grant NRF-CRP2- 2007-03. The first author is supported by the Singapore International Graduate (SINGA) Scholarship.

References

1. Biham, E., Dunkelman, O., Keller, N.: Linear cryptanalysis of reduced round serpent. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 16–27. Springer, Heidelberg (2002)
2. Biryukov, A., De Cannière, C., Quisquater, M.: On multiple linear approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
3. Cho, J.Y., Hermelin, M., Nyberg, K.: A new technique for multidimensional linear cryptanalysis with applications on reduced round serpent. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 383–398. Springer, Heidelberg (2009)
4. Collard, B.: Private Communication (2010)
5. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Improved and multiple linear cryptanalysis of reduced round serpent. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 51–65. Springer, Heidelberg (2008)
6. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent - Description of the Linear Approximations (2007) (unpublished manuscript)

7. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Improving the time complexity of matsui's linear cryptanalysis. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 77–88. Springer, Heidelberg (2007)
8. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms. Hill Higher Education. McGraw-Hill Higher Education, New York (2001)
9. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional linear cryptanalysis of reduced round serpent. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 203–215. Springer, Heidelberg (2008)
10. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional extension of matsui's algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009)
11. Hermelin, M., Nyberg, K.: Dependent linear approximations: The algorithm of biryukov and others revisited. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 318–333. Springer, Heidelberg (2010)
12. Kaliski Jr., B.S., Robshaw, M.J.B.: Linear cryptanalysis using multiple approximations. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)
13. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
14. Nguyen, P.H., Wei, L., Wang, H., Ling, S.: On multidimensional linear cryptanalysis. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 37–52. Springer, Heidelberg (2010)
15. Matsui, M.: The first experimental cryptanalysis of the data encryption standard. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 1–11. Springer, Heidelberg (1994)
16. Rao Yarlagadda, R.K., Hershey, J.E.: Hadamard Matrix Analysis and Synthesis: with Applications to Communications and Signal/image Processing. Kluwer Academic Publishers, Norwell (1997)
17. Dunkelman, O., Indesteege, S., Keller, N.: A differential-linear attack on 12-round serpent. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 308–321. Springer, Heidelberg (2008)

Appendix

A Proof and Description of Lemma 2

Proof. The Lemma 1 and the corollary 1 show that the probability distribution p of g can be calculated from correlations $\rho(a)$ of combined approximations $a \cdot g$ for $a \in V_m$. The $a \cdot g$ combined approximations are defined as follows:

$$a \cdot g := \begin{cases} 0 & \text{if } a \text{ is a zero vector} \\ a \cdot (u_1 P, \ldots, u_m P) \oplus f(K_n \oplus C_l) & a \in V_m \ , a \neq 0 \ . \end{cases}$$

Let $\mathbf{r}^{2^m \times 2^l}[a, K]$ be the matrix whose elements $\mathbf{r}[a, K]$ are the correlations of linear approximations $a \cdot g$ when $K_n = K$. We have 2 following cases:

1. a is zero vector. Since $a \cdot g$ is equal to 0, $\mathbf{r}[0, K] = 1$, $(\forall K \in V_l)$.

2. a is not zero vector. *method 1* below will compute $\mathbf{r}[a,K]$ $(K \in V_l)$.

Since our aim is to reduce as much as possible the number of times of working with N samples for computing $\mathbf{r}[a,K]$, we will need 2 assistant tables $T^{2^m \times 2^l}$ and $E^{2^m \times 2^l}$.

In practice, the empirical $\mathbf{r}[a,K]$ is computed as follows [7]:

$$\mathbf{r}[a,K] = \frac{\sharp\{(P_t,C_t) : a\cdot g(P_t,C_t,K)=0\} - \sharp\{(P_t,C_t) : a\cdot g(P_t,C_t,K)=1\}}{N}, \quad t=1,\ldots,N.$$

We go through all $a \in V_m, (P_t,C_t)$ for $t=1,\ldots,N$ to compute the value of table T. The elements of this table are computed as follows:

$$a(u_1P_t,\ldots,u_mP_t) = \begin{cases} 0 \rightarrow T(a,(C_l)_t) ++ \\ 1 \rightarrow T(a,(C_l)_t) -- \end{cases},$$

where $(C_l)_t$ is the l bits of ciphertext C_t involved in function $f(\cdot)$, $\forall a \in V_m$ and (P_t,C_t) $(t=1,\ldots,N)$.

Let $T[a,v] = T[a,v]/N$ $(a \in V_m, v \in V_l)$, then

$$\mathbf{r}[a,K] = \sum_{v=0}^{2^l-1} (-1)^{f(K\oplus v)} T[a,v].$$

Let $\mathbf{S}^{2^l \times 2^l}$ be a matrix with $\mathbf{S}[i,j] = (-1)^{f(i\oplus j)}$ $(\forall i,j \in V_l)$. Hence,

$$\mathbf{r}[a,\cdot] = \mathbf{S}T[a,\cdot]. \tag{3}$$

Based on the proposition in [7], matrix $\mathbf{S}$ is a *circulant* matrix. Therefore, vector $\mathbf{r}[a,\cdot]$ is computed with complexity $3l2^l$ and we only need 2^l memory units to store the first column $((-1)^{f(0)},\ldots,(-1)^{f(2^l-1)})^T$ of matrix $\mathbf{S}$ to calculate $\mathbf{r}[a,\cdot]$.

The time complexity is very high if we directly compute T from N samples, i.e., $\mathcal{O}(2^m N)$. We show a way to compute T more efficiently by assistant table E. We construct $\mathbf{E}^{2^m \times 2^l}$ as follows:

$$E[h_1,h_2] = \sharp\{(P_t,C_t), t=1,\ldots,N : (u_1\cdot P_t,\ldots,u_m\cdot P_t) = h_1 \text{ and } (C_l)_t = h_2\}, \quad h_1 \in V_m, h_2 \in V_l.$$

The time complexity for constructing $\mathbf{E}$ is mN and the memory complexity is 2^{m+l}. If we do m computing $u_1\cdot P_t,\ldots,u_m\cdot P_t$ for each sample (P_t,C_t) at the same time, then the time complexity for constructing E is N.

For $\forall v \in V_l, \forall a \in V_m$, We have

$$T[a,v] = \sum_{h_1=0}^{2^m-1} (-1)^{ah_1} E[h_1,v]. \tag{4}$$

Let $\mathbf{H}^{2^m \times 2^m}$ be a matrix with $H[i,j] = (-1)^{ij}$ $(\forall i, j \in V_m)$. Then,

$$T[\cdot, v] = \mathbf{H}E[\cdot, v]. \tag{5}$$

Since matrix $\mathbf{H}$ is a *Hadamard* matrix, $T[\cdot, v]$ can be computed using $FWHT$ algorithm with time complexity $m2^m$ and memory complexity 2^m.

In summary, the steps taken in *method 1* are as follows:

1. Compute $\mathbf{E}^{2^m \times 2^l}$ with N time complexity and 2^{m+l} memory complexity.
2. For $\forall v \in V_l$, compute $T[\cdot, v] = (T[0, v], \ldots, T[2^m - 1, v])$ using $FWHT$ algorithm in (5). Time complexity is $2^l m 2^m (= m2^{l+m})$ and memory complexity is 2^{m+l}.
3. For $\forall a \in V_m, a \neq 0$, compute $\mathbf{r}[a, \cdot]$ by the algorithm of *circulant* matrix [7] in (3). Time complexity is $2^m 3l2^l (= 3l2^{l+m})$ and memory complexity is 2^l.
4. If a is a zero vector, then $\mathbf{r}[a, \cdot]$ is vector 1.
5. We need 2^{m+l} memory units to store all $\mathbf{r}[a, K]$ $(\forall K \in V_l, \forall a \in V_m)$.
6. If we need to compute $\mathbf{p}[K, \cdot]$ $(\forall K \in V_l)$, then time complexity is $3m2^m2^l (= 3m2^{l+m})$ and 2^{m+l} memory units needed for all $\mathbf{p}$.

The total time complexity is $\mathcal{O}(N + 2^{l+m}(4m + 3l))$ and memory complexity is $\mathcal{O}(2^{m+l})$.

B Proof and Description of Lemma 3

Proof. The combined approximations $a \cdot g$ are defined as follows:

$$a \cdot g = \begin{cases} a \cdot f_1(K_1 \oplus P_{l_1}) \oplus f_2(K_n \oplus C_{l_2}) \\ \quad \text{if } a \text{ is not a zero vector} \\ 0 \text{ if } a \text{ is a zero vector} \end{cases},$$

where $f_1(K_1 \oplus P_{l_1}) = (f_1^1(K_1 \oplus P_{l_1}), \ldots, f_1^m(K_1 \oplus P_{l_1}))$, $\forall a \in V_m$.

Let $\mathbf{r}^{2^m \times 2^{l_1} \times 2^{l_2}}$ be the matrix containing all the correlations of $a \cdot g$ with $a \in V_m, K_1 \in V_{l_1}, K_n \in V_{l_2}$. We have the following 2 cases:

1. If a is vector 0, then $\mathbf{r}[a, k_1, k_2] = 1$ $(\forall k_1 \in V_{l_1}, \forall k_2 \in V_{l_2})$.
2. If a is not vector 0, then we can compute $\mathbf{r}[a, k_1, k_2]$ $(\forall k_1 \in V_{l_1}, \forall k_2 \in V_{l_2})$ by *method 2* below.

We need 2 assistant tables $T^{2^m \times 2^{l_1} \times 2^{l_2}}$ and $E^{2^{l_1} \times 2^{l_2}}$.

Let

$$\mathbf{r}[a, k_1, \cdot] = (\mathbf{r}[a, k_1, 0], \ldots, \mathbf{r}[a, k_1, 2^{l_2} - 1]),$$
$$T[a, k_1, \cdot] = (T[a, k_1, 0], \ldots, T[a, k_1, 2^{l_2} - 1]).$$

For $\forall a \in V_m$, (P_t, C_t) $(t = 1, \ldots, N)$, $\forall k_1 \in V_{l_1}$, the elements of table T are calculated as follows:

$$af_1(k_1 \oplus (P_{l_1})_t) = \begin{cases} 0 \rightarrow T[a, k_1, (C_{l_2})_t] + + \\ 1 \rightarrow T[a, k_1, (C_{l_2})_t] - - \end{cases},$$

where $(P_{l_1})_t$ and $(C_{l_2})_t$ are l_1 bits of P_t and l_2 bits of C_t involved in the attack, respectively.

Let $T[a, k_1, k_2] = T[a, k_1, k_2]/N$.

With the same argument in *method 1*, we have

$$\mathbf{r}[a, k_1, k_2] = \sum_{v=0}^{2^{l_2}-1} (-1)^{f_2(k_2 \oplus v)} T[a, k_1, v].$$

Let $\mathbf{S}_2^{2^{l_2} \times 2^{l_2}}$ be a matrix with $\mathbf{S}_2[i, j] = (-1)^{f_2(i \oplus j)}$ $(\forall i, j \in V_{l_2})$. Hence,

$$\mathbf{r}[a, k_1, \cdot] = \mathbf{S}_2 T[a, k_1, \cdot]. \tag{6}$$

According to [7], matrix $\mathbf{S}_2$ is a *circulant* matrix. Hence, vector $\mathbf{r}[a, k_1, \cdot]$ is computed with time complexity $3l_2 2^{l_2}$ and 2^{l_2} memory units.

In order to efficiently compute $T[a, k_1, v]$, we construct the table $\mathbf{E}^{2^{l_1} \times 2^{l_2}}$ as follows:

$$E[h_1, h_2] = \sharp\{(P_t, C_t), t = 1, \ldots, N : (P_{l_1})_t = h_1, (C_{l_2})_t = h_2\}, \forall h_1 \in V_{l_1}, \forall h_2 \in V_{l_2}.$$

The time complexity to construct $\mathbf{E}$ is N and the memory complexity is $2^{l_1+l_2}$. Then,

$$T[a, k_1, v] = \sum_{u=0}^{2^{l_1}-1} (-1)^{af_1(k_1 \oplus u)} E[u, v]. \tag{7}$$

Let $\mathbf{S}_a^{2^{l_1} \times 2^{l_1}}$ be a matrix with $\mathbf{S}_a[i, j] = (-1)^{af_1(i \oplus j)}$ $(\forall i, j \in V_{l_1})$. Then,

$$T[a, \cdot, k_2] = \mathbf{S}_a E[\cdot, v], \tag{8}$$

where $E[., v]$ is the column of matrix $\mathbf{E}$.

Based on [7], matrix S_a is a *circulant* matrix. Hence, the vector $T[a, \cdot, k_2]$ is computed in $3l_1 2^{l_1}$ computations and the memory needed is 2^{l_1}.

In summary, the steps taken in *method 2* are as follows:

1. Construct table $\mathbf{E}$. Time complexity is N and the memory needed for $\mathbf{E}$ is $2^{l_1+l_2}$.
2. For $\forall a \in V_m, \forall v \in V_{l_2}$, we compute all $T[a, \cdot, k_2]$ by (8). Time complexity is $2^m 2^{l_2} 3l_1 2^{l_1} (= 3l_1 2^{l_1+l_2+m})$. We need $2^{m+l_1+l_2}$ memory units for all $T[a, k_1, v]$ $(\forall a \in V_m, \forall k_1 \in V_{l_1}, \forall v \in V_{l_2})$.
3. For $\forall a \in V_m, a \neq 0, \forall k_1 \in V_{l_1}$ compute all $\mathbf{r}[a, k_1, \cdot]$ by (6). The time complexity is $2^m 2^{l_1} 3l_2 2^{l_2} (= 3l_2 2^{l_1+l_2+m})$. If a is a vector zero, the $\mathbf{r}[a, k_1, \cdot]$ is the vector 1 $(\forall k_1 \in V_{l_1})$. We need $2^{l_1+l_2+m}$ memory for $\mathbf{r}$.
4. If we need to compute $\mathbf{p}$, then time complexity is $3m 2^m 2^{l_1+l_2} (= 3m 2^{l_1+l_2+m})$ and memory complexity is $2^{m+l_1+l_2}$.

Consequently, the total time complexity is $\mathcal{O}(N + 3(l_1 + l_2 + m) 2^{l_1+l_2+m})$ and the memory complexity is $\mathcal{O}(2^{m+l_1+l_2})$.