Noname manuscript No.
(will be inserted by the editor)

A Practical Iterative Side Channel Cube Attack on AES-128/256

Erfan Aghaee · Majid Rahimi · Hamed Yusefi

Received: date / Accepted: date

Abstract The Side Channel Cube Attack (SCCA) is a kind of Algebraic Side Channel Attack (ASCA) consisting of theoretical and practical aspects. This paper presents a general framework for the SCCA (called an Iterative SCCA (ISCCA)) on block ciphers in which these aspects are explained and the requirements are listed. On the theoretical side, we use extracting quadratic equations, recognizing iterated chosen plaintexts, and cube iteration to improve the SCCA on block ciphers. On the experimental side, we define a feasible scenario in which ISCCA can be applied on block ciphers. Then, we implement the ISCCA on AES and verify the results on an ARM microcontroller. Finally, we compare the proposed SCCA (ISCCA) with the Simple Power Analysis, the previous SCCAs, and the previous attacks on AES. This comparison is based on the template building and data, time, and memory complexity. We show that the SCCA can recover 128 and 256 key bits of the AES-128/256 only with data complexity $2^{7.3}$, time complexity $2^{15.74}$, and memory complexity $2^{7.89}$ on AES-128, and data complexity $2^{7.75}$, time complexity $2^{16.2}$, and memory complexity $2^{8.21}$ on AES-256. We show only nine interesting points are needed for template matching phase. This is the most efficient SCCA on AES-128/256.

Keywords Side Channel Cube Attack · AES-128/256 · Hamming Weight · ARM microcontroller

E. Aghaee
Faculty of Technical and Engineering, University of Ardabil, Ardabil, Iran
E-mail: erfanaghaei69@gmail.com

1 Introduction

1.1 Motivation

The SCCA is a kind of Algebraic Side Channel Attacks (ASCA). It is a combination of side channel [20] and cube attack [24]. In the cube attack, the goal is to distinguish monomials in which just a single or one key bit exists. In other words, it detects linear monomials based on key bits in ANF representation. After finding linear monomials, the cube attack will be able to recover key bits by solving these linear equations.

In practical, however, because the degree of ANF representation is very high for a cipher bit, the cube attack usually is applied to the simplified version of ciphers. In this situation, the side channel attack can be used for acquiring the leakage information in intermediate states, where the degree of ANF representation is low. The cube attack can utilize the side channel attack to reduce the degree of ANF representation, which the merging of these two types of attacks result in Side Channel Cube Attack (SCCA) that introduced by Dinur and Shamir [1] in 2009. They applied the SCCA on single bit at the end of round 3 of Serpent (requiring about 2^{18} times for full key recovery) and round 2 of AES (requiring about 2^{35} times for full key recovery). Their results show that the SCCA requires significant amount of memory and chosen plaintext to recover all key bits of AES. As a result, it seems that the SCCA is not an efficient attack on AES. After that, the researches on the SCCA target light-weight block ciphers such as PRESENT [2–7] , NOEKEON [8], KATAN [9], Hummingbird-2 [10], and LBlock [11, 12], in which the cipher algorithm have simple algebraic structure and show that the SCCA is an efficient attack on light-weight block ciphers. However, there is no an obvious scenario that shows to the attacker in which scenario is able to apply the SCCA on block ciphers. They also did not do a comprehensive comparison the SCCA with other attacks on block ciphers in respect of data, time, and memory complexity till the adversary know that how much the SCCA is efficient on block ciphers. Moreover, it is not clear effect of the key length on the complexity of the SCCA.

On the other hand, information of intermediate states which are obtained by side channel attack are quiet noisy. It means that the success rate of the SCCA decreased. Furthermore, the most of previous the SCCAs on block ciphers are based on theoretical aspects and simulation, except in Zaho [4] that they consider the experimental side of the SCCA and propose a method to increase HW deductions accuracy by defining a new correlation factor, while our investigation show that their proposed correlation factor is not a general and accurate HW deduction for different target devices. Therefore, the necessity of practical implementation of the SCCA on different cryptosystems is felt.

To improve and make the SCCA more efficient on block ciphers, we consider AES-128 as a target cipher algorithm. Then, instead of applying the SCCA on single bit of round 2 of AES (Dinur and Shamir [1]), we apply the ISCCA on output byte of each S-box at the first round of AES and extract quadratic beside the linear polynomials of secret variables, and we also use a method to recognize iterated chosen plaintexts. By preventing iterated chosen plaintexts and considering S-box output in first round, we decrease complexity of the SCCA and improve random search space to local search space of an S-box. We also recover more key bits by extracting quadratic equations. After that, to show the effect of key length on the complexity of SCCA, we consider AES-256 as the target cipher algorithm. We show that as the length of key increase, the complexity of the SCCA increase as well. An over increased key length makes the SCCA impractical on block ciphers. Therefore, to reduce effect of key length on the complexity of the SCCA, we proposed cube iteration that minimizes relation of key length and complexity.

In addition, as it mentioned above the most important challenge of the SCCA is the online phase, where the SCCA use leakage information. However, the majority of researches are theoretical, and there is no clear physical experiment which shows practical scenario to apply the SCCA on block ciphers in physical experiment. Therefore, we define feasible scenario which the proposed SCCA (ISCCA) can be applied to cryptosystems. Then, we describe the importance of finding interesting points in leakage information and show that for HW deduction, acquiring of Signal-to-Noise (SNR) target devices, is necessary. The SNR is calculated with iteration and average of power traces for same inputs, and we show that Changing in correlation factor is not a general way.

This way, we present a general framework for the SCCA on block ciphers and show that it can be implemented efficiently on AES. We compare the proposed SCCA (ISCCA) with the previous attacks on AES in respect to data, time, and memory complexity, in which data complexity is the amount of data (plaintext and ciphertext) that is required to execute the attack, time complexity is the number of en-/decryption that is needed to perform the attack and memory complexity is the amount of memory that is needed to hold all data during the attack. Finally, we illustrate existing challenges against the SCCA in black box scenario.

1.2 Our Contribution

The contribution of this paper is fivefold:

1. To improve the SCCA, we proposed some techniques such as recognizing iterated plaintexts, extract quadratic equations in first round and cube iteration and called the improved SCCA as Iterative SCCA (ISCCA).
2. We show the ISCCA is applicable on AES in practical scenario with trivial complexity.
3. We compare the ISCCA with previous attacks on AES in respect to number of HW deductions, data, time, and memory complexity.

4. We define a practical scenario, in which the ISCCA is able to be applied on AES and we show the necessity of finding intersting points to increase HW deduction accuracy.
5. We illustrate existing challenges against the ISCCA in black-box scenario.

1.3 Structure of Paper

This paper aims to improve an practical SCCA by using iteration and defining practical scenario. In section 2 gives prelimineries about the cube attack and SCCA. The related work attacks on AES-128/256 is given in section 3. In section 4, we mention to review the proposed steps for improving the SCCA. In section 5, we describe our proposed techniques to imrpove SCCA in theoretical and practical aspects. The practical results is given in section 6. Eventually, Section 7 gives conclusion.

2 Preliminaries

2.1 Cube Attack

The main point of the cube attack is that any output bit of the cryptosystem (ciphertext) can described by multivariate polynomial $F(p_1, .., p_m, k_1, .., k_n)$ which it contains m public variables and n secret variables. In fact, the cube attack provides a method that extracts linear equations of key. The cube attack divided into two basic phases. First, the preprocessing phase is for extracting linear equations of key. Second, the online phase acquires the value of linear equations of key, extracted in preprocessing phase, and it solves these equations by different techniques such as Gaussian elimination. Consider a block cipher and its ciphertext bits function $(c_1, .., c_m) = F_i(p_1, .., p_m, k_1, .., k_n)$, where c_i, k_j, and p_m are ciphertext bit, key, and plaintext respectively. An ciphertext bit can described by a multivariate polynomial in the plaintext and key bits $c_i = F_i(p_1, .., p_m, k_1, .., k_n)$. Then F can presented as equation (1).

$$F_i(p_1, .., p_m, k_1, .., k_n) \equiv t_I \times F_{S(I)} + G(p_1, .., p_m, k_1, .., k_n) \tag{1}$$

The subset $I \subseteq \{1, ..., m\}$ is indexes of all public variables called cube. $t_I = \prod_{i \in I} p_i$ Produce of all public variable included in I. $F_{S(I)}$ is superpoly of I and $G(p_1, .., p_m, k_1, .., k_n)$ is all monomials that are not dividible by t_I.

If $Deg(F_{S(I)}) = 1$ then t_I is a maxterm of F. For example, consider a polynomial $F(p_1, p_2, p_3, k_1, k_2, k_3)$:

$c_0 = F(p_1, p_2, p_3, k_1, k_2, k_3) = p_1p_2p_3 + p_1p_2k_1 + p_1p_3k_1 + p_2p_3k_1 + p_1p_2k_3 + p_1p_3k_2 + p_1p_2 + 1$

the degree of the F is three. we choose a subset I with size of d-1=2. therefore, $I = \{1, 2\}$, $t_I = p_1p_2$, $F_{S(I)} = k_1 + k_3 + 1$, $G(p_1, p_2, p_3, k_1, k_2, k_3) = p_1p_2p_3 + p_1p_3k_1 + p_2p_3k_1 + p_1p_3k_2 + 1$. if we sum F over t_I , there are four assignment to $p_1$and p_2, and we assume all public variables that are not in t_I are zero. With the sum of all assignments we will have $F_I = F_{S(I)}$. If we choose other I's with size of d-1, we can create a system of key equations. We are able to recover key bits by solving the system of key equations.

2.2 Side Channel Cube Attack

As it mentioned above, because the degree of ANF representation ($F(p_1, p_2, p_3, k_1, k_2, k_3)$) is very high for a cipher bit in a block cipher, the cube attack is applied to intermediate state bits. The cube attack can utilize the side channel attack to aqcuire leakage information in intermediate state, which merging of these two types of attacks result in Side channel Cube Attack.

In the SCCA, we can describe an intermediate state bit by $i_i = F(p_1, p_2, p_3, k_1, k_2, k_3)$. We can acquire superpolys either based on one bit in intermediate states or based on Hamming weight of bits in intermediate states or in generally way according to considered power model. Researches show that extract superpolys based on Hamming weight of bits in intermediate states is more accurate and simpler. The Hamming weight of one byte in intermediate state can be described as equation 2, in which x_i is a bit in intermediate state.

$$HW = \sum_{i=0}^{7} x_i \tag{2}$$

In the preprocessing phase, The SCCA extracts superpolys based on the Hamming weight of bytes in intermediate states. After that, in online phase, the power consumption of target device while is running a cipher algorithm is measured. Then, by using the correlation between HW power model and power trace, we can deduce HW of intemediate states.

3 Related Works

As we mentioned, the SCCA is a combinational attack. Therefore, it is comparable in different aspects with other attacks on AES. On the one hand, on the basis of the kerchkhoffs assumption [13], it is chosen plaintext attack, in which an adversary is able to choose plaintext and obtained corresponding ciphertext. Therefore, three complexities are important, namely time, data, and memory. On the other hand, the online phase of SCCA is based on template attacks. Accordingly, it should be compared with other template attacks in respect to number of Hamming weights, building template, and matching template strategy.

There are several cryptanalysis for reduced-round AES. In [14], the square attack for six rounds of AES is shown, and the collision attack on 7 rounds of AES is given in [15]. In [16], impossible deferential of 5-round AES is presented. Based on deferential cryptanalysis, the boomerang for six rounds of AES is given in [17]. In [18], the related key is shown for seven rounds of AES, and the algebraic attack is presented in [19]. All the algorithms that mentioned are known as shortcut attacks. In other words, those attacks have the complexity less than that of brute-force. The cryptanalysis attacks are compared based on data, time, and memory complexity. All the shortcut algorithms that mentioned above have data complexity about 2^{32} , various time complexities in range about 2^{31} through 2^{224} , and memory complexity about 2^{32} on AES-128/256. Regarding to abovementioned shortcut algorithms that can be applied to AES reduced to seven or eight rounds, it seems that AES is immune against these attacks.

The prominent attack on AES based on Template attack is Simple Power Analysis (SPA) proposed by Mangard in [20], in which 81 Hamming weight of the attacked round key are extracted, and after extracting HW, the search space reduced to $2^{28.26}$ for recovering all key bits. After that, first combinational attack on AES is Algebraic Side Channel Attack (ASCA) that is presented In [21].The ASCA is an attack that is combined with side channel attacks and utilized leakage information. The ASCA included two offline and online phases. In the offline phase, system of AES equation was built, it results in a system of approximately 1800 equations in 10 000 variables (27000 monomials) that occupies significant memory. On the other hand, solving the system of equations which represent the AES is generally too hard.

Dinur and Shamir describe the SCCA on AES with time complexity about 2^{35} . They also show a new model of SCCA that could tolerate against 11% of noise of the leaked bits in practical scenario by using erasure code [22], but the success of this model is based on the assumption that the attacker knows which measurement is correct and which one is not. Since the initial version of SCCA, some block ciphers have been analyzed such as PRESENT, NOEKEON, KATAN, Hummingbird-2, and LBlock. However, the majority of investigations is on the PRESENT because the PRESENT has a lowest implementation cost and has simplest algebraic function comparing to other ultra-light weight block ciphers. These researches on SCCA are vague in respect to amount of efficiency on block ciphers like AES and building template. In [1] claims that the SCCA can work under black-box settings where neither ASCA nor DPA [23] can, while it does not describe the SCCA challenges against black box scenario. In this paper, we try to eliminate these ambiguities and we list the challenges against black-box scenario.

4 Outline of the Attack on AES-128/256

First and possibly the most important aspect of a cryptography attack is that the adversary needs to know in which scenario he/she is able to apply the ISCCA on a cryptosystem. First of all, therefore, we define a feasible attack scenario, and we list which requirements need to be fulfilled. After defining attack scenario, we apply the proposed SCCA (ISCCA) on AES and extract linear and quadratic equations from first round Hamming weight of S-box output. After extracting linear and quadratic equations, we build template for output Hamming weight of S-box. According to pipeline structure of ARM microcontroller, and replacement of instructions in this structure, finding related points of measured power is a challenge. Therefore, we calculate the correlation between measured power and the power model. Peaks of the correlation show our interesting points. Next, we generate the plaintexts regarding to the extracted cubes and deduce the HW with correlation between template and power trace, where key is secret. After finding first 128 key bits as abovementioned. We substitute key variables with the key recovered bits in AES-256 and repeat the SCCA on AES S-box in round 2. This way, we decrease the effect of key length on the SCCA's complexity.

5 Practical Iterative Side Channel Cube Attack (ISCCA)

As mentioned in section 2, The SCCA consisted of two basic phases: preprocessing and online phase. The preprocessing phase aims to extract superpolys. It divided into three sub phases such as extracts degree of F (a function in intermediate state), linearity test, and extract superpolys respectively. The online phase deduces the value of equations that extracted in preprocessing phase. In below, we try to explain the ISCCA method and show the importance of iteration to reduce attack's complexity. We also describe required steps to HW deduction in a physical experiment.

5.1 Attack Scenario

Some requirements are needed to be fulfilled till we could able to apply the ISCCA on the target cipher algorithm practically. Because the ISCCA is a template attack, the requirements are same the Mangard's SPA [20]. In the below, to apply the ISCCA on AES, all requirements are listed. It is necessary to mention that following requirements considered in white box scenario, in which the attacker has information about target cipher algorithm.

1. We should able to characterize device under attack. It means that we can create templates on the device.
2. The attacker can measure the power consumption of the device when it is running AES encryption with different plaintexts (cube) and the same key.
3. The attacker can assign each intermediate state to a part of power traces.

4. One intermediate state should be a function of the plaintext and a low degree of key bits.

After expressing the requirements of ISCCA for applying it to AES, we denote a bit of intermediate state as $I^{\{Round\},\{Bit\_Index\}}_{\{Target\_Byte\},\{Target\_Operand\}}$. For example, $I^{3,1}_{2,S\_box}$ expresses that the target bit is first bit on second byte of S-box in third round.

5.2 Preprocessing phase

In order to improve the preprocessing phase of the SCCA, we divide the preprocessing phase to four subphases such that it incorporates identify the target state, linearity and quadratic test, extract superpolys, and recognizing iterated superpolys. The preprocessing phase of ISCCA is based on the following methods.

5.2.1 Identify the target state

The target intermediate location plays an important role in success rate of the SCCA. An ideal intermediate state is places that small changes in the input produce big changes in output. On the other hand, the polynomial degree of intermediate state function should be as small as possible. Because the degree of intermediate states increases exponentially with the rounds, we consider output byte of each S-box as target byte of intermediate state, in which the highest degree for each 8-bit input 8-bit output S-box is seven. Therefore, we consider first byte of S-box in first round and we denote it as $I_{byte} = \sum_{j=0}^{7} I^{1,j}_{1,S\_box}$.

5.2.2 Linearity and Quadratic Test

We acquired that the polynomial degree of target byte (I_{byte}) is $d = 7$. The goal is finding expressions of the t_I such that extracts linear and quadratic superpolys. To do so, Steps of linearity test are listed:

1. Consider a t_I that it contains $d - 1$. plaintext variables. We need 2^{d-1} queries for each t_I.
2. We set zero to remain $(m - (d - 1))$ plaintext variables.
3. Next consider two random secret variables(k', k'').
4. We calculate the equation of $p(0) + p(k') + p(k'') + p(k' + k'') = 0$ for more 100 different couple random keys. If the equations for 100 different couple random keys equal to zero, we can recognize with high probability this t_I have linear equation of secret key.

The procedure for finding expressions of the t_I such that extracts linear superpolys is shown in Algorithm 1. As depicted and shown in Algorithm 1, Query(k, I) is function of all possible plaintext queries according to I array.

Therefore, we do 2^{d-1} queries from AES for each random k. For example, assume d = 3 and $I = \{1, 3\}$ the number of queries that for each random k is required are listed in Table 1.

Algorithm 1 Linearity Test

```
1: procedure LinearityTest(I, LT)
2:     $R_0 \leftarrow Query(0, I)$                ▷ Query from system in respect to k = 0
3: loop:
4:     $k' \leftarrow Random()$
5:     $k'' \leftarrow Random()$
6:     if $i < 100$ then
7:         $R\_k' \leftarrow Query(k', I)$
8:         $R\_k'' \leftarrow Query(k'', I)$
9:         $k'\_XOR\_k'' \leftarrow XOR(k', k'')$
10:        $R\_k'\_XOR\_k'' \leftarrow Query(k'\_XOR\_k'', I)$
11:        $result \leftarrow XOR(R_0, R\_k', R\_k'', R\_k'\_XOR\_k'')$
12:        $i \leftarrow i + 1.$
13:        $HW\_result \leftarrow Hamming\_weight(result)$
14:        if $HW\_result == 1$ then return 0
15:        goto loop.
    return 1
```

Table 1 Random k queries(I=1,3)

$$\boxed{\begin{array}{l}\sum_{j=0}^{7} I^{1,j}_{0,S-Box}(0,0,0,...,0,k') = \{0/1\}\\ \sum_{j=0}^{7} I^{1,j}_{0,S-Box}(1,0,0,...,0,k') = \{0/1\}\\ \sum_{j=0}^{7} I^{1,j}_{0,S-Box}(0,0,1,...,0,k') = \{0/1\}\\ \sum_{j=0}^{7} I^{1,j}_{0,S-Box}(1,0,1,...,0,k') = \{0/1\}\end{array}}$$

Then, in order to finding expressions of the t_I such that extracts quadratic superpolys, we consider four following steps:

1. Consider a t_I that it contains $d - 2$. plaintext variables. We need 2^{d-2} queries for each t_I.
2. We set zero to remain $(m - (d - 2))$ plaintext variables.
3. Next consider two random secret variables(k', k'', k''').
4. We calculate the equation of $p(0) + p(k') + p(k'') + p(k''') + p(k' + k'') + p(k' + k''') + p(k'' + k''') + p(k' + k'' + k''') = 0$ for more 100 different random key pairs. If the equations for 100 different random key pairs equal to zero, we can recognize with high probability this t_I have quadratic equation of secret key.

The procedure for finding expressions of the t_I such that extracts quadratic superpolys is shown by in Algorithm 2.

Algorithm 2 Quadratic Test

```
1: procedure QuadraticTest(I, QT)
2:     $R_0 \leftarrow Query(0, I)$
3: loop:
4:     $k' \leftarrow Random()$
5:     $k'' \leftarrow Random()$
6:     $k''' \leftarrow Random()$
7:     if $i <$ 100 then
8:         $R\_k' \leftarrow Query(k', I)$
9:         $R\_k'' \leftarrow Query(k'', I)$
10:        $R\_k''' \leftarrow Query(k''', I)$
11:        $k'\_XOR\_k'' \leftarrow XOR(k', k'')$
12:        $k'\_XOR\_k''' \leftarrow XOR(k', k''')$
13:        $k''\_XOR\_k''' \leftarrow XOR(k'', k''')$
14:        $k'\_XOR\_k''\_XOR\_k''' \leftarrow XOR(k', k'', k''')$
15:        $R\_k'\_XOR\_k'' \leftarrow Query(k'\_XOR\_k'', I)$
16:        $R\_k'\_XOR\_k''' \leftarrow Query(k'\_XOR\_k''', I)$
17:        $R\_k''\_XOR\_k''' \leftarrow Query(k''\_XOR\_k''', I)$
18:        $result \leftarrow XOR(R_0, R\_k', R\_k'', R\_k''', R\_k'\_XOR\_k'', R\_k'\_XOR\_k''',$
               $R\_k''\_XOR\_k''', R\_k'\_XOR\_k''\_XOR\_k''')$
19:        $i \leftarrow i + 1.$
20:        $HW\_result \leftarrow Hamming\_weight(result)$
21:        if $HW\_result ==$ 1 then return 0
22:        goto loop.
       return 1
```

5.2.3 Extract Superpolys

The goal is extracting the related t_I. The linear superpolys can present as equation 3 and quadratic superpolys can present as equation 4. The previous subphases, Linear and Quadratic Test, helps us to find Is , which extracts linear and quadratic superpolys. In table 2, $p \in C_I$ define all possible plaintexts values according to variables in I. For example, we consider an intermediate state, in which the function of a bit in intermediate state has three public and secret variables ($I^{1,j}_{0,S-Box}(p_1, p_2, p_3, k_1, k_2, k_3)$). In linear and quadratic test subphases, we find $I = \{1, 2\}$ for extracting linear superpolys and $I = \{1, 3\}$ for extracting quadratic one. Because m is constant, it depends on k=0 and because $a_i s$ are linear factors, they depend on i-th key bit and k=0. Therefore, for extracting constant value of equation 3, we assing zero to secret variables (k_1, k_2, k_3), and for extarcting i-th key bit, we assign one to i-th key bit. The number of queries that are needed for extracting linear superpolys are listed in table 2. Because $a_{ij} s$ are quadratic factors, they depend on i-th key bit, j-th key bit and k=0. Therefore, for extracting quadratic superpolys, we assign one to i-th and j-th key bit. The number of quries that are needed for extracting quadratic equations are listed in table 3.

$$\begin{gathered} F_{S(I)} = m + \sum a_i k_i \ modulo\ 2 \\ m, a_i \in \{0, 1\} \end{gathered} \tag{3}$$

$$\begin{gathered} F_{S(I)} = m + \sum a_i k_i + \sum a_{ij} k_i . k_j \ modulo\ 2 \\ 0 \le i < j \le n - 1 \end{gathered} \tag{4}$$

Table 2 Queries that are needed for extracting linear superpolys

Extracting linear superpolys ($\sum_{j=0}^{7} \sum_{p \in C_I} I_{0,S-Box}^{1,j}(p_1, p_2, p_3, k_1, k_2, k_3)$)
Number of queries for each $I_{0,S-Box}^{1,j}(P, K)$ (e.g. I={1,2})
$\left.\begin{array}{l} \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,0,0,0,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,1,0,0,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,0,0) \end{array}\right\} \bigoplus \rightarrow \{0/1\} = m$
$\left.\begin{array}{l} \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,0,0,1,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,1,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,1,0,1,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,1,0,0) \end{array}\right\} \bigoplus \rightarrow \{0/1\} \bigoplus m = a_0$
$\left.\begin{array}{l} \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,0,0,0,1,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,1,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,1,0,0,1,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,1,0) \end{array}\right\} \bigoplus \rightarrow \{0/1\} \bigoplus m = a_1$
$\left.\begin{array}{l} \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,0,0,0,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,1,0,0,0,0) \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,0,0) \end{array}\right\} \bigoplus \rightarrow \{0/1\} \bigoplus m = a_2$

5.2.4 Recognizing iterated chosen plaintexts

In extracted linear and quadratic equations a lot of chosen plaintexts are iterated so that recognizing and eliminating them reduce the SCCA data complexity significantly. For example, consider a polynomial $F(p_1, p_2, p_3, k_1, k_2, k_3)$ of degree 3:

$$i_0 = F(p_1, p_2, p_3, k_1, k_2, k_3) = p_1p_2p_3 + p_1p_2k_1 + p_1p_3k_1 + p_2p_3k_1 + p_1p_2k_3 + p_1p_3k_2 + p_1p_2 + 1$$

We consider $I_1 = \{1, 2\}$ then $F_{S(I)} = k_1 + k_3 + 1$. If we consider $I_2 = \{1, 3\}$ then $F_{S(I)} = k_1 + k_2$. So, we should query four chosen plaintext from system to acquire the amount of two extracted equations. All queries that are needed for acquiring the amount of extracted superpolys are listed in table 4. We can saw two first chosen plaintext are iterated. To reduce the ISCCA data complexity we can prevent from this iteration.

5.3 Online phase

The online phase of the ISCCA exploits that the power consumption depends on data that is processed. In this phase, we assume that the attacker possess the same device of the device under attack that he has fully control on it. He

Table 3 Queries that are needed for extracting quadratic superpolys

Extracting quadratic superpolys ($\sum_{j=0}^{7}\sum_{p\in C_I} I_{0,S-Box}^{1,j}(p_1,p_2,p_3,k_1,k_2,k_3)$)	
Number of queries for each $I_{0,S-Box}^{1,j}(P,K)$ (e.g. I={1,3})	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,0,0,1,1,0)$	$\bigoplus \to \{0/1\} \bigoplus m \bigoplus a_0 \bigoplus a_1 = a_0.a_1$
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,1,1,0)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,1,0,1,1,0)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,1,1,0)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,0,0,1,0,1)$	$\bigoplus \to \{0/1\} \bigoplus m \bigoplus a_0 \bigoplus a_2 = a_0.a_2$
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,1,0,1)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,1,0,1,0,1)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,1,0,1)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,0,0,0,1,1)$	$\bigoplus \to \{0/1\} \bigoplus m \bigoplus a_0 \bigoplus a_1 = a_1.a_2$
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,1,1)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(0,1,0,0,1,1)$	
$\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(1,0,0,0,1,1)$	

is able to measure the power traces for different input bits. We consider the mean of iteration for same inputs as template.

5.3.1 Building Template Phase

In order to build template for AES S-box, we execute instruction table 5 for 5000 random pairs of plaintext and keys, in which each input 500 times is iterated, and the power consumption mean of each iteration is recorded. We acquire number of iteration for each input regarding to Signal-to-Noise (SNR) AT91SAM256 ARM microcontroller in experimental way. Then, we gather the traces regarding to the same Hamming weight of the output S-box and estimate the mean vector for each input. After that, we use Correlation to determine the interesting points. The number of interesting points are denoted by N_{IP} and show only nine interesting points are enough to building template. Figure 1 show correlation on AES for finding Interesting points for first four byte. In figure of 1, we have three peak of correlation that we consider these three maxterms as our interesting points. These interesting points show a part of power trace that related to output first byte of S-box in first round.With these

Table 4 Chosen plaintext iteration while k is secret.

$I_1 = \{1,2\}$	$I_1 = \{1,3\}$	Iterated Chosen plaintext
$F(0,0,0,k)$	$F(0,0,0,k)$	Iterated
$F(1,0,0,k)$	$F(1,0,0,k)$	Iterated
$F(0,1,0,k)$	$F(0,0,1,k)$	Not-Iterated
$F(1,1,0,k)$	$F(1,0,1,k)$	Not-Iterated

nine interesting points we can deduce the Hamming weight of out S-box with high accuracy.

5.3.2 Matching Template Phase

A method that we use to measure a linear relationship between two value is the correlation coefficient $\rho(X, Y)$. The correlation coefficient is defined in 5. It can only take values between plus and minus one: $-1 \leq \rho \leq +1$

$$r = \frac{\sum_{i=1}^{N_{IP}} (x_i - \bar{x_i})(y_i - \bar{y_i})}{\sqrt{\sum (x_i - \bar{x_i})^2) \sum (y_i - \bar{y_i})^2)}} \tag{5}$$

5.3.3 Cube Iteration

After finding 128 key bits in first round we can replace all 128 bits that are recoverd in first round with related key bit variables. Then, we repeat the ISCCA on AES S-box in round 2. In this way, the effect of key length on ISCCA's complexity reduced significantly.

6 Practical Results

The preprocessing phase of ISCCA was first implemented in Visual C++ 5.0 on a 3.10 GHz Core-i3 PC. In the preprocessing phase of the ISCCA, we find seven linearity superpolys in 0.468 seconds and one quadratic superpolys in 1.197 seconds for each AES S-box. This suffices to recover all the 128 key bits in each round. According to degree of each S-box, we consider cube of dimension 6 for linear superpoly and dimension 5 for quadratic superpoly, which are listed in table 6. In AES-256, after finding 128 key bits in first round, we substitute key variables to recovered key bits and run again preprocessing phase and find relation between inputs S-box in second round with plaintext input, which this relationship is listed in table 7. According to cube iteration and substitution key variables to key recovered bits, we hold cube dimension 6 and 5 for linear and quadratic superpolys respectively in second round.

Table 5 Implemented Code

```
PIO_Set_TRIG
$for(i = 0; i < 16; i++)$
    $state[i] = Plaintext[i] \wedge key[i];$
$for(i = 0; i < 16; i++)$
    $state[i] = S - Box[state[i]];$
PIO_Clear_TRIG
```

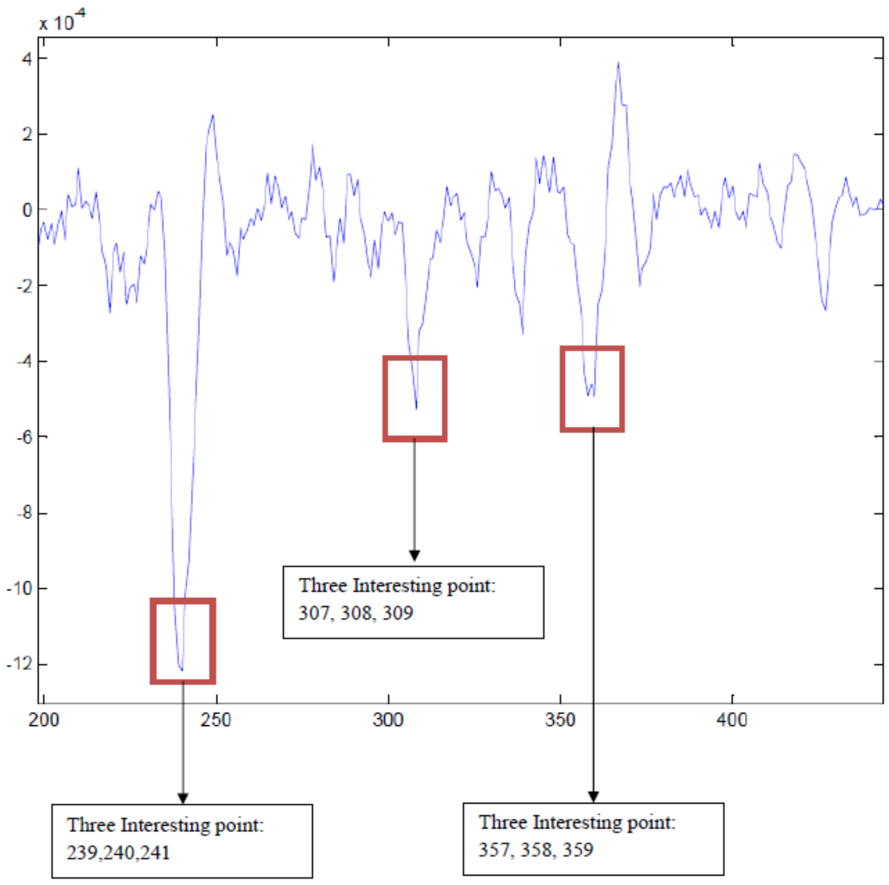

Fig. 1 Location of Interesting points

Table 6 Extracted linear and quadratic superpolys for each S-box

Cube Index	Superpoly for each S-box
$\{0,1,2,3,4,5\}$	$1 + k_6 + k_7$
$\{0,1,2,3,4,6\}$	k_5
$\{0,1,2,3,5,7\}$	$1 + k_4$
$\{0,1,2,3,5,6\}$	k_3
$\{0,1,3,4,5,6\}$	$1 + k_2 + k_7$
$\{0,2,3,4,5,6\}$	$k_1 + k_7$
$\{1,2,3,4,5,6\}$	$1 + k_0 + k_7$
$\{0,1,2,3,5\}$	$k_4 + k_7 + k_4k_6 + k_4k_7$

Table 7 relation between inputs S-box in second round with plaintext input

Input Plaintext	Input Byte S-box in Second Round
$\{0, 5, 10, 15\}$	$\{0, 1, 2, 3\}$
$\{3, 4, 9, 14\}$	$\{4, 5, 6, 7\}$
$\{2, 7, 8, 13\}$	$\{8, 9, 10, 11\}$
$\{1, 6, 11, 12\}$	$\{12, 13, 14, 15\}$

The online phase of ISCCA was implemented on AT91SAM7S256 ARM microcontroller that runs at 48 MHz. A digital Oscilloscope(with 150 M sample/s) measures the voltage changes across a resistor connected to VDDCORE pin of AT91SAM7S256. We send data from a PC to ARM microcontroller by a USB interface. During the online phase of the SCCA, to build the template, we send 5000 random pairs of plaintext and keys. After that, the power traces are recorded while the device under attack is running each pair of inputs. We divide all power traces into 9 HW groups regarding to HW of the output byte of S-box in first round. Our investigations show that only 9 interesting point for each S-box is enough for building template. We find these interesting points by using methods that described in section 6.2.1. As depicted in figure 2, the interesting points for first byte are 240,308, and 358, for second byte are 363,432, and 482, for third byte are 478, 548,598, and for forth byte are 599, 647, and 664. We consider 2 samples around of each peak of correlation as the interesting points.

In figure 3, the correlation and number of iteration for Hamming weight four is given. Accordingly, if we want to increase probability success of the online phase of ISCCA, we should increase number of iteration of each plaintext according to extracted cubes in table 6. As depicted in figure 3, with 350 iteration we can deduce Hamming weight with 10^{-4} accuracy.

In order to find minimum interesting points to extract correct Hamming weight, we consider 3, 9, 15, and 30 interesting points as the templates. We calculate maximum and average iteration for each scenario, which is shown in figure 4. We show that only nine interesting points are suitable for extracting Hamming weights.

In the online phase, we need to extract 156 different HW for each S-box. Therefore, the data complexity of the ISCCA is (156) $2^{7.3}$ for extracting 128 key bits and (156 + 60) $2^{7.75}$ for extracting 256 key bits AES-128 and 256 respectively. The time spending for each plaintext query with 350 iterations is about 20 second that deduces all HWs about one hour in our physical scenario.

The time complexity of ISCCA is (156*350) $2^{15.74}$ for recovering 128 key bits and $2^{16.2}$ for recovering 256 key bits AES-128 and 256 respectively. According to 9 interesting points for the template, the memory complexity has reduced significantly, in which is equal (156+9*9) $2^{7.89}$ for AES-128 and $2^{8.21}$ for AES-256.

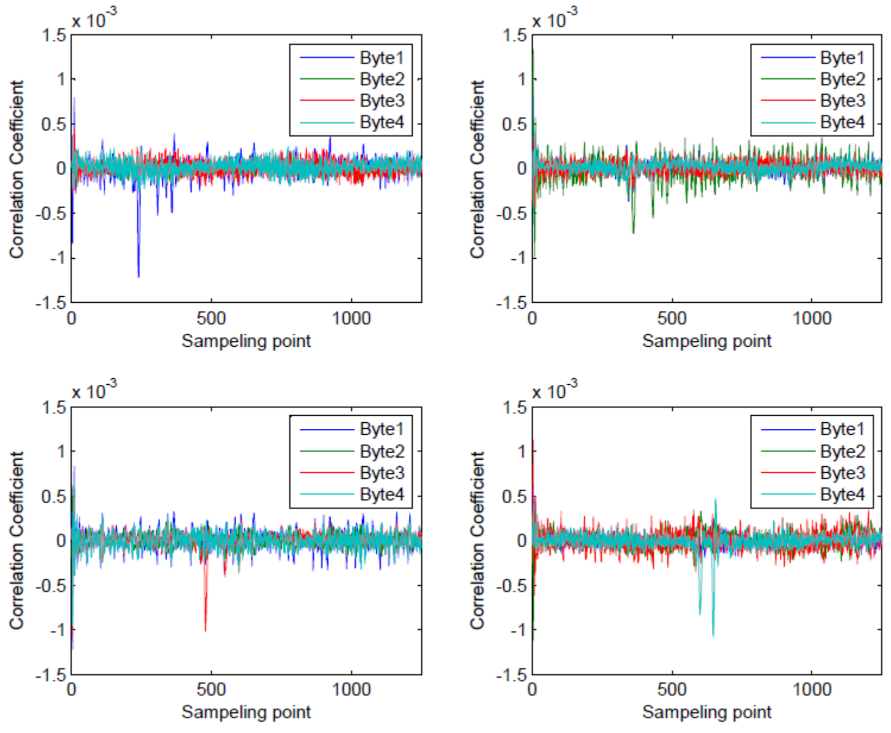

Fig. 2 Peaks shows the Location of interesting points for four first S-box.

7 Discussion

We compare below the proposed ISCCA with other attacks on AES-128/256 in respect complexity, number of HW deduction, and countermeasures against leakage information.

7.1 Comparison of complexity

To improve and reduce data complexity of the SCCA on AES-128/256, instead of applying the SCCA on single bit in round 2 of AES (Dinur and Shamir [1]), we apply the proposed ISCCA on each output byte of S-box in first round of AES. This way, we decrease the degree of each ANF from 2^{14} to 2^{7}. After that, our investigation shows that if we just extract linear equations in first round, we were able to recover only 48 key bits. While with extracting quadratic key bits, we can recover 128 key bits in each round. To reduce data complexity, we also prevent from common chosen plaintexts so that 416 chosen plaintexts reduced to 156 different chosen plaintexts for recovering 128 key bits in each round. In other words sixty-eight percent of chosen plaintext in the early SCCA is iterated. It means that eliminating unnecessary queries

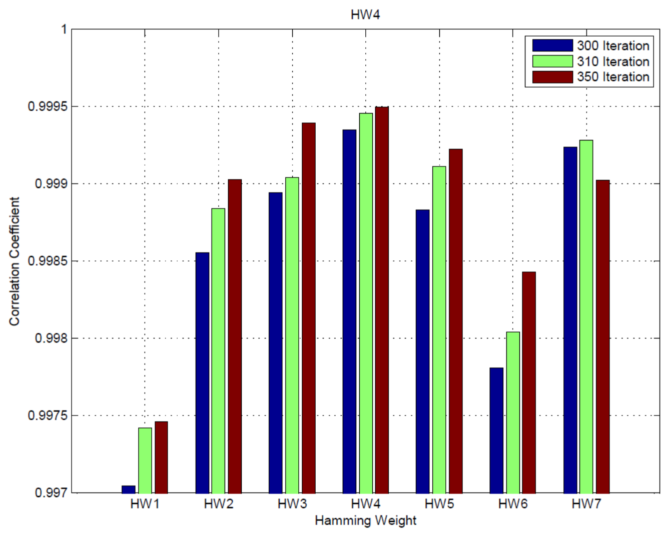

Fig. 3 Deduce Hamming weight four by using correlation (HW of intemediate state equals four).

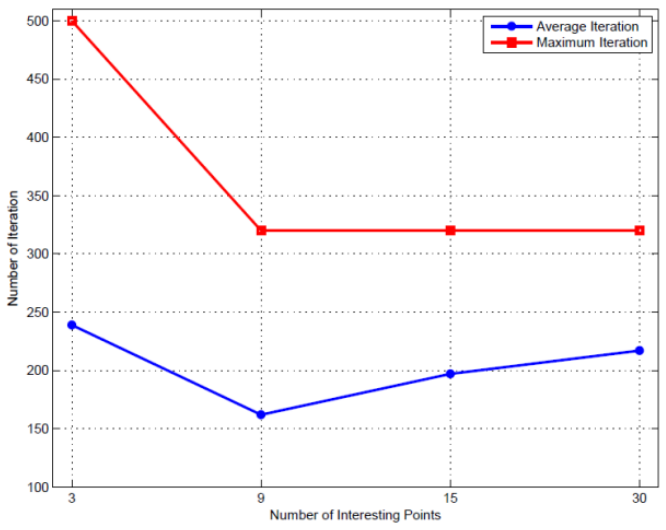

Fig. 4 Number of Iteration and Intersting points.

can reduce substantial amount of data complexity. The comparison between the proposed SCCA (ISCCA) and previous attacks on AES in respect to data complexity is shown in table 8.

According to abovementioned techniques for reducing data complexity, time complexity reduced as well. The comparison between the proposed SCCA (ISCCA) and previous attacks on AES in respect to time complexity is shown in table 9.

Table 8 The comparison between SCCA and other algorithms on AES in respect to data complexity.

Attack	Key Size	Number of rounds	Data Complexity
Square	All	6	2^{32}
Collision	256	7	2^{32}
Impossible differential	128	5	$2^{29.5}$
Impossible differential	256	7	$2^{92.5}$
Boomerang	128	6	2^{71}
SPA	128	FULL	$2^{28.26}$
ASCA	128	FULL	1
SCCA	128	FULL	2^{28}
ISCCA	128	FULL	$2^{7.3}$
ISCCA	256	FULL	$2^{7.75}$

In addition, we show the importance of finding number of interesting points to reduce memory complexity and increase HW deduction accuracy. In experimental way, we find that only nine interesting points are enough to deduce the HW. The comparison between the proposed SCCA (ISCCA) and previous attacks on AES in respect to memory complexity is shown in table 10.

According to the building template phase of the proposed ISCCA, we can compare the ISCCA with Mangards SPA [20] in respect to number of HW deduction. Mangards SPA targets key scheduling in AES and show with 81 HW deduction, he can recover 128 key bits while this is 156 HW deduction for the ISCCA. However, what make the ISCCA superior to Mangards SPA is that in Mangards SPA after acquiring 81 HW should apply the brute force attack to 2^{28} all possible search space state of key. While, in the ISCCA, we will face 128 simple linear and quadratic equations of key that solving of them is very simple. As a result, if we could extract HW of intermediate states with high accuracy, the ISCCA is more efficient attack than Mangards SPA on AES.

7.2 Countermeasures

There are several techniques such as masking for preventing leakage of side channel information. Our investigation shows that the ISCCA just can be

Table 9 The comparison between SCCA and other algorithms on AES in respect to time complexity.

Attack	Key Size	Number of rounds	time complexity
Square	All	6	2^{72}
Collision	256	7	2^{192}
Impossible differential	128	5	2^{31}
Impossible differential	256	7	$2^{250.5}$
Boomerang	128	6	2^{71}
SPA	128	FULL	$2^{28.26}$
ASCA	128	FULL	NP-Complete
SCCA	128	FULL	2^{32}
ISCCA	128	FULL	$2^{15.74}$
ISCCA	256	FULL	$2^{16.2}$

applied in which the target algorithm use a constant mask for each round. Let consider a constant mask, namely, u. u injects random data to output of each S-box and prevents from leaking of side channel information. In this situation instead of targeting S-box output in one round, the ISCCA targets xor output S-box in two different rounds. As depicted in equation 6, the ISCCA can eliminate the effect of u mask with Boolean summation. But if a cipher algorithm uses different mask for each round, the ISCCA will be an unusable attack.

$$\begin{gathered}\sum_{j=0}^{7} I_{0,S-Box}^{1,j}(V,K) + u + \sum_{j=0}^{7} I_{0,S-Box}^{2,j}(V,K) + u = \\ \sum_{j=0}^{7} I_{0,S-Box}^{1,j}(V,K) + \sum_{j=0}^{7} I_{0,S-Box}^{2,j}(V,K)\end{gathered} \tag{6}$$

8 Conclusion and Open Issues

In this paper, we present a general frame work for SCCA on block ciphers like AES and because of using iteration to reduce effect of key length in complexity and reduce data, time complexity, we call the proposed SCCA as Iterative SCCA (ISCCA). We define a practical scenario and show that is able to apply on AES with trivial complexity. The results show that the complexity of ISCCA is lower than other previous attacks on AES. The results also indicate

Table 10 The comparison between SCCA and other algorithms on AES in respect to memory complexity.

Attack	Key Size	Number of rounds	memory complexity
Square	All	6	2^{32}
Collision	256	7	2^{32}
Impossible differential	128	5	2^{42}
Impossible differential	256	7	2^{153}
Boomerang	128	6	2^{33}
SPA	128	FULL	$2^{28.26}$
SCCA	128	FULL	2^{28}
ISCCA	128	FULL	$2^{7.89}$
ISCCA	256	FULL	$2^{8.21}$

the ISCCA can be considered as a serious attack on block ciphers, in which the HW can be deduced accurately. However, although in white-box scenario block ciphers like AES are vulnerable, in black box scenario for finding number of interesting points and changing key make the SCCAs assumption too weak. Therefore, more researches in this field is flet.

References

1. I. Dinur, A. Shamir, "Side channel cube attacks on block ciphers", Cryptology ePrint Archive, http://eprint.iacr.org/2009/127.pdf
2. L. Yang, M. Wang, and S. Qiao,"Side Channel Cube Attack on PRESENT", *In Proceeding of the 8th International Conference on Cryptology and Network Security- CANS 2009*,LNCS, vol. 5888, pp. 379- 391.
3. S.F. Abdul-Latip, M.R. Reyhanitabar, W. Susilo, J. Seberry,"Extended cubes enhancing the cube attack by extracting low-degree non-linear equations",*In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security ASIACCS 2011*, ACM Society, pp. 296305, 2011.
4. X. Zhao, S. Guo, F. Zhang,T. Wang, Z. Shi, H. Liu, K. Ji,J. Huang,"Efficient Hamming weight-based side-channel cube attacks on PRESENT", *Journal of Systems and Software*, Volume 86, Issue 3, Pages 728743, March 2013.
5. X. Zhao, T. Wang, S. Guo, "Improved side channel cube attacks on PRESENT", Cryptology ePrint Archive, http://eprint.iacr.org/2011/165.pdf
6. X. Zhao, S. Guo, F. Zhang ,T. Wang, Z. Shi, H. Liu, K. Ji, J. Huang,"Black-Box Side-Channel Cube Attacks on Present-Like Ciphers", *IMCCC '13 Proceedings of the 2013 Third International Conference on Instrumentation*, Measurement, Computer, Communication and Control. Pp. 1352-1358, 2013.

7. Z. Li, B. Zhang, J. Fan, I. Verbauwhede,"A New Model for Error-Tolerant Side-Channel Cube Attacks",*Cryptographic Hardware and Embedded Systems - CHES 2013*, Lecture Notes in Computer Science Volume 8086, pp 453-470, 2013.
8. S.F. Abdul-Latip, M.R. Reyhanitabar, W. Susilo, J. Seberry, "On the Security of NOEKEON against side channel cube attacks" *In: Proceedings of the 5th InformationSecurity Practice and Experience Conference.* Future Conference ISPEC2010, LNCS, vol. 6047, pp. 4555, 2010.
9. Bard, N.T. Courtois, J. Nakahara, P. Sepehrdad, B. Zhang, "Algebraic, AIDA/cube and side channel analysis of KATAN family of block ciphers", *In: Progress in Cryptology-indocryt*, LNCS, vol. 6498, pp. 176196, 2010.
10. X. Fan, G. Gong, "On the security of Hummingbird-2 against side channel cube attacks", *In: Proceedings of WEWoRC 2011*, pp. 100104, 2011.
11. Z. Li, B. Zhang, Y. Yao, D. Lin," Cube Cryptanalysis of LBlock with Noisy Leakage",*Information Security and Cryptology ICISC 2012*, Lecture Notes in Computer Science Volume 7839, pp 141-155, 2013.
12. S. Islam, M. Afzal, A. Rashdi,"On the Security of LBlock against the Cube Attack and Side Channel Cube Attack",*Security Engineering and Intelligence Informatics*, Lecture Notes in Computer Science Volume 8128, pp 105-121, 2013
13. A. Kerchkoff. "La Cryptographie Militaire".*Journal des Sciences Militaires.*Vol 9,pp. 5-38, 1883.
14. J. Daemen,L. Knudsen, , and V. Rijmen, "The Block Cipher SQUARE. Fast Sofware Encryption", LNCS 1267, Springer-Verlag, pp. 149-165, 1997.
15. H. Gilbert, and M. Minier, "A collision attack on 7 rounds of Rijndael". 3rd AES candidate conference, New York, April 13-14, pp. 230-241, 2000.
16. E. Biham, and N. Keller, "Cryptanalysis of Reduced Variants of Rijndael". Submitted to the 3rd AES Candidate Conference. Available at: http://csrc.nist.gov/encryption/aes/round2/conf3 /papers/35-ebiham.pdf, 2000.
17. D. Wagner, "The Boomerang Attack". Fast Software Encryption 6, LNCS 1636, Springer-Verlag, pp. 156-170, 1999.
18. T. Jakobsen, and L. Knudsen, The interpolation attack on block ciphers. Fast Software Encryption. LNCS 1267, SpringerVerlag, pp. 28-40, 1997.
19. H. Dobbertin, L. Knudsen, and M. Robshaw, " The Cryptanalysis of the AES A Brief Survey". Advanced Encryption Standard AES: 4th International Conference, Bonn, Germany, 2004.
20. S. Mangard ,"A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion" ,Springer-Verlag Berlin Heidelberg, pp. 343358, 2003.
21. C. Clavier and K. Gaj, "Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA", CHES 2009, LNCS 5747, pp. 97111, 2009.
22. M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. Information Theory, IEEE Transactions on, 47(2):569 584, 2001.
23. S. Mangard, E. Oswald, R. Popp, "Power Analysis Attacks". Springer, Berlin Heidelberg, 2007.
24. I. Dinur, A. Shamir, "Cube Attacks on Tweakable Black Box Polynomials",EUROCRYPT 2009, LNCS 5479, pp. 278299, 2009.