Error-Tolerant Side-Channel Cube Attack Revisited

Zhenqi Li[1(✉)], Bin Zhang[2,3], Arnab Roy[4,5], and Junfeng Fan[6]

[1] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
lizhenqi@tca.iscas.ac.cn

[2] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China

[3] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
zhangbin@tca.iscas.ac.cn

[4] University of Luxembourg, Luxembourg, Luxembourg

[5] Technical University of Denmark, Kongens Lyngby, Denmark
arroy@dtu.dk

[6] Nationz Technologies Inc, Shenzhen, China
fanjunfeng@gmail.com

Abstract. Error-tolerant side-channel cube attacks have been recently introduced as an efficient cryptanalytic technique against block ciphers. The known Dinur-Shamir model and its extensions require error-free data for at least part of the measurements. Then, a new model was proposed at CHES 2013, which can recover the key in the scenario that each measurement contains noise. The key recovery problem is converted to a decoding problem under a binary symmetric channel. In this paper, we propose a high error-tolerant side-channel cube attack. The error-tolerant rate is significantly improved by utilizing the polynomial approximation and a new variant of cube attack. The simulation results on PRESENT show that given about $2^{21.2}$ measurements, each with an error probability of 40.5 %, the new model achieves a success probability of 50 % for the key recovery. The error-tolerant level can be enhanced further if the attacker can obtain more measurements.

Keywords: Cube attack · Side-channel attack · PRESENT

This work was supported by the National Grand Fundamental Research 973 Program of China(Grant No. 2013CB338002), the programs of the National Natural Science Foundation of China (Grant No. 60833008, 60603018, 61173134, 91118006, 61272476).

Most of Arnab Roy's work was done when he was in the University of Luxembourg.

A. Joux and A. Youssef (Eds.): SAC 2014, LNCS 8781, pp. 261–277, 2014.
DOI: 10.1007/978-3-319-13051-4_16

1 Introduction

Cube attack was formally proposed by Dinur and Shamir at Eurocrypt 2009 [8] as an efficient cryptanalytic technique which can be applied to many types of well-designed cryptosystems by exploiting an low degree multivariate polynomial of a single output bit. It is an extension of high-order differential attacks [13] and algebraic IV differential attacks [17,18]. It shows superior performance on several stream ciphers [1,2,7,8,11], however, most block ciphers are immune to it, as they iterate a highly non-linear round function for a number of times and the degree of the polynomial for the ciphertext bits is much higher.

Since the master polynomials of some intermediate variables in the early rounds are of relatively low degree, cube attack becomes a convincing method to attack block ciphers by combining physical attacks, where the attackers can exploit some leaked information about the intermediate variables, i.e., state registers. The attacker only needs to learn the value of a single wire or register in each execution, it is thus ideal for probing attacks. The main challenge is to overcome the measurement noise, thus how to launch an efficient error-tolerant side-channel cube attack in a realistic setting is a highly interesting topic.

Dinur and Shamir initialized the first study on error-tolerant side-channel cube attack (ET-SCCA) [10]. They treat the uncertain bits as new erasure variables and it was further enhanced in [6,9] by utilizing more trivial equations of high dimensional cubes to correct the errors. The success of this model is based on an assumption that the adversary possesses the exact knowledge of error positions and partial measurements are error-free. Then, at CHES 2013 [19], Li *et. al.* proposed a new model, which can recover the key when each measurement contains noise. The key recovery problem is converted to decoding a $[L, n]$ linear code. However, the error-tolerant level is still very low.

This paper introduces a new ET-SCCA which can tolerate heavy noise interference. The error-tolerant rate can be improved significantly by utilizing the polynomial approximation technique and applying a new variant of cube attack. The main idea of polynomial approximation is to appropriately remove some key variables to reduce the code dimension n of a $[L, n]$ code. Moreover, a new variant of cube attack is proposed, inspired by the idea of dynamic cube attack [7]. The main idea is to increase the number of linear equations, i.e., code length L, by adaptively choosing the plaintext. Consequently, the bound of error probability has been refined. Compared with the simulation results on PRESENT in [19], the error probability for each measurement can be improved to 40.5 % given about $2^{21.2}$ measurements and $2^{27.6}$ time complexity. The error-tolerant rate can be enhanced further if the attacker can obtain more measurements. Table 1 summarized our simulation results on PRESENT.

This paper is organized as follows. We first introduce the basic idea of cube attack and ET-SCCA in Sect. 2. In Sect. 3, we present the new model. Error probability evaluation is developed and analyzed in Sect. 4. Section 5 presents the simulations on PRESENT. The comparison is given in Sect. 6, followed by some further discussions. Finally, we conclude the paper in Sect. 7.

Table 1. Simulation results on PRESENT

Time complexity	*Data (measurements)*	*Error probability*	*Scenario*
$2^{31.6}$	$2^{10.1}$	23.2 %	*Lower measurements*
$2^{27.6}$	$2^{16.2}$	29.5 %	*Balanced*
$2^{27.6}$	$2^{21.2}$	40.5 %	*Higher error tolerance*

The success probability is about 50 %.
The memory requirement is negligible.

2 Preliminaries

2.1 Cube Attack

Consider a block cipher $\mathbb{T}$ and its encryption function $(c_1, ..., c_m) = \mathrm{E}(k_1, ..., k_n, v_1, ..., v_m)$, where c_i, k_j and v_s are ciphertext, encryption key and plaintext bits, respectively. One can always represent c_i, $i \in [1, m]$, with a multivariate polynomial in the plaintext and key bits, namely, $c_i = p(k_1, ..., k_n, v_1, ..., v_m)$. Let $I \subseteq \{1, ..., m\}$ be an index subset, and $t_I = \prod_{i \in I} v_i$, the polynomial p is divided into two parts: $p(k_1, ..., k_n, v_1, ..., v_m) = t_I \cdot p_{S(I)} + q(k_1, ..., k_n, v_1, ..., v_m)$, where no item in q contains t_I. Here $p_{S(I)}$ is called the superpoly of I in p. A maxterm of p is a term t_I such that $\deg(p_{S(I)}) \equiv 1$ verified by the BLR test [4] and this $p_{S(I)}$ is called maxterm equation of t_I.

Example 1. Let $p(k_1, k_2, k_3, v_1, v_2, v_3) = v_2v_3k_1 + v_2v_3k_2 + v_1v_2v_3 + v_1k_2k_3 + k_2k_3 + v_3 + k_1 + 1$ be a polynomial of degree 3 in 3 secret variables and 3 public variables. Let $I = \{2, 3\}$ be an index subset of the public variables. We can represent p as $p(k_1, k_2, k_3, v_1, v_2, v_3) = v_2v_3(k_1 + k_2 + v_1) + (v_1k_2k_3 + k_2k_3 + v_3 + k_1 + 1)$, where $t_I = v_2v_3$, $p_{S(I)} = k_1 + k_2 + v_1$ and $q(k_1, k_2, k_3, v_1, v_2, v_3) = v_1k_2k_3 + k_2k_3 + v_3 + k_1 + 1$.

Let d be the size of I, then a *cube* on I is defined as a set C_I of 2^d vectors that cover all the possible combinations of t_I and leave all the other variables undetermined. Any vector $\tau \in C_I$ defines a new derived polynomial $p_{|\tau}$ with $n - d$ variables. Summing these derived polynomials over all the 2^d possible vectors in C_I results in exactly $p_{S(I)}$ (cf. Theorem 1, [8]). For p and I defined in Example 1, we have $C_I = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, where $\tau_1 = [k_1, k_2, k_3, v_1, 0, 0], \tau_2 = [k_1, k_2, k_3, v_1, 0, 1], \tau_3 = [k_1, k_2, k_3, v_1, 1, 0]$, and $\tau_4 = [k_1, k_2, k_3, v_1, 1, 1]$. It is easy to verify that $p_{|\tau_1} + p_{|\tau_2} + p_{|\tau_3} + p_{|\tau_4} = k_1 + k_2 + v_1 = p_{S(I)}$. Here $p_{S(I)}$ is called the maxterm equation of t_I. In the off-line phase, the attacker tries to find as many maxterms and their corresponding maxterm equations as possible.

In the on-line phase, the secret key is fixed. The attacker chooses plaintexts $\tau \in C_I$ and obtains the evaluation of p at τ. By summing up $p_{|\tau_i}$ for all the 2^d vectors in C_I, he obtains $p_{S(I)}$, a linear equation in k_i. The attacker repeats this process for all the maxterms found in the off-line phase, and obtains a group of equations, which he can solve to recover the key.

2.2 Error-Tolerant Side-Channel Cube Attack(ET-SCCA)

At CHES 2013, Li *et. al.* [19] proposed a new model for ET-SCCA, which can retrieve the key when *all* the leaked bits are noisy. The leaked data observed is regarded as the received channel output of some linear code transmitted through a binary symmetric channel (BSC). The problem of recovering the n secret key bits in L linear equations can be considered as the problem of decoding a binary linear $[L, n]$ code with L being the code length and n the dimension as follows.

$$\begin{cases} l_1: \ a_1^1 k_1 + a_1^2 k_2 + ... + a_1^n k_n = b_1 \\ l_2: \ a_2^1 k_1 + a_2^2 k_2 + ... + a_2^n k_n = b_2 \\ \quad \vdots \\ l_L: \ a_L^1 k_1 + a_L^2 k_2 + ... + a_L^n k_n = b_L \end{cases} \tag{1}$$

where $a_i^j \in \{0,1\}$ $(1 \leq i \leq L, 1 \leq j \leq n)$ denotes the coefficient. Note that $b_i \in \{0,1\}$ is obtained by summing up the evaluation of the maxterm equation over the i^{th} cube C_i, namely, $b_i = \sum_{\tau \in C_i} p_{|\tau}$. The value of $p_{|\tau}$ is obtained via measurements. Ideally, the measurement is error-free and the attacker obtains the correct sequence $B = [b_1, b_2, ..., b_L]$. In reality, however, the attacker is likely to observe a different sequence $Z = z_1, z_2, ..., z_L$ due to the measurement errors.

Denote q as the probability that a bit may flip in each measurement and assume that $q < 1/2$, then $1 - q = 1/2 + \mu$ is the probability of an accurate measurement and $\mu = 0$ means a random guess. Since $b_i = \sum_{\tau \in C_i} p_{|\tau}$, and C_i has $t = 2^{\bar{d}}$ elements ($\bar{d}$ is the average size of cubes), and each measurement can be treated as an independent event, according to the piling-up lemma [14], $Pr\{b_i = z_i\} \stackrel{\Delta}{=} 1 - p = \frac{1}{2} + 2^{t-1}\mu^t$. Thus, each z_i can be regarded as the output of a BSC with $p = 1/2 - \varepsilon$ ($\varepsilon = 2^{t-1}\mu^t$) being the crossover probability. Therefore, the key recovery problem is converted to decoding a $[L, n]$ linear code. Maximum likelihood decoding (ML-decoding, see Appendix C) is used and they derive the error-tolerant bound in Lemma 1.

Lemma 1. *To ensure* 50 % *success probability of decoding a* $[L, n]$ *code to retrieve the key, the error probability* q *of each measurement should satisfy* $q \leq \frac{1}{2} \cdot (1 - (\frac{0.35 \cdot n}{L})^{\frac{1}{2 \cdot t}} \cdot 2^{\frac{1}{t}})$, *where* $t = 2^{\bar{d}}$ *denotes the number of summations to evaluate each linear equation.*

The simulation results on PRESENT-80 show that given about $2^{10.2}$ measurements, each with an error probability $q = 19.4\,\%$, it achieves 50.1 % of success rate for the key recovery. However, the error-tolerant rate is still very low.

3 A New ET-SCCA with Higher Error-Tolerant Rate

3.1 Polynomial Approximation

The main target is to remove several secret variables while keeping the number of maxterm equations reduced as few as possible. In this way, the code dimension

n can be reduced while keeping the code length L reduced as little as possible. However, removing secret variables might be a challenging task as previous studies on Trivium [8], Serpent [9,10], KATAN [12], LBlock [20] and PRESENT [19] show that most of the maxterm equations have a low density. Removing secret variables will probably lead to the reduction of the maxterm equations. We propose two basic strategies of removing key variables as follows.

Lower Reduction Factor. The removed secret variables should not be those that solely exist in the maxterm equations and should be those that exist in the maxterm equations with multiple secret variables. (e.g., suppose we have derived 2 maxterm equations, one is $k_1 + k_2$ and the other is $k_3 + 1$, then the removed secret variables should contain k_1 or k_2, but not k_3, since removing k_3 will lead to the second maxterm equation become a trivial one.) Note that this selection process can be finished in the off-line phase, since all the maxterm equations are available. Suppose the number of removed key variables is n' and the number of maxterm equations reduced is $\gamma \cdot n'$, where γ is the reduction factor. $\gamma = 0$ means the removed secret variables will not influence the number of the maxterm equations and the value of γ depends on the choice of removed secret variables. The problem now convert to decoding a $[L - \gamma \cdot n', n - n']$ code.

Higher Approximation Rate. Suppose a polynomial p containing n secret variables and m public variables, the removed key set is $R = \{k_{i_1}, k_{i_2}, ..., k_{i_r}\}$, where $1 \leq i_q \leq n$, $1 \leq q \leq r$. The approximation rate between p and $\tilde{p}$ after removing variables in R is defined as $\Lambda(p, \tilde{p})|_R = e/2^{m+n} = 1/2 + \sigma$, where e is the number of the equal evaluations and σ $(0 < \sigma < 1/2)$[1] is the bias factor. In reality, there might be more than one leakage function. Suppose $P = \{p_1, p_2, ..., p_u\}$ are all the associated leakage functions and the corresponding removed key variable sets are $R_1, R_2, ..., R_u$ respectively[2], the average approximation rate is defined as

$$\bar{\Lambda} = \frac{\sum_{t=1}^{u} \Lambda(p_t, \tilde{p}_t)|_{R_t}}{u}. \tag{2}$$

The candidate key variables to remove should be those with maximum $\bar{\Lambda}$. Note that this process can also be finished in the off-line phase, i.e., all the removed key variables are set to 0 for the evaluation of $\tilde{p}$.

3.2 A New Variant of Cube Attack

The main idea is to increase the number of maxterm equations by choosing the static public variables, which are those variables that are not part of the cube variables. In the traditional applications of cube attacks and cube testers [1,2,8], these static variables will be set to constant values. We find that multiple

[1] For the case of $1/2 - \sigma$, convert it to $1/2 + \sigma$ by adding 1 to the evaluation of $\tilde{p}$

[2] $R_1 = \{k_{i_1}^{[1]}, k_{i_2}^{[1]}, ..., k_{i_{r_1}}^{[1]}\}$, $R_2 = \{k_{i_1}^{[2]}, k_{i_2}^{[2]}, ..., k_{i_{r_2}}^{[2]}\}$,...

maxterm equations can be derived for each maxterm by choosing static variables. In Example 1 of Sect. 2, the maxterm equation for the maxterm $t_I = v_2v_3$ is $p_{S(I)} = k_1 + k_2 + v_1$, where v_1 is a static variable. If we set $v_1 = 0$, then we can derive a maxterm equation $k_1 + k_2$. Similarly, if we set $v_1 = 1$, another variant maxterm equation $k_1 + k_2 + 1$ can be derived. Then, we have the following theorem (please refer to Appendix A for the details of the proof).

Theorem 1. *For the maxterm equation $p_{S(I)}$ of maxterm t_I, the number of variant maxterm equations which can be derived is at most 2^{m-d} and each can be classified into the following two types.*

1. $p^*_{S(I)} + C$, *where* $C \in \{0, 1\}$. *(Type I)*
2. $p^*_{S(I)} + C_0 + C_1k_{n_1} + C_2k_{n_2} + ... + C_rk_{n_r}$, *where* $C_i \in \{0, 1\}$, $\bigvee_{i=1}^{r} C_i \neq 0$ *and* C_0 *represents a constant term. (Type II)*

$p^*_{S(I)}$ *is the equation of* $p_{S(I)}$ *when we set all static variables to* 0.

The previous Example 1 of Sect. 2 describes the scenario of Type I. The following example shows the scenario of Type II.

Example 2. Suppose a polynomial $p = v_1v_2k_1 + v_1v_2v_3k_2 + v_3v_4k_1k_2k_3 + v_1v_2 = v_1v_2 \cdot (k_1 + v_3k_2 + 1) + v_3v_4k_1k_2k_3$, then $t_I = v_1v_2$ with $I = \{1, 2\}$ is a maxterm, $p_{S(I)} = k_1 + v_3k_2 + 1$ is the maxterm equation and $p^*_{S(I)} = k_1 + 1$.

The static variables is thus $\{v_3, v_4\}$. If we choose $v_3 = 1$, then a variant of maxterm equation appears as $k_1 + k_2 + 1 = p^*_{S(I)} + k_2$, which fits into Type II.

In the traditional cube attack, most of these variant maxterm equations are trivial and make no contribution to the key recovery. However, in our model, these variants can be treated as redundant information, which are beneficial to the decoding algorithm. For a linear code considering polynomial approximation $[L - \gamma \cdot n', n - n']$, the total number of maxterm equations can be increased by a factor of E. Now the problem of key recovery is converted to decoding a $[(L - \gamma \cdot n') \cdot E, n - n']$ linear code, where $1 \leq E \leq 2^{m-\bar{d}}$.

4 Error Probability Evaluation

By utilizing ML-decoding, we derive a new bound for error-tolerant rate in Corollary 1 (Please refer to Appendix B for the details of the proof).

Corollary 1. *To ensure 50 % success probability of decoding a $[L^*, n^*]$ code to retrieve the key, the error probability q of each measurement should satisfy*

$$q \leq \frac{1}{2} \cdot (1 - (\frac{0.35 \cdot n^*}{L^*})^{\frac{1}{2 \cdot t}} \cdot 2^{\frac{1}{t}}), \tag{3}$$

where $n^* = n - n'$, $L^* = (L - \gamma \cdot n') \cdot E$, $1 \leq E \leq 2^{m-\bar{d}}$ and $t = 2^{\bar{d}}$.

If $n' = 0$ and $E = 1$, it reduces to the original ET-SCCA. The cost for the polynomial approximation is that the removed key variables will add more noise to those associated maxterm equations, but this kind of noise can be ignored if we only remove a few key variables and keep the number of maxterm equations influenced as little as possible. Moveover, the rest of the n' key variables removed can be exhaustively searched. The cost for choosing static public variables is that the number of measurements will increase accordingly.

Suppose $L = 1000$, then the error probabilities under different number of removed key variables $n' = 0, 10, 30$ with $\gamma = 1$ and $E = 1$ are depicted in Fig. 1.

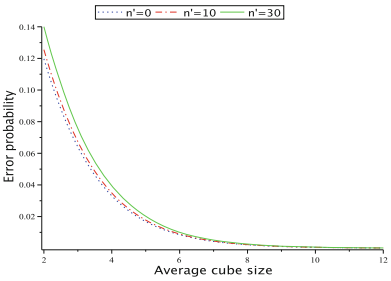

Fig. 1. Error probability q as a function of $\bar{d}$ (Given $n' = 0, 10, 30$, $\gamma = 1$, $E = 1$ and $L = 1000$)

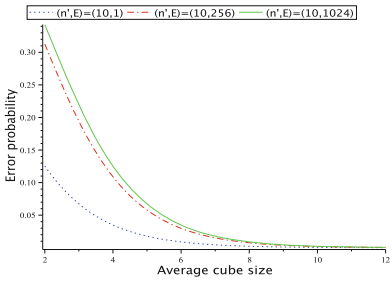

Fig. 2. Error probability q as a function of $\bar{d}$ (Given $n' = 10$, $\gamma = 1$, $E = 1, 256, 1024$ and $L = 1000$)

Figure 1 shows that the error probability gradually increased with the growth of n'. By applying the new variant of cube attack with $E = 1, 256$ and 1024, shown in Fig. 2, which demonstrates that the error probability increased with the growth of E. Similar results can also be obtained if we choose other size of L. These results demonstrate that the error probability can be further improved under the same noise channel and utilizing the same decoding algorithm.

5 Simulations on PRESENT

To compare our model with the original ET-SCCA in [19], we will apply the model to PRESENT-80, a standardized round based lightweight block cipher [5]. We assume PRESENT cipher is implemented on a 8-bit processor. Under Hamming weight leakage model, the attacker exploits the Hamming weight leakage containing noise when the state variables are loaded from memory to ALU.

5.1 Off-Line Phase

We enumerate all the small candidate cubes, each size is at most 2. The time complexity is thus $P = \binom{64}{1} \cdot 2 + \binom{64}{2} \cdot 2^2 = 2^{13}$ encryptions. The leakage function is the LSB (least significant bit) of the Hamming weight of the state byte after the

first round. There are altogether 8 state bytes $byte_1, byte_2, ..., byte_8$, corresponding to 8 leakage functions. We can derive 304 maxterm equations containing 64 key variables (Appendix D) and the average cube size is $\bar{d} = 1.9$. Compared with the special cube searching strategy in the original ET-SCCA [19], the process of off-line phase in our model requires no knowledge of the internal round function.

Now we need to figure out which key variables should be removed according to the distribution of key variables in all the maxterm equations. For the leakage function of $byte_1$ (or $byte_2$), there are only 16 maxterm equations (Tables 9, 10 Appendix D), each of which only contains a single key variable, it is thus hard to decide which key variables should be removed. For the leakage function of $byte_3$, considering those maxterm equations containing $\{k_{17}, k_{18}, k_{19}, k_{20}\}$, removing k_{17} will lead to 3 maxterm equations (corresponding to maxterm $\{2, 3\}$, $\{2, 4\}$ and $\{3, 4\}$ respectively) become trivial and removing k_{20} will lead to 2 maxterm equations (corresponding to maxterm $\{1, 2\}$ and $\{1, 3\}$ respectively) reduced for the leakage function of $byte_5$. Removing k_{18} or k_{19} is a good choice since it only lead to one maxterm equation (corresponding to maxterm $\{3\}$ or $\{2\}$) reduced for the leakage function of $byte_1$ and it will not affect other state bytes. We choose k_{18} as a representative variable. Similarly, we can also derive other representative variables $k_{22}, k_{26}, k_{30}, k_{34}, k_{38}, k_{42}, k_{46}$ from other ranges for the leakage function of $byte_3$. All the candidate key variables are summarized in Table 2.

Table 2. Candidate key variables for removing

State byte	*Candidate key variables*
$byte_3, byte_7$	$k_{18}, k_{22}, k_{26}, k_{30}, k_{34}, k_{38}, k_{42}, k_{46}$
$byte_4, byte_8$	$k_{50}, k_{54}, k_{58}, k_{62}, k_{66}, k_{70}, k_{74}, k_{78}$

These variables will not lead to any reduction of the maxterm equations for both $byte_5$ and $byte_6$, each only lead to one reduction of the maxterm equations for $byte_1$ or $byte_2$. Therefore, $n' \in [0, 16]$ and the reduction factor $\gamma = 1$.

5.2 Polynomial Approximation for PRESENT-80

Now we need to select the optimal combinations of these candidates that can maximize the approximation rate. We enumerate all the possible combinations and calculate the average approximation rate according to equation (2). The optimal combinations for each $|R| \in [1, 8]$ are listed in the following table.

From Table 3, we can see that the value of σ decreases with the growth of $|R|$. We will not consider those candidate key variables set with $|R| > 8$, since the bias σ becomes trivial and more removed key variables will add more noise to the evaluations of those associated maxterm equations.

Table 3. Optimal combinations

R	σ	R	σ
$\{k_{30}\}$	0.267	$\{k_{18}, k_{22}, k_{26}, k_{42}, k_{50}\}$	0.162
$\{k_{26}, k_{74}\}$	0.263	$\{k_{42}, k_{50}, k_{54}, k_{58}, k_{66}, k_{74}\}$	0.154
$\{k_{34}, k_{38}, k_{78}\}$	0.207	$\{k_{22}, k_{50}, k_{54}, k_{58}, k_{62}, k_{70}, k_{74}\}$	0.152
$\{k_{18}, k_{66}, k_{70}, k_{78}\}$	0.176	$\{k_{18}, k_{26}, k_{30}, k_{34}, k_{38}, k_{42}, k_{46}, k_{54}\}$	0.148

5.3 On-Line Phase

From the previous analysis, we know that $n' = |R|$ and $\gamma = 1$. The key recovery problem is now equivalent to decoding a $[L - n', n - n']$ linear code.

For the sake of comparison, grouping strategy and list decoding are also utilized in the model as in [19]. More precisely, all the key variables are divided into 4 groups G_1, G_2, G_3 and G_4 with several overlapping bits. ML-decoding is applied in each group as a direct application of the ML-decoding has a time complexity of $2^{64-n'}$. To increase the success probability, we save a candidate list of T closest solutions for each group. The configurations with $R = \{k_{18}, k_{66}, k_{70}, k_{78}\}$ are listed in the following Table 4.

Table 4. Groups with $R = \{k_{18}, k_{66}, k_{70}, k_{78}\}$

Group	$[L, n]$	*Key bits*	*Overlapping bits*
G_1	$[113, 23]$	$[k_{17}, k_{19}, ..., k_{40}]$	8 with G_2
G_2	$[114, 24]$	$[k_{33}, k_{34}, ..., k_{56}]$	8 with G_1, 8 with G_3
G_3	$[94, 19]$	$[k_{49}, k_{50}, ..., k_{68}]$	8 with G_2, 8 with G_4
G_4	$[92, 17]$	$[k_{61}, k_{62}, ..., k_{80}]$	8 with G_3

We have simulated the decoding algorithm for 100 runs with $T = 200$. For each run, we randomly generate a key and construct the linear code in each group. The noise was simulated by a random binary number generator according to the crossover probability p (e.g., suppose $k_0 = 1, k_1 = 0$ and there is a maxterm equation $1 + k_0 + k_1 = 0$, the value 0 will flip to 1 with probability p and remain unchanged with probability $1-p$). We have conducted the simulation for 10 times and the average number of successful decoding out of a batch of 100 runs are recorded. The comparison results are shown in Fig. 3, which indicates that under the same success probability, decoding with removing 4 key variables can tolerate more noise. The comparisons with various size of R are summarized in Fig. 4, which demonstrates that under the same success probability, more noise can be tolerated with the growth of $|R|$.

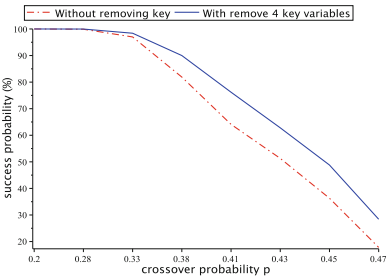

Fig. 3. Comparison results of list decoding

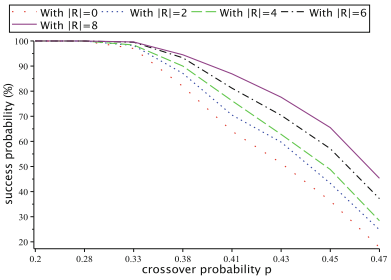

Fig. 4. Comparison results of list decoding with various size of R

p is the crossover probability for each evaluation of the maxterm equation. Since $p = 1/2 - 2^{t-1}\mu^t$, $t = 2^{\bar{d}}$ and $1 - q = 1/2 + \mu$, the error probability q for each measurement are listed in Table 5.

Table 5. q with various R under the decoding success probability of 50 %

$\|R\|$	p	q	$\|R\|$	p	q
0	0.428	0.191	6	0.459	0.232
2	0.439	0.204	8	0.467	0.247
4	0.447	0.214			

The whole attack contains two phases, the first phase is the decoding in each group. The results are the candidate lists. Denote t_i as the time complexity of decoding in group G_i, η as the number of the groups and n_i as the code dimension in G_i, the time complexity is thus $\sum_{i=1}^{\eta} t_i$, where $t_i = 2^{n_i}$ key trials. The second is the verification phase. Since each candidate only contains $64 - n'$ master key variables, we need to verify it by combining the removed n' key variables and the rest of 16 master key bits, using the known plaintext/ciphertext pairs. The time complexity is thus $V(T) = T^{\eta} \cdot 2^{n'+16}/2^r$ encryptions[3], where 2^r is the reduction factor related to the number of overlapping bits. Therefore, the attack complexity is bounded by $max\{ \sum_{i=1}^{\eta} t_i,\ T^{\eta} \cdot 2^{n'+16}/2^r \}$. The attack results with various R on PRESENT are given in Table 6.

From Table 6, we can see that the model can tolerate more errors if more candidate key variables are removed. The growth of the key variables removed will lead to the increase of the time complexity in the verification phase and will add more noise due to the polynomial approximation. Therefore, R should be carefully chosen so that the error-tolerant rate can be optimal while keeping the time complexity practical.

[3] The verification complexity in [19] is actually incorrect. When 16 bits of the key are missing, they can be exhaustively searched (in time complexity of 2^{16}).

Table 6. Simulations by utilizing polynomial approximation

Size of R	*Time*	*Data (measurements)*	*Reduction factor r*	*Error probability*
0	$2^{25.1}$	1152	24	19.1 %
2	$2^{24.7}$	1148	24	20.4 %
4	$2^{27.6}$	1144	23	21.4 %
6	$2^{31.6}$	1140	21	23.2 %
8	$2^{31.6}$	1136	23	24.7 %

The decoding success probability is about 50 %.

5.4 Applying the New Variant of Cube Attack to the On-Line Phase

In this section, we will show that the error-tolerant rate can be further improved by choosing static public variables. The key recovery problem is now converted to decoding a $[(L-\gamma \cdot n') \cdot E, n-n']$ linear code, where $1 \leq E \leq 2^{m-\bar{d}}$ represents the size of the redundant maxterm equations.

For PRESENT-80, $m = 64$ and $\bar{d} \approx 2$, then $1 \leq E \leq 2^{62}$. All the complexity remain unaltered except for the data complexity. Deriving more redundant maxterm equations will lead to higher measurements. Suppose $n' = 4$, the simulation results under various E are summarized in Table 7.

Table 7. Simulations by choosing static variables

E	*Data (measurements)*	*Total size of L*	*Error probability*
1	$2^{10.2}$	304	21.4 %
2^6	$2^{16.2}$	19456	29.5 %
2^8	$2^{18.2}$	77824	34.1 %
2^{10}	$2^{20.2}$	311296	38.6 %
2^{11}	$2^{21.2}$	622592	40.5 %

The decoding success probability is about 50 %.
The time complexity is $2^{27.6}$ encryptions (verification phase).
$R = \{k_{18}, k_{66}, k_{70}, k_{78}\}$.

From Table 7, it is shown that the error-tolerant rate increased with the growth of the E. It can be increased closely to 50 % on the condition that we can obtain more measurements, which means that our model can still work even if the measurement contains heavy noise.

6 Comparison and Discussions

6.1 ET-SCCA Comparisons

Compared with the original ET-SCCA [19], the error-tolerant level of the new ET-SCCA is improved significantly by utilizing the polynomial approximation and applying the new variant of cube attack. It is more flexible, since the attacker can choose appropriate size of E according to his ability (e.g., accuracy of the measurements). The comparison results are summarized in Table 8.

Table 8. Comparison between the original ET-SCCA and our model

Time complexity	*Data (measurements)*	*Error probability*	*Reference*
$2^{34.6}$	$2^{10.2}$	19.1 %	[19]
$2^{31.6}$	$2^{10.1}$	24.7 %	this paper
$2^{27.6}$	$2^{21.2}$	40.5 %	this paper

The success probability is about 50 % for both models.

6.2 Motivation of the New Variant of Cube Attack

The motivation of the new variant of cube attack comes from the dynamic cube attack [7]. The difference is that dynamic cube attacks transform some of the static public variables to dynamic variables and each one of these dynamic variables is assigned a function that depends on some of the cube variables and some expressions of secret variables. These functions are carefully chosen usually to zero some state bits to simplify the expression and amplify the bias of the cube tester. It requires a more complex analysis of the internal structure of the cipher. Moveover, the main purpose of dynamic cube attack is to improve the standard cube testers and construct a more efficient distinguisher, then filtering right key using this distinguisher. While the new variant (also mentioned in [15]) applied in this paper is to derive more redundant maxterm equations to facilitate the decoding process, which requires no knowledge of the round function.

6.3 About the Definition of Maxterm Equation

Recall the formal definition of maxterm equation in [8]. The maxterm equation $p_{S(I)}$ of a maxterm t_I satisfies $\deg(p_{S(I)}) \equiv 1$, which holds whenever static variables are 0s or 1s. In most applications, e.g., Trivium, all the maxterm equations are derived when they are set to 0s. However, some researchers [3] verified all the maxterm equations derived from Trivium [8] by chosen the static variables. Among 1000 runs, each of which a random IV was chosen, almost all the maxterm equations pass through a linear test with probability of about 50 %, i.e., $\deg(p_{S(I)}) \equiv 1$ cannot hold for half of the runs. All these will have a negligible influence to our simulation on PRESENT, since the number of variant maxterm equations (i.e., E) is low and we can always get enough maxterm equations by choosing the static variables.

6.4 Attacking Implementations with Masking

Masking is a widely used countermeasure against side-channel attacks. The principle is to randomly split every sensitive variable (e.g., variables involving secret keys) occurring in the computation into $d+1$ shares, where d is called the masking order and plays the role of a security parameter. Suppose a state byte S is split into $d+1$ random shares $S_0, S_1, ..., S_d$, satisfying $S_0 \oplus S_1 \oplus ... \oplus S_d = S$ and the computations are on the masked data. Suppose the attacker observe the value of each share containing noise as $S_0 \oplus e_0$, $S_1 \oplus e_1$,...,$S_d \oplus e_d$, where e_i is the observation noise. By summing all these values up, $S \oplus \sum_{i=0}^{d} e_i$ can be derived. Compared with an implementation without masking, the only influence to the ET-SCCA is that the observation noise for a masking implementation is relatively higher, which is exponentially increased with the growth of the masking order d (according to piling-up lemma). However, in reality, d is small since almost all the current masking schemes suffered from the efficiency problems when d becomes bigger. Therefore, we believe that our model can still be available to a implementation with masking.

7 Conclusion and Open Problems

In this paper, we have revisited the error-tolerant side-channel cube attack and proposed a more robust model. By appropriately utilizing the polynomial approximation technique, the error-tolerant rate can be improved compared to the original ET-SCCA. We also presented an efficient way of finding the key variables that should be removed, by defining the average approximation rate. Moreover, a new variant of cube attack was proposed inspired by the idea of dynamic cube attack. The error-tolerant rate has been refined. Both theoretical analysis and simulation results indicated that the improved model is more flexible, exploiting measurements with heavy noise interference, which solves one of the open problems listed in [19]. The simulation results on PRESENT show that given about $2^{21.2}$ measurements, each with an error probability of 40.5 %, it achieves 50 % success probability of the key recovery. The error-tolerant level can be enhanced further if the attacker can obtain more measurements. Hence, we believe these results have both a theoretical and practical relevance.

A Proof of Theorem 1

Proof. The fact that we can derive at most 2^{m-d} variant maxterm equations is obvious, since the number of static public variables is $m-d$. The master multivariate polynomial p can be represented as $p(k_1, ..., k_n, v_1, ..., v_m) = t_I \cdot p_{S(I)} + q(k_1, ..., k_n, v_1, ..., v_m)$. Since t_I is a maxterm, then the degree of $p_{S(I)}$ in secret variables is $\deg(p_{S(I)}) \equiv 1$. Then $p_{S(I)}$ can be represented as $p_{S(I)} = C + C_0 + C_1 k_1 + C_2 k_2 + ... + C_n k_n$, where $C \in \{0, 1\}$ is a constant and C_i, $0 \leq i \leq n$ contains only static public variables.

Since $\deg(p_{S(I)}) \equiv 1$, then $\bigvee_{i=1}^{n} C_i \neq 0$, which means that there is at least one key variables k_i with $C_i = 1$. Suppose the set of key variables with all their coefficients equal to 1 is $K = \{k_{i_1}, k_{i_2}, ..., k_{i_u}\}$ and each $C_{i_t} = 1$, $1 \leq t \leq u$. By setting all the static public variables to 0, then $p^*_{S(I)} = C + \sum_{j=i_1}^{i_u} k_j$. The variant maxterm equations by choosing those static public variables can thus be represented as

$$p'_{S(I)} = p^*_{S(I)} + C_0 + \sum_{j \notin \{i_1, i_2, ..., i_u\}} C_j k_j \tag{4}$$

Then, if the chosen static public variables make all the $C_j = 0$, where $1 \leq j \leq n$ and $j \notin \{i_1, i_2, ..., i_u\}$, then equation (4) is the Type I variant. Inversely, if the chosen static public variables make that there is at least one $C_j \neq 0$, where $1 \leq j \leq n$ and $j \notin \{i_1, i_2, ..., i_u\}$, the equation (4) is the Type II variant. □

B Proof of Corollary 1

Proof. The key recovery problem is now converted to decoding a $[(L - \gamma \cdot n') \cdot E, n - n']$ linear code, where $1 \leq E \leq 2^{m-\bar{d}}$. Suppose $L^* = (L - \gamma \cdot n') \cdot E$ and $n^* = n - n'$. Recall that the error probability p for each evaluation of the maxterm equation is $p = 1/2 - \varepsilon$. The capacity of BSC can be approximated as $C(p) \approx \varepsilon^2 \cdot 2/(ln(2))$. Simulations [16] show that the critical length $L^* \geq 0.35 \cdot n^* \cdot \varepsilon^{-2}$ provides the probability of successful decoding close to 1/2. Thus we get $\varepsilon \geq \sqrt{\frac{0.35 \cdot n^*}{L^*}}$. Since $\varepsilon = 2^{t-1}\mu^t$ holds, then we can derive $\mu \geq (\frac{0.35 \cdot n^*}{L^*})^{\frac{1}{2 \cdot t}} \cdot 2^{\frac{1}{t}-1}$. From $q = 1/2 - \mu$, we have $q \leq \frac{1}{2} \cdot (1 - (\frac{0.35 \cdot n^*}{L^*})^{\frac{1}{2 \cdot t}} \cdot 2^{\frac{1}{t}})$. □

C ML-decoding

Siegenthaler [16] firstly proposed the use of ML-decoding in cryptanalysis of a stream cipher by exhaustively searching through all the codewords of $[L, n]$ code. The complexity is about $O(2^n \cdot n/C(p))$. Let $A = (a_i^j)_{L \times n}$ $(1 \leq i \leq L, 1 \leq j \leq n)$ be the generator matrix of (1) and A_i denote the i-th row vector of A. The aim of the decoding is to find the closet codeword $(b_1, b_2, ..., b_L)$ to the received vector $(z_1, z_2, ..., z_L)$, and decode the key variables $\mathbf{k} = (k_1, k_2, ..., k_n)$ such that $b_i = \mathbf{k} \cdot A_i^T$, where T denotes the matrix transpose, i.e., find such $\mathbf{k}$ that minimizes $D(\mathbf{k}) = \sum_{i=1}^{L} (z_i \bigoplus b_i)$. It is known that ML-decoding is optimal since it has the smallest error probability among all decoding algorithms.

D Maxterm Equations for All the 8 Leakage Functions

Table 9. Maxterms and maxterm equations

Leakage function of $byte_1$			
Cube Indexes	*Maxterm equations*	*Cube indexes*	*Maxterm equations*
{2}	$k19$	{3}	$1 + k18$
{6}	$k23$	{7}	$1 + k22$
{10}	$k27$	{11}	$1 + k26$
{14}	$k31$	{15}	$1 + k30$
{18}	$k35$	{19}	$1 + k34$
{22}	$k39$	{23}	$1 + k38$
{26}	$k43$	{27}	$1 + k42$
{30}	$k47$	{31}	$1 + k46$
Leakage function of $byte_2$			
{34}	$k51$	{35}	$1 + k50$
{38}	$k55$	{39}	$1 + k54$
{42}	$k59$	{43}	$1 + k58$
{46}	$k63$	{47}	$1 + k62$
{50}	$k67$	{51}	$1 + k66$
{54}	$k71$	{55}	$1 + k70$
{58}	$k75$	{59}	$1 + k74$
{62}	$k79$	{63}	$1 + k78$
Leakage function of $byte_3$			
{1, 2}	$k19 + k20$	{1, 3}	$k18 + k20$
{1, 4}	$k18 + k19$	{2, 3}	$k17$
{2, 4}	$1 + k17$	{3, 4}	$1 + k17$
{5, 6}	$k23 + k24$	{5, 7}	$k22 + k24$
{5, 8}	$k22 + k23$	{6, 7}	$k21$
{6, 8}	$1 + k21$	{7, 8}	$1 + k21$
{9, 10}	$k27 + k28$	{9, 11}	$k26 + k28$
{9, 12}	$k26 + k27$	{10, 11}	$k25$
{10, 12}	$1 + k25$	{11, 12}	$1 + k25$
{13, 14}	$k31 + k32$	{13, 15}	$k30 + k32$
{13, 16}	$k30 + k31$	{14, 15}	$k29$
{14, 16}	$1 + k29$	{15, 16}	$1 + k29$
{17, 18}	$k35 + k36$	{17, 19}	$k34 + k36$
{17, 20}	$k34 + k35$	{18, 19}	$k33$
{18, 20}	$1 + k33$	{19, 20}	$1 + k33$
{21, 22}	$k39 + k40$	{21, 23}	$k38 + k40$
{21, 24}	$k38 + k39$	{22, 23}	$k37$
{22, 24}	$1 + k37$	{23, 24}	$1 + k37$
{25, 26}	$k43 + k44$	{25, 27}	$k42 + k44$
{25, 28}	$k42 + k43$	{26, 27}	$k41$
{26, 28}	$1 + k41$	{27, 28}	$1 + k41$
{29, 30}	$k47 + k48$	{29, 31}	$k46 + k48$
{29, 32}	$k46 + k47$	{30, 31}	$k45$
{30, 32}	$1 + k45$	{31, 32}	$1 + k45$
Leakage function of $byte_4$			
{33, 34}	$k51 + k52$	{33, 35}	$k50 + k52$
{33, 36}	$k50 + k51$	{34, 35}	$k49$
{34, 36}	$1 + k49$	{35, 36}	$1 + k49$
{37, 38}	$k55 + k56$	{37, 39}	$k54 + k56$
{37, 40}	$k54 + k55$	{38, 39}	$k53$
{38, 40}	$1 + k53$	{39, 40}	$1 + k53$
{41, 42}	$k59 + k60$	{41, 43}	$k58 + k60$
{41, 44}	$k58 + k59$	{42, 43}	$k57$
{42, 44}	$1 + k57$	{43, 44}	$1 + k57$
{45, 46}	$k63 + k64$	{45, 47}	$k62 + k64$
{45, 48}	$k62 + k63$	{46, 47}	$k61$
{46, 48}	$1 + k61$	{47, 48}	$1 + k61$
{49, 50}	$k67 + k68$	{49, 51}	$k66 + k68$
{49, 52}	$k66 + k67$	{50, 51}	$k65$
{50, 52}	$1 + k65$	{51, 52}	$1 + k65$
{53, 54}	$k71 + k72$	{53, 55}	$k70 + k72$
{53, 56}	$k70 + k71$	{54, 55}	$k69$
{54, 56}	$1 + k69$	{55, 56}	$1 + k69$
{57, 58}	$k75 + k76$	{57, 59}	$k74 + k76$
{57, 60}	$k74 + k75$	{58, 59}	$k73$
{58, 60}	$1 + k73$	{59, 60}	$1 + k73$
{61, 62}	$k79 + k80$	{61, 63}	$k78 + k80$
{61, 64}	$k78 + k79$	{62, 63}	$k77$
{62, 64}	$1 + k77$	{63, 64}	$1 + k77$

Table 10. Maxterms and maxterm equations

Leakage function of $byte_5$			
Cube Indexes	*Maxterm equations*	*Cube indexes*	*Maxterm equations*
{1, 2}	$1 + k20$	{1, 3}	$k20$
{1, 4}	$1 + k18 + k19$	{2, 4}	$1 + k17$
{3, 4}	$k17$	{5, 6}	$1 + k24$
{5, 7}	$k24$	{5, 8}	$1 + k22 + k23$
{6, 8}	$1 + k21$	{7, 8}	$k21$
{9, 10}	$1 + k28$	{9, 11}	$k28$
{9, 12}	$1 + k26 + k27$	{10, 12}	$1 + k25$
{11, 12}	$k25$	{13, 14}	$1 + k32$
{13, 15}	$k32$	{13, 16}	$1 + k30 + k31$
{14, 16}	$1 + k29$	{15, 16}	$k29$
{17, 18}	$1 + k36$	{17, 19}	$k36$
{17, 20}	$1 + k34 + k35$	{18, 20}	$1 + k33$
{19, 20}	$k33$	{21, 22}	$1 + k40$
{21, 23}	$k40$	{21, 24}	$1 + k38 + k39$
{22, 24}	$1 + k37$	{23, 24}	$k37$
{25, 26}	$1 + k44$	{25, 27}	$k44$
{25, 28}	$1 + k42 + k43$	{26, 28}	$1 + k41$
{27, 28}	$k41$	{29, 30}	$1 + k48$
{29, 31}	$k48$	{29, 32}	$1 + k46 + k47$
{30, 32}	$1 + k45$	{31, 32}	$k45$
Leakage function of $byte_6$			
{33, 34}	$1 + k52$	{33, 35}	$k52$
{33, 36}	$1 + k50 + k51$	{34, 36}	$1 + k49$
{35, 36}	$k49$	{37, 38}	$1 + k56$
{37, 39}	$k56$	{37, 40}	$1 + k54 + k55$
{38, 40}	$1 + k53$	{39, 40}	$k53$
{41, 42}	$1 + k60$	{41, 43}	$k60$
{41, 44}	$1 + k58 + k59$	{42, 44}	$1 + k57$
{43, 44}	$k57$	{45, 46}	$1 + k64$
{45, 47}	$k64$	{45, 48}	$1 + k62 + k63$
{46, 48}	$1 + k61$	{47, 48}	$k61$
{49, 50}	$1 + k68$	{49, 51}	$k68$
{49, 52}	$1 + k66 + k67$	{50, 52}	$1 + k65$
{51, 52}	$k65$	{53, 54}	$1 + k72$
{53, 55}	$k72$	{53, 56}	$1 + k70 + k71$
{54, 56}	$1 + k69$	{55, 56}	$k69$
{57, 58}	$1 + k76$	{57, 59}	$k76$
{57, 60}	$1 + k74 + k75$	{58, 60}	$1 + k73$
{59, 60}	$k73$	{61, 62}	$1 + k80$
{61, 63}	$k80$	{61, 64}	$1 + k78 + k79$
{62, 64}	$1 + k77$	{63, 64}	$k77$
Leakage function of $byte_7$			
{1, 2}	$k19 + k20$	{1, 3}	$k18 + k20$
{1, 4}	$k18 + k19$	{2, 3}	$1 + k17$
{2, 4}	$k17$	{3, 4}	$k17$
{5, 6}	$k23 + k24$	{5, 7}	$k22 + k24$
{5, 8}	$k22 + k23$	{6, 7}	$1 + k21$
{6, 8}	$k21$	{7, 8}	$k21$
{9, 10}	$k27 + k28$	{9, 11}	$k26 + k28$
{9, 12}	$k26 + k27$	{10, 11}	$1 + k25$
{10, 12}	$k25$	{11, 12}	$k25$
{13, 14}	$k31 + k32$	{13, 15}	$k30 + k32$
{13, 16}	$k30 + k31$	{14, 15}	$1 + k29$
{14, 16}	$k29$	{15, 16}	$k29$
{17, 18}	$k35 + k36$	{17, 19}	$k34 + k36$
{17, 20}	$k34 + k35$	{18, 19}	$1 + k33$
{18, 20}	$k33$	{19, 20}	$k33$
{21, 22}	$k39 + k40$	{21, 23}	$k38 + k40$
{21, 24}	$k38 + k39$	{22, 23}	$1 + k37$
{22, 24}	$k37$	{23, 24}	$k37$
{25, 26}	$k43 + k44$	{25, 27}	$k42 + k44$
{25, 28}	$k42 + k43$	{26, 27}	$1 + k41$
{26, 28}	$k41$	{27, 28}	$k41$
{29, 30}	$k47 + k48$	{29, 31}	$k46 + k48$
{29, 32}	$k46 + k47$	{30, 31}	$1 + k45$
{30, 32}	$k45$	{31, 32}	$k45$
Leakage function of $byte_8$			
{33, 34}	$k51 + k52$	{33, 35}	$k50 + k52$
{33, 36}	$k50 + k51$	{34, 35}	$1 + k49$
{34, 36}	$k49$	{35, 36}	$k49$
{37, 38}	$k55 + k56$	{37, 39}	$k54 + k56$
{37, 40}	$k54 + k55$	{38, 39}	$1 + k53$
{38, 40}	$k53$	{39, 40}	$k53$
{41, 42}	$k59 + k60$	{41, 43}	$k58 + k60$
{41, 44}	$k58 + k59$	{42, 43}	$1 + k57$
{42, 44}	$k57$	{43, 44}	$k57$
{45, 46}	$k63 + k64$	{45, 47}	$k62 + k64$
{45, 48}	$k62 + k63$	{46, 47}	$1 + k61$
{46, 48}	$k61$	{47, 48}	$k61$
{49, 50}	$k67 + k68$	{49, 51}	$k66 + k68$
{49, 52}	$k66 + k67$	{50, 51}	$1 + k65$
{50, 52}	$k65$	{51, 52}	$k65$
{53, 54}	$k71 + k72$	{53, 55}	$k70 + k72$
{53, 56}	$k70 + k71$	{54, 55}	$1 + k69$
{54, 56}	$k69$	{55, 56}	$k69$
{57, 58}	$k75 + k76$	{57, 59}	$k74 + k76$
{57, 60}	$k74 + k75$	{58, 59}	$1 + k73$
{58, 60}	$k73$	{59, 60}	$k73$
{61, 62}	$k79 + k80$	{61, 63}	$k78 + k80$
{61, 64}	$k78 + k79$	{62, 63}	$1 + k77$
{62, 64}	$k77$	{63, 64}	$k77$

References

1. Aumasson, J.-P., Dinur, I., Henzen, L., Meier, W. and Shamir, A.: Efficient FPGA implementations of high-dimensional cube testers on the stream cipher Grain-128. In: Special Purpose Hardware for Attacking Cryptographic Systems-SHARCS'09' (2009)
2. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)
3. Bedi, S.S., Rajesh Pillai, N.: Cube attacks on Trivium. Cryptology ePrint Archive. Report 2009/015 (2009)
4. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. J. Comput. Syst. Sci. **47**, 549–595 (1993)
5. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an altra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
6. Dinur, I., Shamir, A.: Applying cube attacks to stream ciphers in realistic scenarios. Crypt. Commun. **4**, 217–232 (2012)
7. Dinur, I., Shamir, A.: Breaking grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011)
8. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
9. Dinur, I., Shamir, A.: Generic analysis of small cryptographic leaks. In: 2010 Workshop on Fault Diagnosis and Tolerance in Cryptography. pp. 39–48 (2010)
10. Dinur, I., Shamir, A.: Side channel cube attacks on block ciphers. Cryptology ePrint Archive. Report 2009/127 (2009)
11. Fouque, P.-A., Vannet, T.: Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 502–517. Springer, Heidelberg (2014)
12. Bard, G.V., Courtois, N.T., Nakahara Jr., J., Sepehrdad, P., Zhang, B.: Algebraic, AIDA/Cube and side channel analysis of KATAN family of block ciphers. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 176–196. Springer, Heidelberg (2010)
13. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello Jr., D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography: Two Sides of One Tapestry. The Springer International Series in Engineering and Computer Science, vol. 276, pp. 227–233. Springer, New York (1994)
14. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
15. Quedenfeld, F.-M., Wolf, C.: Algebraic Properties of the Cube Attack. Cryptology ePrint Archive. Report 2013/800 (2013)
16. Siegenthaler, T.: Decrypting a class of stream ciphers using ciphertext only. IEEE Trans. Comput. **34**(1), 81–85 (1985)
17. Vielhaber, M.: AIDA Breaks (BIVIUM A and B) in 1 Minute Dual Core CPU Time. IACR Cryptology ePrint Archive, 402 (2009)
18. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. IACR Cryptology ePrint Archive, 413 (2007)

19. Li, Z., Zhang, B., Fan, J., Verbauwhede, I.: A new model for error-tolerant side-channel cube attacks. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 453–470. Springer, Heidelberg (2013)
20. Li, Z., Zhang, B., Yao, Y., Lin, D.: Cube cryptanalysis of LBlock with noisy leakage. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 141–155. Springer, Heidelberg (2013)