The Journal of Systems and Software 86 (2013) 728–743

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Controversy Corner

Efficient Hamming weight-based side-channel cube attacks on PRESENT

Xinjie Zhao[a,b,*], Shize Guo[b], Fan Zhang[c], Tao Wang[a], Zhijie Shi[c], Huiying Liu[a], Keke Ji[a], Jing Huang[d]

[a] Department of Information Engineering, Ordnance Engineering College, Shijiazhuang 050003, China
[b] The Institute of North Electronic Equipment, Beijing 100083, China
[c] Department of Computer Science and Engineering, University of Connecticut, Storrs 06269, USA
[d] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 12 November 2011
Received in revised form 11 October 2012
Accepted 3 November 2012
Available online 16 November 2012

Keywords:
Hamming weight
Side-channel cube attack
PRESENT
Random delay
Masking

ABSTRACT

The side-channel cube attack (SCCA) is a powerful cryptanalysis technique that combines the side-channel and cube attack. This paper proposes several advanced techniques to improve the Hamming weight-based SCCA (HW-SCCA) on the block cipher PRESENT. The new techniques utilize non-linear equations and an iterative scheme to extract more information from leakage. The new attacks need only $2^{8.95}$ chosen plaintexts to recover 72 key bits of PRESENT-80 and $2^{9.78}$ chosen plaintexts to recover 121 key bits of PRESENT-128. To the best of our knowledge, these are the most efficient SCCAs on PRESENT-80/128. To show the feasibility of the proposed techniques, real attacks have been conducted on PRESENT on an 8-bit microcontroller, which are the first SCCAs on PRESENT on a real device. The proposed HW-SCCA can successfully break PRESENT implementations even if they have some countermeasures such as random delay and masking.

1. Introduction

The side-channel cube attack (SCCA) is a powerful attack proposed by Dinur and Shamir (2009). It combines the side-channel attack (SCA) (Kocher et al., 1999) and cube attack (Dinur and Shamir, 2008, 2009). The cube attack can break any cryptosystem where a bit in the ciphertext can be represented by a low degree multivariate polynomial composed of public and key variables. In practice, however, the cube attack can only break variants of ciphers with reduced rounds because the degrees of polynomials are very high in well designed ciphers. To reduce the complexity, SCA can be utilized to obtain information from the leakage (e.g., timing (Kocher, 1996), power (Kocher et al., 1999), and electromagnetic radiation (Quisquater and Samyde, 2000)) about the intermediate states of the cipher, where the polynomials have lower degrees. The combination of the two types of attacks results in SCCA (Dinur and Shamir, 2009).

Currently, there are two leakage models suitable for SCCA. A common one is *single bit leakage model* (SBLM), which targets only one bit each time. Due to the difficulties in obtaining the value of a single bit in a real experiment, current SBLM based SCCAs (SB-SCCAs) (Abdul-Latip et al., 2010, 2011; Bard et al., 2010; Dinur and Shamir, 2009; Yang et al., 2009; Zhao et al., 2011) are normally done with simulations. A more practical model is the *Hamming weight leakage model* (HWLM), which has been studied in many SCAs. The HWLM based SCCA (HW-SCCA) (Abdul-Latip et al., 2011) targets multiple bits (e.g., one byte). Compared with SBLM, the measurement cost of HWLM is much lower. As multiple bits are involved, more information from the leakage is available to adversaries in HWLM than in SBLM. Thus more key bits can be retrieved, even with fewer chosen plaintexts.

Since the first SCCA (Dinur and Shamir, 2009), many block ciphers have been analyzed, such as PRESENT (Abdul-Latip et al., 2011; Yang et al., 2009; Zhao et al., 2011), NOEKEON (Abdul-Latip et al., 2010), KATAN (Bard et al., 2010) and Hummingbird-2 (Fan and Gong, 2011). Most of the literature is based on SBLM. Only the work in Abdul-Latip et al. (2011) is based on a 64-bit HWLM. Meanwhile, there has been no physical experiment of SCCA. This paper proposes several enhancements to HW-SCCA and confirms the results with real experiment on a lightweight block cipher PRESENT (Bogdanov et al., 2007) implemented on an 8-bit processor. PRESENT has two versions: PRESENT-80 and PRESENT-128, which use 80 and 128 bits keys, respectively. The new techniques in this paper can break both. The new attacks are also effective on PRESENT implementations with some countermeasures such as random delay and masking.

1.1. Related work and motivations

Yang et al. (2009) proposed the first SB-SCCA on PRESENT-80. They extracted the linear equations of key variables from a single bit leakage in round 3. With 2^{15} chosen plaintexts, they extracted 48 key bits and reduced the key search space to 2^{32}. Zhao et al. (2011)

* Corresponding author. Tel.: +86 13643319969.
E-mail address: zhaoxinjieem@163.com (X. Zhao).

0164-1212/$ – see front matter © 2012 Published by Elsevier Inc.
http://dx.doi.org/10.1016/j.jss.2012.11.007

improved the attacks. Using the same leakage location in Yang et al. (2009), they showed that 48 key bits can be extracted with only $2^{11.92}$ chosen plaintexts. To recover more key bits, they also targeted a leakage bit in round 4. Abdul-Latip et al. (2011) studied HW-SCCA on PRESENT. Assuming that the 64-bit Hamming weight (HW) of the output of the first round is available, they exploited the non-linear equations of key variables besides the linear ones. Using about 2^{13} chosen plaintexts, they can recover 64 key bits for both PRESENT-80 and 128.

All of the previous SCCAs on PRESENT (Abdul-Latip et al., 2011; Yang et al., 2009; Zhao et al., 2011) are based on theoretical analysis and simulation. SB-SCCAs require the correct value of a single bit, which is difficult to achieve in practice. The work in Abdul-Latip et al. (2011) assumes accurate deductions of 64-bit HW from leakage, which is difficult in practice, too. In addition, the 64-bit value is usually produced by several sequential operations, and thus cannot be deduced by a single measurement.

We develop two techniques that allow us to improve HW-SCCAs, proposed in Abdul-Latip et al. (2011). The first is to exploit leakage states of smaller sizes, such as a byte (8-bit) or a nibble (4-bit), allowing for good measurements in real attacks and making the attacks applicable to more implementations. The second is to exploit the states in deeper rounds, which provide more information to recover more key bits. This paper exploits the HW of a byte instead of a 64-bit value, and studies the leakages in the third round instead of those in the first round. The reason that we choose to exploit the HW of a byte is that the 4-bit hardware platform is rarely in use. The example implementation of PRESENT is also written for 8-bit processors (Klose, 2011). The improved attacks are demonstrated on an 8-bit microcontroller ATMEGA324P, where the power consumption is highly correlated to the HW of the targeted state.

1.2. Contributions and organization

The main contribution of this paper is twofold.

On the theoretical side, the paper presents a framework of HW-SCCA, proposes several techniques to enhance HW-SCCA, and applies them to PRESENT. Section 3 describes the five major steps in HW-SCCA. Section 4 describes an efficient HW-SCCA on PRESENT under HW-HWLM in round 3. Firstly, the basic linear HW-SCCA is considered and it is shown that $2^{8.90}$ chosen plaintexts are required to reveal 48 key bits of PRESENT. We observe that many chosen plaintexts for different cubes are duplicated and the results in previous work (Abdul-Latip et al., 2011; Yang et al., 2009; Zhao et al., 2011) are inaccurate. Then, two techniques are introduced to enhance HW-SCCA by exploiting non-linear equations of key variables and recovering key bits iteratively. The improved attacks require $2^{8.95}$ chosen plaintexts to recover 72 key bits of PRESENT-80. Finally, the attacks are also extended to PRESENT-128 where 121 key bits can be recovered with only $2^{9.78}$ chosen plaintexts.

On the experimental side, the paper reports the results of real attacks on PRESENT and demonstrates the advantages of HW-SCCA over correlation power analysis (CPA) (Brier et al., 2004). Section 5 presents the results of HW-SCCAs on PRESENT implemented on an 8-bit microcontroller, which are the first SCCAs on block ciphers on a physical device. A new method is proposed to improve the accuracy and practicality of HW deductions. The results also show that with accurate HW deductions, HW-SCCA outperforms CPA especially when some countermeasures, such as random delay and masking, are used. Section 6 compares SCCA with other SCA techniques and discusses the limitations of HW-SCCA on PRESENT, further enhancements, and impacts on cipher designs.

2. Preliminaries

2.1. Cube attack

The cube attack (Aumasson et al., 2009; Dinur and Shamir, 2008, 2009; Lai, 1994; Vielhaber, 2007; Zhang et al., 2009) can be divided into several phases. The preprocessing phase determines what queries should be sent to a black-box oracle. The online phase deduces the values of the low-degree equations of key variables by querying a polynomial oracle with chosen public variables. Finally, solving a system of low-degree equations recovers the key.

A bit of an intermediate state in ciphers can be represented as a multivariate polynomial of public variables (e.g., plaintext variables in block ciphers and MACs, initial vector variables in stream ciphers) and key variables. Assume a cipher has m public variables $V=\{v_1, \ldots, v_m\}$ and n key variables $K=\{k_1, \ldots, k_n\}$. Each variable is a single bit. Let $X=V\cup K$. An intermediate bit can be described as a multivariate master-polynomial $f(X)$. Let the degree of $f(X)$ be $D_{\mathrm{eg}}(f)$. In the preprocessing phase, the adversary randomly chooses t_I, a product of the selected public variables. Then $f(X)$ can be represented as

$$f(X) = f(v_1, \ldots, v_m, k_1, \ldots, k_n) = t_I p_{S(I)} + q_I(X) \tag{1}$$

I, called *a cube*, is a set of indexes of public variables. An index in I is *a cube index*. t_I is a *maxterm* denoting the product of all public variables included in I. $p_{S(I)}$ is called a *superpoly* of I. q_I contains all monomials that are not divisible by t_I. For example, consider a polynomial $f(X)$ of degree 4 with 8 variables and 14 monomials:

$$\begin{aligned} f(X) &= v_1v_4k_1 + v_1v_4k_3 + v_2v_4k_4 + v_3v_4k_2 + v_1k_1k_2 + \\ &\quad v_1k_2k_3 + v_1k_1 + v_2v_4 + v_3v_4 + k_2k_3 + k_2k_4 + v_3 + v_4 + 1 \end{aligned} \tag{2}$$

If we merge monomials that have the same public variables, $f(X)$ can be written as

$$\begin{aligned} f(X) &= v_1v_4(k_1 + k_3) + v_2v_4(k_4 + 1) + v_3v_4(k_2 + 1) + \\ &\quad v_1(k_1 + k_1k_2 + k_2k_3) + k_2k_3 + k_2k_4 + v_3 + v_4 + 1 \end{aligned} \tag{3}$$

If $I=\{1, 4\}$, it is a cube of size 2. Then $t_I=v_1v_4$, $p_{S(I)}=k_1+k_3$, $q_I=v_2v_4(k_4+1)+\ldots+v_4+1$. Assume all public variables not in t_I are 0. For an assignment a to the public variables in t_I, $f(X)$ becomes $f_a(K)$. The sum of $f_a(K)$ over $GF(2)$ for all assignments is exactly $p_{S(I)}$. When $v_2=v_3=0$, there are four assignments to v_1 and v_4. The sum of the four $f_a(K)(0\le a\le 3)$ is k_1+k_3, which is the superpoly of $I=\{1,4\}$. If $f_a(K)$ can be deduced from SCA, k_1+k_3 is known. Repeating the process with other cubes can obtain more information about key bits.

2.2. The PRESENT algorithm

PRESENT is a 31-round block cipher with an SPN structure. The block size is 64 bits. The input to the first round is the plaintext and the ciphertext is the output of the 31st round XORed with the post-whitening key K^{32}. Each round consists of three major operations:

1 addRoundKey (AK). The 64-bit input is XORed with the round key.
2 sBoxlayer (SL). 16 identical 4×4 S-boxes are used in parallel.
3 pLayer (PL). Bit i is moved to position $P(i)$, as specified in a permutation table P.

Below is the description of the key schedule for PRESENT-80 (For PRESENT-128, please refer to Bogdanov et al. (2007)). The 80-bit key is stored in a register $K=k_{79}k_{78}\ldots k_0$. Then the round key $K^i i$ $(1\le i\le 31)$ and the post-whitening key K^{32} are generated by extracting the 64 leftmost bits of K. After each generation, K is rotated by 61 bits to the left. Then, an S-box operation is applied

to the 4 leftmost bits of K. Finally, a round-counter i is XORed with bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of K.

Table 1 defines the notations used in HW-SCCA on PRESENT.

3. The framework of HW-SCCA

In HW-SCCA, SCA measures the side-channel leakage, deduces the HW of the targeted state, and obtains the value of a single bit $f(X)$ while the cube attack uses $f(X)$ to recover the key. This section describes the framework of HW-SCCA.

3.1. Preprocessing phase

3.1.1. Identify the targeted state

The location of the targeted state is very important to HW-SCCA. A good location covers all the key variables while minimizing the complexity of the polynomial $f(X)$. We use three criteria from our empirical experiences to select the state.

(1) $N_k(X)$, the number of key variables that appear in $f(X)$. At the selected state, $N_k(X)$ should be as large as possible, to recover more key bits.

(2) $D_{eg}(X)$, the polynomial degree of $f(X)$. At the selected state, $D_{eg}(X)$ should be as small as possible. Since $D_{eg}(X)$ increases exponentially with the rounds, a state in deeper rounds is not preferred.

(3) $N_{mo}(X)$, the number of monomials in $f(X)$. At the selected state, $N_{mo}(X)$ should be minimized to reduce the length of the extracted superpolys and the complexity of merging monomials.

3.1.2. Represent the Hamming weight

In SB-SCCA, the deduced value from leakage is either 0 or 1, which can be used as the value of $f(X)$. In HW-SCCA, however, it is necessary to explore how $H(X)$ should be used. Take one byte $X=(x_7x_6x_5x_4x_3x_2x_1x_0)$ as an example. $H(X)$ can be described as a 4-bit value $Y=(y_3y_2y_1y_0)$. x_0 and y_0 are the least significant bits (LSBs). The bits in Y can be calculated as:

$$y_3=\prod_{i=0}^{7}x_i,\qquad y_2=\sum x_ix_jx_mx_n(0\le i<j<m<n\le 7),$$
$$y_1=\sum x_ix_j(0\le i<j\le 7),\qquad y_0=\sum_{i=0}^{7}x_i \tag{4}$$

Any one of the bits in Y is dependent on all eight bits in X and can be utilized in cube attacks. Since the degree of y_0 is 1, y_0 is the best option. Although y_1, y_2, y_3 can also be utilized, they lead to much larger complexities.

3.1.3. Extract superpolys

In white-box attack scenarios as in Yang et al. (2009), the adversaries know the detail of the cipher design and can represent the polynomial $f(X)$ with the public and key variables. They can merge the monomials that have the same public variables and extract the cubes and corresponding superpolys. To accelerate the search for superpolys, a common method is to merge the monomials that have at most one key variable (i.e., one key variable and some public variables or public variables only). However, this method does not work well when the cube size is large. New techniques are needed for extracting superploys.

3.2. Online phase

3.2.1. Deduce the Hamming weight

In this step, the values of $H(X)$ are deduced from the leakage. Firstly, a HW model is established to describe the leakage patterns for different HWs of the targeted state. Secondly, leakage traces are collected when chosen plaintexts are encrypted on the device with the unknown key. Finally, the correlations between leakages traces and the HW model are examined to obtain the correct HW.

3.2.2. Recover the secret key

In this step, the value of the superpolys is computed from the symbolic sums of y_0 in $H(X)$ over GF(2). Many low-degree equations on the key bits are obtained. Then existing tools, such as zChaff (Princeton University, 2007), MiniSat (Een and Sorensson, 2005), and CryptoMiniSat (Soos et al., 2011), can be leveraged to solve the equations and retrieve the involved key bits. If necessary, a brute-force search can be invoked to recover the full key.

4. Efficient HW-SCCA on PRESENT

This section introduces the preprocessing phase of efficient HW-SCCA on PRESENT. The HWLM used here is different from that in Abdul-Latip et al. (2011). The targeted state is a byte, not a 64-bit value and the byte is in the third round, not in the first round. As listed in Table 1, $L^{i,j}_{m,n}$ denotes a bit in the HW, where $i\in[1\ldots31]$ is the round number, $j\in\{AK, SB, PL\}$ is the operation, $m\in[1\ldots8]$ is the index of the targeted byte, and $n\in[0\ldots3]$ is the bit index in the HW representation.

4.1. Identify the targeted byte

In PRESENT, the output of PL in the second round is the place where all 64 bits of K^1 are involved in one byte for the first time. In this paper, the targeted PRESENT implementation is written by Klose Klose (2011). In this implementation, deducing the HW of the output of AK is easier than the output of PL. So one byte after the AK step in the third round, $L^{3,AK}_m$, will be the targeted state.

Fig. 1(a) shows how $D_{eg}(L^{3,AK}_{m,n})$ changes with m and n after the AK step in round 3. Among the choices for n, $D_{eg}(L^{3,AK}_{m,0})$ is the smallest. Among the values of m, $D_{eg}(L^{3,AK}_{1,0})$ and $D_{eg}(L^{3,AK}_{2,0})$ are relatively small. Fig. 1(b) shows how the number of monomials changes with m. Since $N_{mo}(L^{3,AK}_{1,0})$ and $N_{mo}(L^{3,AK}_{2,0})$ are small, $L^{3,AK}_{1,0}$ and $L^{3,AK}_{2,0}$ are identified as the targeted byte.

4.2. HW-SCCA with linear superpolys

We start our search for the superpolys using the common method mentioned in Section 3. We choose n=0 and consider only the linear part of a HW representation. A basic HW-SCCA on both $L^{3,AK}_{1,0}$ and $L^{3,AK}_{2,0}$ are performed. The result for $L^{3,AK}_{2,0}$ is listed in Table 2, which has 56 linear superpolys. In comparison, the estimated value of $N_{cp}(L^{3,AK}_{2,0})$ is $24\times2^2+32\times2^5\approx2^{10.13}$. The result for $L^{3,AK}_{1,0}$ is listed in Table 7.

4.3. New observations on the number of chosen plaintexts

In the attack, we observed that when the plaintexts are chosen based on the cubes listed in Table 2, many of them are actually duplicated. Recall the simple example in Section 2. Four cubes can be identified {1, 4},{2, 4},{3, 4},{1}. An assignment $v_1=v_2=v_3=v_4=0$ can be used as a chosen plaintext for all four cubes. After removing the duplicated chosen plaintexts in Table 2, $N_{cp}(L^{3,AK}_{2,0})$ can be further reduced from $2^{10.13}$ down to $2^{8.90}$=477.

All existing analyses of SCCAs have simply counted the number of chosen plaintexts based on the sizes of cubes. Our observations reduce the number of chosen plaintexts actually needed in practice. Table 3 lists the updated number of chosen plaintexts for some existing SCCAs on PRESENT.

Table 1
Notations used in this paper.

Variable	Notations	Variable	Notations
N_{SI}	Cube size of I	$L_m^{i,j}$	The m-th byte of the j-th operation in round i
k_i	The i-th bit of master key	$L_{m,n}^{i,j}$	The n-th HW bit of $L_m^{i,j}$
K^i	Subkey in round i	$f(X)$	Polynomial representation of X
K_{SI}	Superpoly of I	$N_k(X)$	Number of key variables in $f(X)$
K_d	New key deductions	$D_{eg}(X)$	Polynomial degree of $f(X)$
$H(X)$	HW of X	$N_{mo}(X)$	number of Monomials in $f(X)$
$H_D(X)$	Deduction of $H(X)$	$N_{cp}(X)$	Number of chosen plaintexts for X

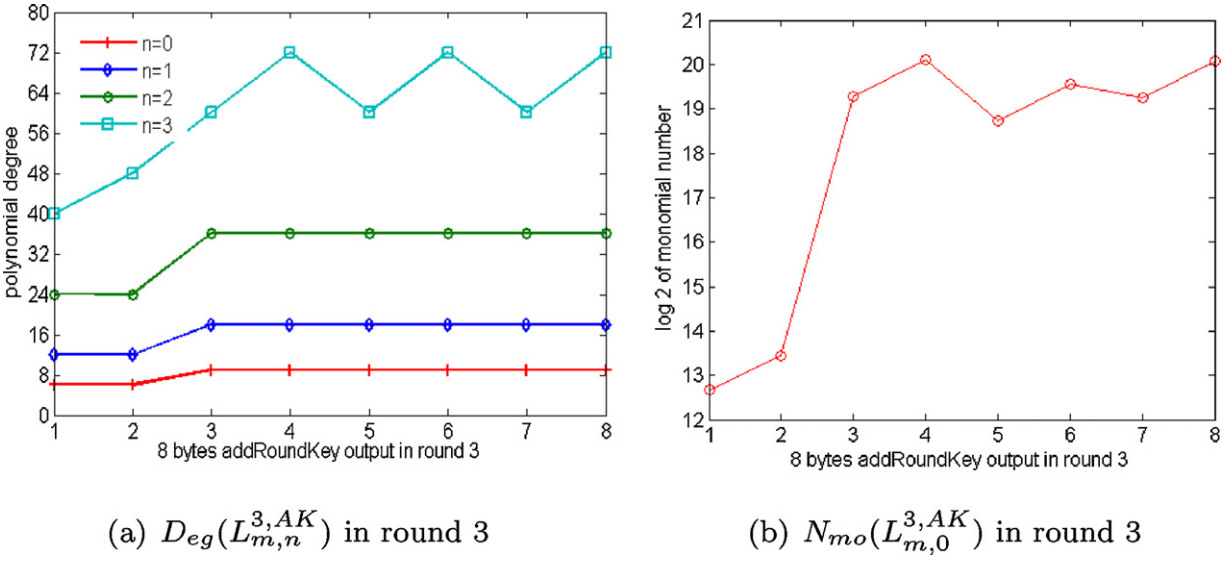


Fig. 1. $D_{eg}(L_{m,n}^{3,AK})$ and $N_{mo}(L_{m,n}^{3,AK})$ in round 3. (a) $D_{eg}(L_{m,n}^{3,AK})$ in round 3; (b) $N_{mo}(L_{m,0}^{3,AK})$ in round 3.

Table 2
Cube indexes and superpolys for $N_k(L_{2,0}^{3,AK})$ (linear equations).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
1, 2	$1+k_{16}$	32, 33	$1+k_{50}$	4, 5, 6, 9, 10	$1+k_{24}$	36, 37, 40, 41, 43	$1+k_{54}$
0, 2	k_{17}	45, 46	$1+k_{60}$	4, 5, 7, 8, 10	k_{25}	36, 37, 40, 41, 42	$k_{54}+k_{55}$
0, 1	$1+k_{18}$	44, 46	k_{61}	4, 5, 7, 8, 9	$1+k_{26}$	36, 37, 38, 41, 42	$1+k_{56}$
13, 14	$1+k_{28}$	44, 45	$1+k_{62}$	4, 5, 6, 8, 9	$k_{26}+k_{27}$	36, 37, 39, 40, 42	k_{57}
12, 14	k_{29}	49, 50	$1+k_{64}$	21, 22, 24, 25, 26	$1+k_{36}$	36, 37, 39, 40, 41	$1+k_{58}$
12, 13	$1+k_{30}$	48, 50	k_{65}	20, 22, 24, 25, 27	k_{37}	36, 37, 38, 40, 41	$k_{58}+k_{59}$
17, 18	$1+k_{32}$	48, 49	$1+k_{66}$	20, 21, 24, 25, 27	$1+k_{38}$	53, 54, 56, 57, 58	$1+k_{68}$
16, 18	k_{33}	61, 62	$1+k_{76}$	20, 21, 24, 25, 26	$k_{38}+k_{39}$	52, 54, 56, 57, 59	k_{69}
16, 17	$1+k_{34}$	60, 62	k_{77}	20, 21, 22, 25, 26	$1+k_{40}$	52, 53, 56, 57, 59	$1+k_{70}$
29, 30	$1+k_{44}$	60, 61	$1+k_{78}$	20, 21, 23, 24, 26	k_{41}	52, 53, 56, 57, 58	$k_{70}+k_{71}$
28, 30	k_{45}	5, 6, 8, 9, 10	$1+k_{20}$	20, 21, 23, 24, 25	$1+k_{42}$	52, 53, 54, 57, 58	$1+k_{72}$
28, 29	$1+k_{46}$	4, 6, 8, 9, 11	k_{21}	20, 21, 22, 24, 25	$k_{42}+k_{43}$	52, 53, 55, 56, 58	k_{73}
33, 34	$1+k_{48}$	4, 5, 8, 9, 11	$1+k_{22}$	37, 38, 40, 41, 42	$1+k_{52}$	52, 53, 55, 56, 57	$1+k_{74}$
32, 34	k_{49}	4, 5, 8, 9, 10	$k_{22}+k_{23}$	36, 38, 40, 41, 43	k_{53}	52, 53, 54, 56, 57	$k_{74}+k_{75}$

Table 3
New chosen plaintext number of SCCAs on PRESENT.

Ref.	Previous chosen plaintext number	New chosen plaintext number
Yang et al. (2009)	2^{15}	$2^{12.50}$
Zhao et al. (2011)	$2^{11.92}$	$2^{10.88}$
Abdul-Latip et al. (2011)	2^{13}	$2^{12.27}$
Section 4.2	$2^{10.13}$	$2^{8.90}$

Table 4
Cube indexes and superpolys for $N_k(L_{2,0}^{3,AK})$ (non-linear superploys).

I	K_{SI}	K_d	I	K_{SI}	K_d
1	$1+k_{16}+k_{19}+k_{18}+k_{16}k_{18}$	k_{19}	33	$1+k_{48}+k_{51}+k_{50}+k_{48}k_{50}$	k_{51}
13	$1+k_{28}+k_{31}+k_{30}+k_{28}k_{30}$	k_{31}	45	$1+k_{60}+k_{63}+k_{62}+k_{60}k_{62}$	k_{63}
17	$1+k_{32}+k_{35}+k_{34}+k_{32}k_{34}$	k_{35}	49	$1+k_{64}+k_{67}+k_{66}+k_{64}k_{66}$	k_{67}
29	$1+k_{44}+k_{47}+k_{46}+k_{44}k_{46}$	k_{47}	61	$1+k_{76}+k_{79}+k_{78}+k_{76}k_{78}$	k_{79}

4.4. Enhanced HW-SCCA with non-linear superpolys

Section 4.2 only explores the linear superpolys. In fact, we can utilize non-linear superpolys with quadratic terms to improve HW-SCCAs (Abdul-Latip et al., 2011; Zhang et al., 2009). Unlike the basic HW-SCCA, we now merge the monomials that have at most two (instead of one) key variables for each cube. The best result for $L_{2,0}^{3,AK}$ is listed in Table 4, which has eight non-linear superpolys. Take the first row of the left half of Table 4 as an example. When I={1}, the non-linear superpoly $1 + k_{16} + k_{19} + k_{18} + k_{16}k_{18}$ can be extracted. Since both k_{16} and k_{18} are already recovered in Table 2, k_{19} can be determined.

Since all the cube indexes in Table 4 already appear in Tables 2 and 4 does not introduce any new chosen plaintexts. The results of HW-SCCA with non-linear terms for $L_{1,0}^{3,AK}$ can be found in Table 8.

4.5. Enhanced HW-SCCA with iterative scheme

To further reduce the key search space, we propose an enhanced SCCA scheme, called the *iterative SCCA*. The main idea is to substitute the key variables with the recovered key bits, update f, V and K, and repeat the standard SCCA. As fewer unknown key variables are in f, both D_{eg} and N_m are reduced. More maxterms and superpolys can then be retrieved. Note that in the iterative SCCA, some of the chosen public variables may have minor changes.

As shown in Tables 2 and 4, if $L_{2,0}^{3,AK}$ is selected, we can extract 64 key bits of K^1 (i.e., $k_{79} \ldots k_{16}$ in the master key). To obtain $k_{15} \ldots k_0$, we substitute K^1 in f with the key bits already recovered. The 64 bits at the output of round 1 ($PL(SL(AK(P, K^1))))$ are used as new public variables, as shown in the iterative place 2 in Fig. 2. Then the previous three-round HW-SCCA on PRESENT-80 becomes a new two-round HW-SCCA. The complexity of f is reduced significantly.

Table 5 shows that 2^5 chosen plaintexts can recover 8 more key bits, thus reducing the key space to 2^8. Considering the duplicated chosen plaintexts, only 18 more plaintexts are actually required. Combining the results above, in total, 495 ($2^{8.95}$) chosen plaintexts are enough to recover 72 key bits. The results of iterative HW-SCCA on $L_{1,0}^{3,AK}$ can be found in Table 9.

5. Physical experiments

This section presents the online phase of HW-SCCA on PRESENT. The power leakage of PRESENT implemented on an 8-bit microcontroller is collected to deduce the values of the superpolys extracted in previous sections. To improve the practicality of HW-SCCA, an accurate HW deduction method based on Pearson correlation factor is proposed.

In traditional SCAs, DPA (Kocher et al., 1999) and CPA (Brier et al., 2004) are two techniques that are widely used. When attacking the implementations on microcontrollers, CPA is more effective than DPA (Coron and Kizhvatov, 2009, 2010; Kocher et al., 2011). Therefore, we use CPA to evaluate the resistance of PRESENT against SCAs and compare it with HW-SCCA. To fully investigate the advantages of HW-SCCA, we conduct CPAs and HW-SCCAs on both unprotected and protected implementations of PRESENT-80. The protected implementations have two types of countermeasures: random delay and masking.

5.1. Implementations of PRESENT on an 8-bit microcontroller

The targeted device is the AMTEL ATMEGA324P 8-bit microcontroller. It runs at 8 MHz. A digital oscilloscope MSO6012A measures the voltage drop on a resistor connected to ATMEGA324P. The oscilloscope communicates with a PC via a USB interface. The sampling rate is 100M points/s.

Our attacks are performed on an 8-bit implementation of PRESENT written by Klose (2011). Both the size optimized and speed optimized implementations use the same C code for the *AK* operation, which updates one byte (instead of a nibble) of the state at a time. The results in this section are from the attacks on the size optimized version of PRESENT-80, targeting a byte after the *AK* step in round 3.

5.2. Enhanced HW-SCCA with accurate HW deduction

In practice, the measured power traces come with a lot of noise, which makes obtaining accurate HW deductions difficult. Any error in the value of $H(X)$ may cause the cube attack to fail. In this section, we introduce new techniques that can deduce the HW value from physical traces with a very high accuracy.

A HW power model must be built first, which normally depends on the targeted operations. To build the power model, we randomly choose pairs of plaintext and key, and collect power traces of the entire encryption for each pair. We separate all the traces into nine categories based on the HW of the first output byte of *AK* in round 3. The power profile for each HW value is shown on the left side of Fig. 3 and each curve is the average of 1000 traces.

To deduce the HW, adversaries calculate the correlation coefficients of two vectors(e.g., the measured trace and the power model). The Pearson correlation factor is widely used in SCAs (Brier et al., 2004; Mayer-Sommer, 2000; Standaert et al., 2004). However, this method does not work very well in deducing the HW from a small number of power traces. For example, four power traces are shown on the right part of Fig. 3, where four different plaintexts are XORed with the same key and the results have different HWs. The correct values are 3, 2, 2 and 3. The deduced values $H_D(X)$ by Pearson correlation factor are 4, 1, 1 and 3, which have ± 1 errors.

We propose a new technique to improve the HW deduction. We introduce a new parameter δ, a metric of the vertical voltage variance in addition to the horizontal phase offset. Suppose P_M is the model and P_R is the measured trace. The new correlation factor $\rho(P_R, P_M)$ is shown in Eq. 5.

$$\rho(P_R, P_M) = \frac{\delta . \sum_{p_r \in P_R, p_m \in P_M} (p_r - \overline{P_R}).(p_m - \overline{P_M})}{(\sqrt{\sum_{p_r \in P_R} (p_m - \overline{P_M})^2 \cdot \sum_{p_m \in P_M} (p_m - \overline{P_M})^2})} \quad (5)$$

$$\delta = \begin{cases} \dfrac{\max(P_M)}{\max(P_R)}, & if \quad \max(P_R) > \max(P_M) \\ \dfrac{\max(P_R)}{\max(P_M)}, & if \quad \max(P_R) \leq \max(P_M) \end{cases}$$

To improve the success rate, we collect the power traces n times for the same plaintext, and obtain n HW deductions. We calculate the frequency of each possible HW value in the deductions and select the one with the maximal frequency. In our experiments, if $n \geq 6$, the probability of getting the correct value is 100%.

5.3. Attacks on unprotected PRESENT

5.3.1. CPA on PRESENT

CPA is a variant of DPA. It guesses the values of some key bits and calculates the HW of the targeted state. The guess is verified with the Pearson correlation factors between the measured trace and the computed HW from the guess.

We choose the first nibble of the output of S-box in the first round as our target in CPA. Fig. 4(a) shows the CPA results for the

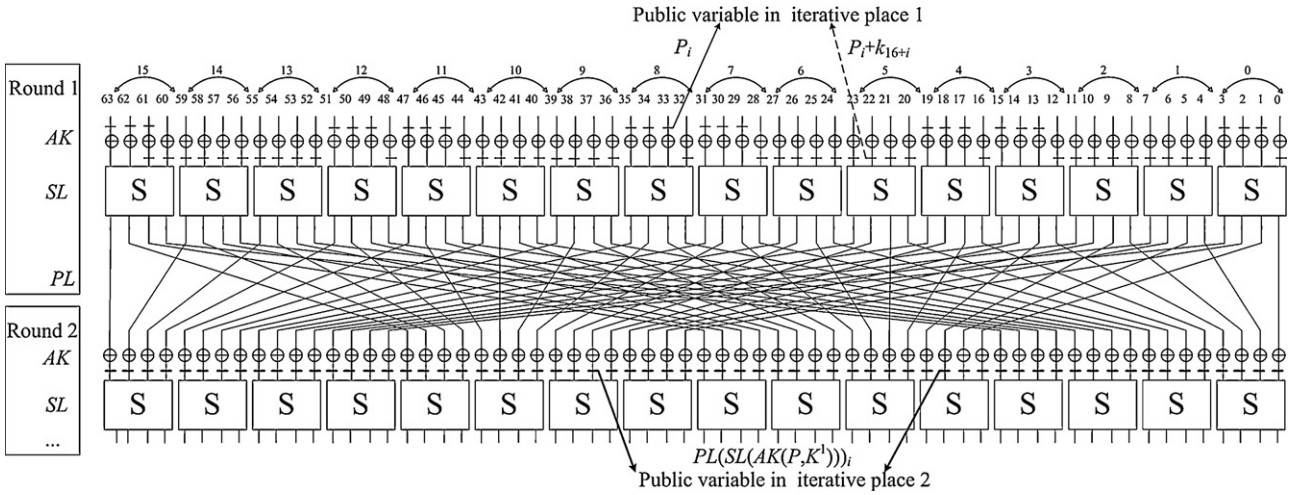


Fig. 2. Locations of the public variables in two iterative SCCAs.

Table 5
Cube indexes and superpolys for $N_k(L_{2,0}^{3,AK})$ (SCCA with iterations).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
32, 46	$1+k_0$	32, 50	$1+k_4$	32, 54	$1+k_8$	32, 58	$1+k_{12}$
32, 45	k_1	32, 49	k_5	32, 53	k_9	32, 57	k_{13}

correct and wrong key guesses. The red curve for the correct key guess has a much higher peak than those in the blue curves (for wrong guesses). Fig. 4(b) shows how the correlation coefficient for the correct key guess changes with the number of power traces. Fig. 4 shows that for the unprotected PRESENT implementation, 200 power traces are enough to recover the first nibble of K^1. To recover all 16 nibbles, our CPA needs 1024 traces.

5.3.2. HW-SCCA on PRESENT

In HW-SCCA on PRESENT, we choose $L_{2,0}^{3,AK}$ as the targeted byte. We first generate the plaintexts according to the extracted cubes in Tables 2, 4 and 5, then deduce the HW with the method in Section 5.2, and finally recover the values of the superpolys by the symbolic sum over $GF(2)$.

Take Fig. 5 as an example. Fig. 5 (a)–(d) shows the four power traces for encrypting different plaintexts for the cube $I=\{1, 2\}$ and the superpoly $K_{SI}=1+k_{16}$. The key is fixed in four encryptions. We can deduce four HWs, 4, 4, 6, and 4, from the traces. The LSBs of the four HW values are 0. Since $1+k_{16}=0+0+0+0=0$, we can deduce $k_{16}=1$. Using the techniques described in Section 4, we have conducted successful attacks on PRESENT implemented on the 8-bit processor. The results show that 72 key bits of PRESENT-80 can be

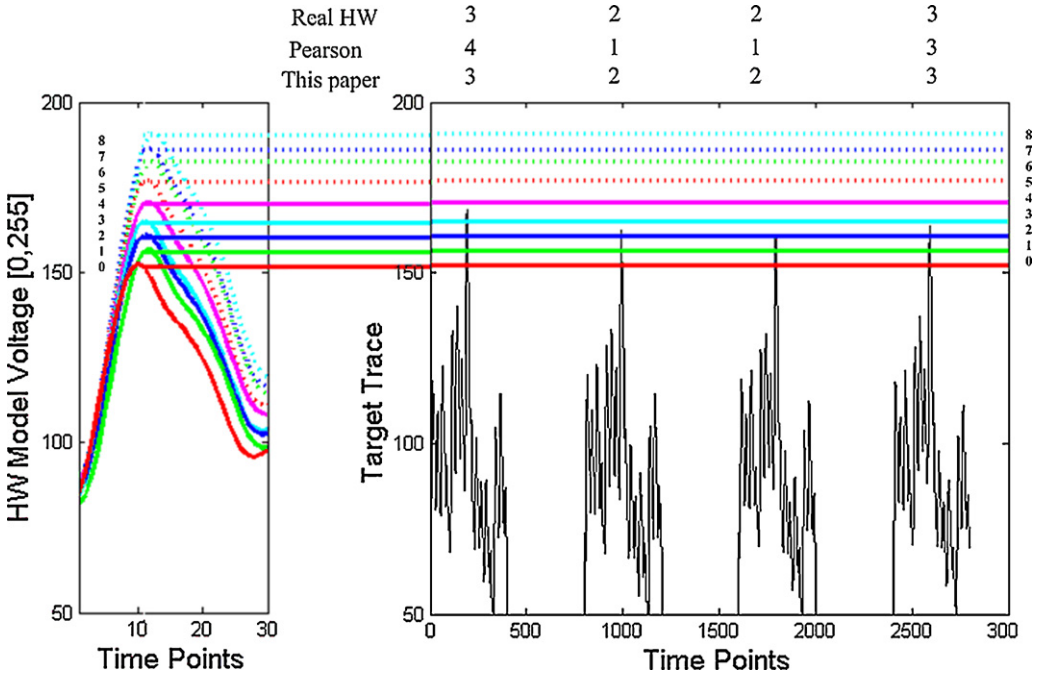


Fig. 3. Hamming weight deductions in SCCA.

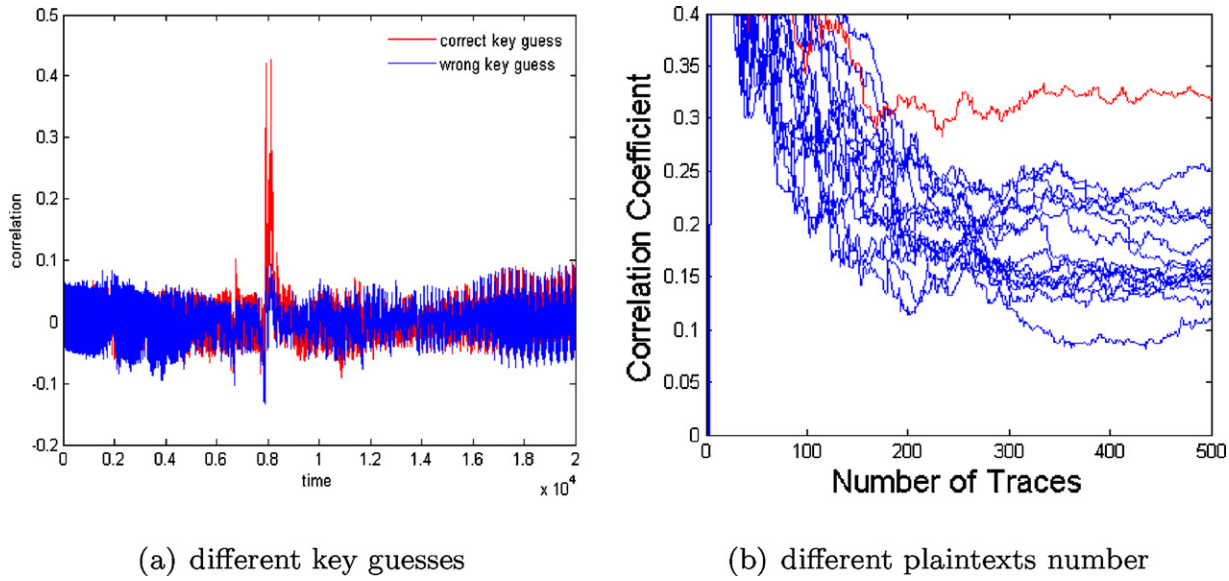


Fig. 4. CPA on PRESENT without Countermeasures (Nibble 1 of K^1). (a) Different key guesses; (b) different plaintexts number. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

recovered with $2^{8.95}$ chosen plaintexts and $2^{11.54} = 6 \times 2^{8.95}$ power traces.

5.4. Attacks on PRESENT protected with random delays

Inserting random delays (Coron and Kizhvatov, 2009, 2010) can change the execution time of the encryption. The randomized delays introduce more noise and make the analysis, such as data alignment and trace average, more difficult. This technique can be used to defeat many SCAs including CPA. It has been widely adopted in many embedded systems.

In most cases, the random delays are implemented with specific numbers of NOP loops. As a result, the delays consume less power than normal operations, and can be identified if the delay is long enough.

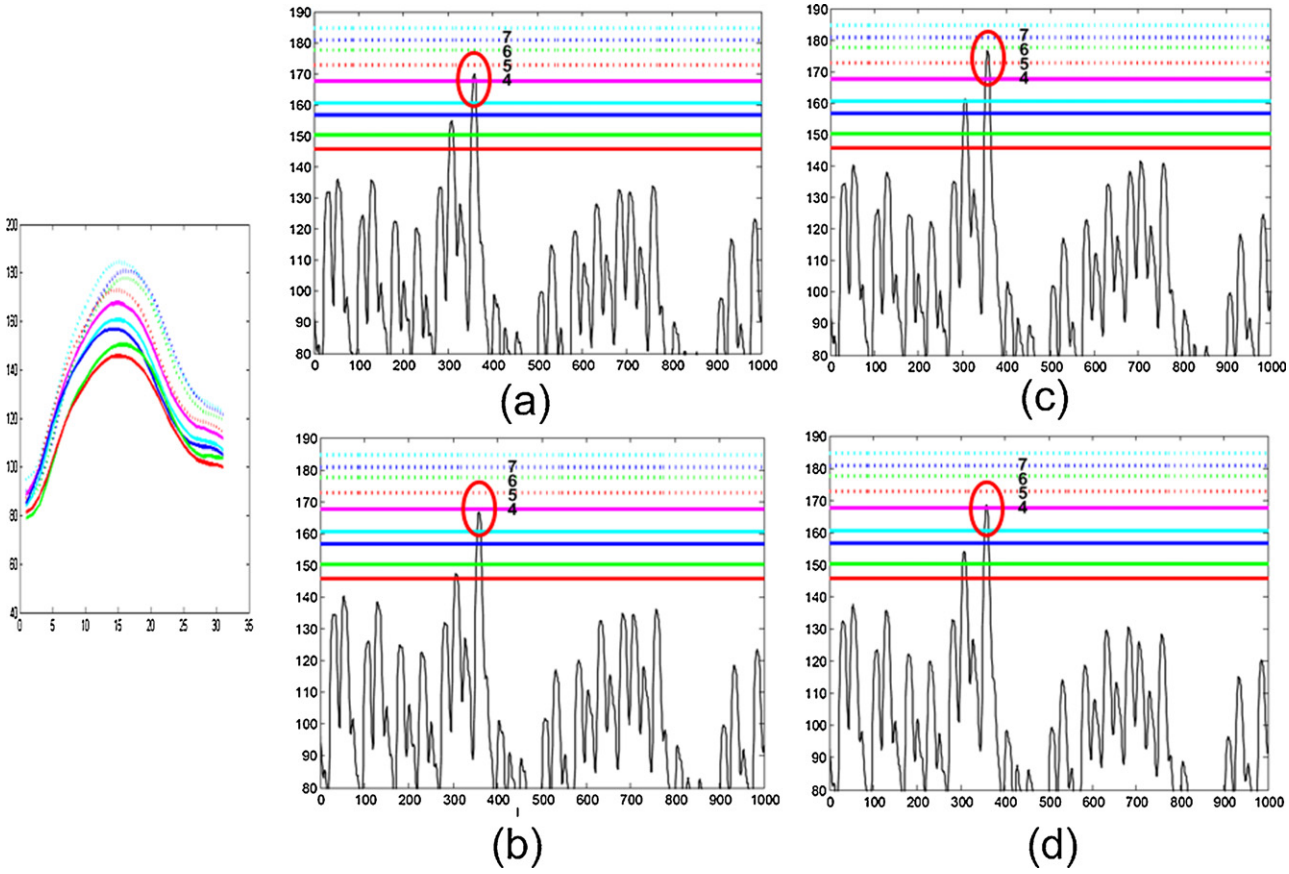


Fig. 5. HW-SCCA on PRESENT without countermeasures.

The effects of random delays depend on the amount of the delays and the way they are added. Suppose the lengths of the delays follow a normal distribution. A simple way to accomplish this is to add a delay with large mean and variance at one location. A more advanced way is to introduce delays at many locations (different operations, different rounds), where each delay has a smaller mean and variance (Coron and Kizhvatov, 2009, 2010). The latter is normally combined with a dummy round to introduce more randomness, although it also causes more performance degradation.

In our experiments, we insert two delays before and after the first *AK* in the first round. The total delay is approximately $3r$ cycles where r is uniformly distributed in $[0 \dots d]$ and d is a parameter. Three experiments are conducted with different values of d.

5.4.1. CPA on PRESENT

Fig 6(a) shows the results for $0 \le d \le 7$. 3000 traces are required in CPA to reveal nibble 1. Fig 6(b) shows the results for $0 \le d \le 15$. 6000 traces are required. Fig 6(c) shows the results for $0 \le d \le 127$. We are not able to guess the correct value of nibble 1 even with 32,768 traces. It is clear to see that CPA can be defeated by random delays.

5.4.2. HW-SCCA on PRESENT

As many block ciphers adopt an iterative structure, the power consumption for different rounds and operations can be easily identified, as noted in Kocher et al. (1999). In our implementation with random delays, the power pattern for the leakage byte in the third round does not change by much. It is easy to identify that pattern and deduce the HW leakage from a single power trace. The data alignment is not required.

Fig. 7 shows the four power traces on $L_{2,0}^{3,AK}$ when encrypting different plaintexts for the cube $I = \{1, 2\}$ and superpoly $K_{SI}=1+k_{16}$. Random delays are added into the implementation. It is clear to see that although the horizontal time position of $L_{2,0}^{3,AK}$ is different in the four traces, the HWs can still be easily deduced and the values are 4, 4, 6, and 4. 72 key bits of PRESENT can also be recovered with $2^{8.92}$ chosen plaintexts and $2^{11.54}$ ($n=6$) power traces.

5.5. Attacks on PRESENT protected with masking

Masking (Akkar and Giraud, 2001; Chari et al., 1999; Goubin and Patarin, 1999; Mangard et al., 2007) is another solution to prevent SCAs. It randomizes the sensitive data and minimizes the correlations between data and leakages. This paper will focus on Boolean masking (Akkar and Giraud, 2001; Chari et al., 1999; Goubin and Patarin, 1999; Mangard et al., 2007).

We implement PRESENT with a masked substitution table proposed by Akkar and Giraud (2001). For each encryption, a 64-bit mask u is randomly generated for the input of *AK*. The mask v for the output of *SB* is chosen as $v = PL^{-1}(u)$. In Fig. 8(a), p_i and k_i are two inputs of *AK*, and $p_i \oplus k_i$ and $S(p_i \oplus k_i)$ are the output of the *AK* and *SB*. The masked table S' is defined as $S'(p_i \oplus k_i \oplus u_i) = S(p_i \oplus k_i) \oplus v_i$, where u_i and v_i are the masks of the i-th nibble of the S-box input and output. $p_i, k_i, u_i, v_i \in GF(2^4)$, $0 \le i \le 15$. In this implementation, the same masks u and v are used in all rounds of one PRESENT execution because recalculation of S' for each new pair (u, v) is too expensive. The values of u_i for the 16 S-boxes are different in the same round.

5.5.1. CPA on PRESENT

Fig. 8(b) shows the CPA coefficient curves for 16 key guesses on the first nibble of K^1 with the increase of the number of power traces. The CPA does not work on implementations with random masking.

5.5.2. HW-SCCA on PRESENT

In HW-SCCA on the same implementation, we measure the HW leakage of $L_m^{2,AK}$ and $L_m^{3,AK}$. We choose the leakage bit as $\{L_{m,0}^{2,AK} \oplus L_{m,0}^{3,AK}\}$, instead of $L_{m,0}^{3,AK}$. As the same u is applied to both round 2 and 3, the new targeted bit is independent of u. Since the polynomial of this bit is slightly different from that in Section 4, we need to update the cubes and superpolys to be extracted. The results of SCCA on $\{L_{1,0}^{2,AK} \oplus L_{1,0}^{3,AK}\}$ and $\{L_{2,0}^{2,AK} \oplus L_{2,0}^{3,AK}\}$ are listed in Tables 20–25, respectively.

Fig. 9 displays the power traces of $L_2^{2,AK}$ and $L_2^{3,AK}$ for the cube $I=\{1, 2\}$ and the superpoly $p_{S(I)}=1+k_{16}$ (listed in Table 23 for masked implementations). Using the method in Section 4, the HW values of $L_2^{2,AK}$ and $L_2^{3,AK}$ for four chosen plaintexts are $\{4, 2, 5, 2\}$ and $\{6, 3, 5, 3\}$ respectively. We can compute the four values of $\{L_{2,0}^{2,AK} \oplus L_{2,0}^{3,AK}\}$ as $\{0, 1, 0, 1\}$, and then deduce that k_{16}=1. 72 key bits of PRESENT can still be recovered with $2^{8.92}$ chosen plaintexts and $2^{11.54}$($n=6$) power traces.

6. Discussions

6.1. Comparison of HW-SCCA, DPA, and ASCA

HW-SCCA, DPA (CPA), and recently proposed ASCA (algebraic side-channel attack (Renauld and Standaert, 2009; Renauld et al., 2009; Zhao et al., 2012)), have different assumptions, thus resulting in different strengths.

6.1.1. DPA (CPA)

This type of attack leverages statistical techniques to identify the relation between data and leakage. It requires few assumptions and is applicable to many platforms. The attack targets the first or last round of ciphers. A divide-and-conquer strategy is usually adopted for the key recovery. The full key is recovered in a byte-by-byte, or nibble-by-nibble manner. Recovering one byte or one nibble of the key requires relatively large numbers of traces although the numbers are highly dependent on the targeted implementations. In addition, the attack is much less effective when countermeasures are present.

6.1.2. ASCA

ASCA can utilize the leakage in all rounds and even break a system with a single trace. In ASCA, all the key bits can be recovered at the same time. The attack can work under unknown plaintext/ciphertext scenarios and defeat certain masking countermeasures. ASCA has strong assumptions on the leakage model and the capability of the adversaries (e.g., the HWs of the intermediate states can be deduced accurately). ASCA needs to profile and analyze many leakages in a single trace. More efforts are spent in building templates for every leakage location.

6.1.3. SCCA

Both SCCA and ASCA have strong assumptions on the leakage model and the capability of the adversaries. Different from DPA (CPA), SCCA deduces the intermediate states from a single power trace. As a result, some countermeasures like random delays are less effective in defeating SCCA. Similar to ASCA, SCCA uses the leakage in deeper rounds. Both SCCA and DPA (CPA) rely on the divide-and-conquer strategy and SCCA recovers one key bit each time. On average, SCCA requires fewer traces than DPA (CPA). As only one leakage in a trace is utilized (templates for single leakages are required), profiling is easier in SCCA than in ASCA. Both SCCA and ASCA can break some masking schemes while it is more difficult for CPA. Most importantly, inheriting the advantages of cube attacks, SCCA can work under black-box settings where neither ASCA nor DPA can. This is a new threat to the security of cipher

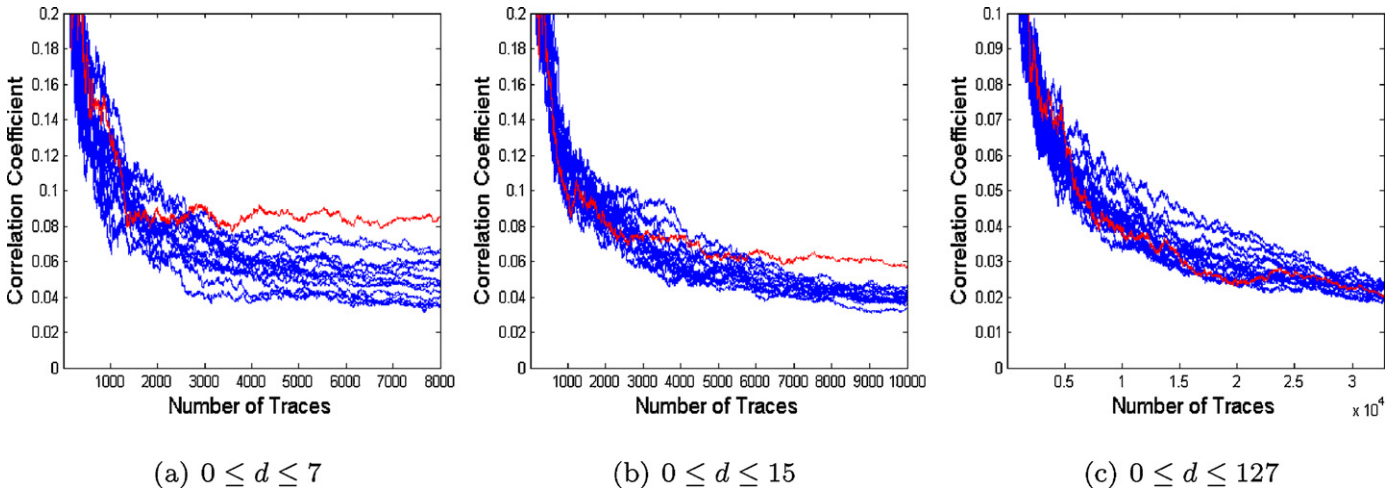


Fig. 6. CPA on PRESENT with random delay countermeasures (Nibble 1 of K^1). (a) $0 \le d \le 7$; (b) $0 \le d \le 15$; (c) $0 \le d \le 127$.

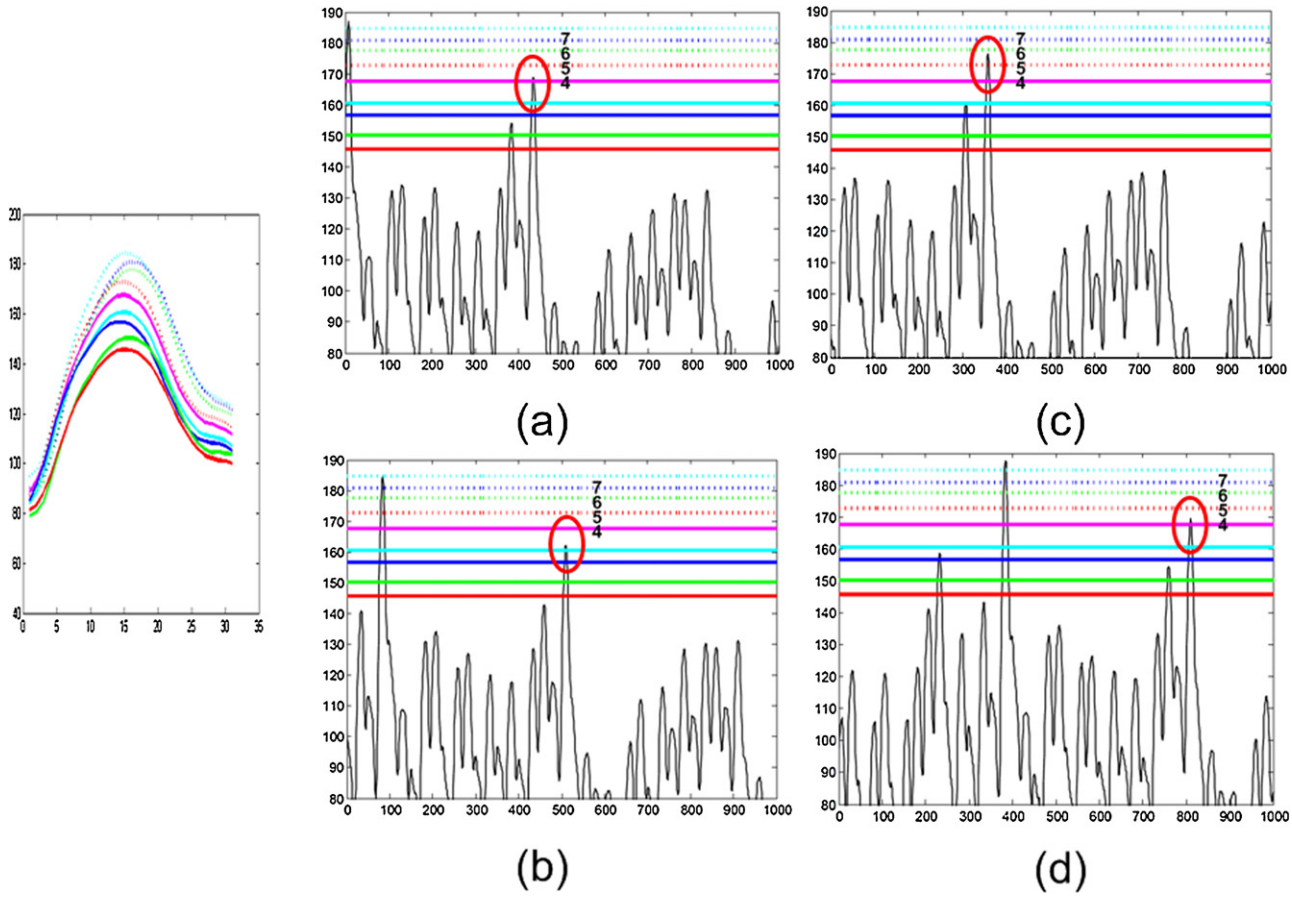


Fig. 7. HW-SCCA on PRESENT with random delay countermeasures.

Table 6
Comparisons between SCCA and SCAs.

Comparisons		DPA (CPA)	ASCA	SCCA
Assumptions		Weak	Strong	Strong
Leakages Measurement	location	First or last round	Any round	Deeper round
	Number	Medium per trace	Maximized per trace	Single per trace
	Difficulty	Easy	Difficult	Medium
Key recovery		Byte (nibble) by Byte (nibble)	Full key at a time	Bit by bit
Data complexity		High	Low	Medium
Scenarios	Random delays	×	-	√
	Masking	×	√	√
	White-box attack	√	√	√
	Black-box attack	×	×	√

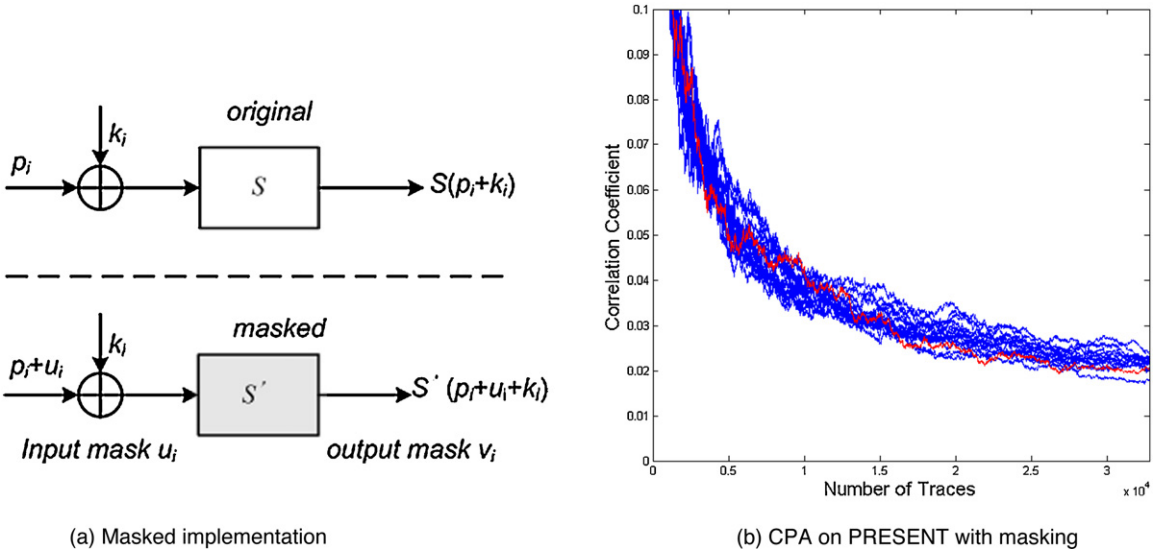


Fig. 8. CPA on protected PRESENT with masking. (a) Masked implementation; (b) CPA on PRESENT with masking.

implementations. The detailed comparisons among SCCA, DPA (CPA) and ASCA are shown in Table 6.

6.2. Limitations, extensions and impacts of HW-SCCA

This paper shows that the HW-SCCA works well on the software implementation of PRESENT on microcontrollers, where the HW of intermediate states can be accurately deduced. Our HW-SCCA might not work when many small random delays are inserted before and after each operation in the first 3 rounds of PRESENT (if the adversary cannot identify the leakage trace of the targeted operation from the delays in the whole power trace), when different masks are utilized in different rounds of PRESENT, or when the byte-oriented computations in either *AK* or *SB* are executed in a random order (Oswald and Schramm, 2005). For hardware implementations, our attacks do not work if the HW of the targeted state cannot be deduced.

The technique of this paper can be easily extended to HW-SCCA on PRESENT with 4-bit implementations or other lightweight block ciphers, especially to some PRESENT-like ciphers such as SmallPresent (Leander, 2010) and EPCBC (Yap et al., 2011).

In the schemes proposed in this paper, the extracted cube size is very small ($\leq$5) and the superpolys are simple (linear or quadric, with the number of key variables $\leq$3). In the black-box SCCA on block ciphers with an iterated structure (e.g., PRESENT), the power pattern for different rounds and operations can be easily identified. The adversary can select some leakage locations and deduce the HWs at those locations. If the HW deduction is accurate, the adversaries can enumerate all the cubes and superpolys, and then test their relations with cube attacks. If the relations hold, the key can be recovered bit by bit. PRESENT can be broken by HW-SCCA under a black-box setting without a prohibitively high data complexity.

The cryptologists should carefully consider HW-SCCA during the cipher design and evaluation phases. Some recent lightweight ciphers took three approaches to mitigate HW-SCCA: (1) increase

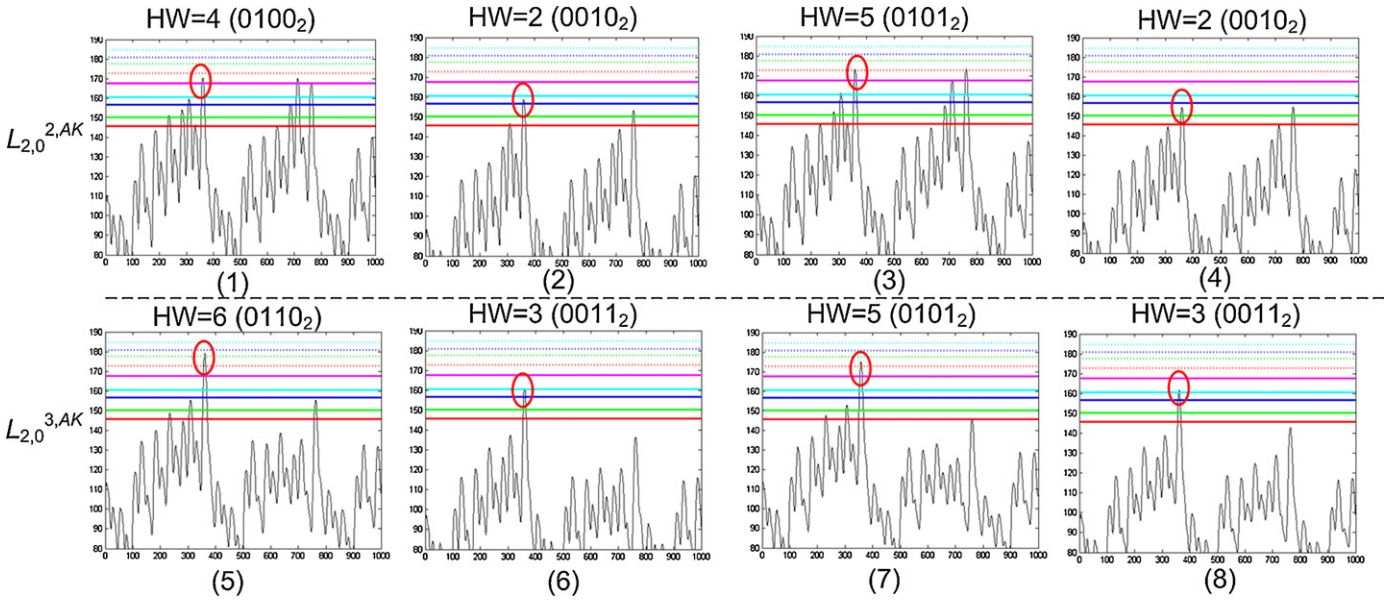


Fig. 9. HW-SCCA on protected PRESENT with masking.

the degree of equations of each output bit, e.g., Klein (Gong et al., 2010), Piccolo (Shibutani et al., 2011); (2) make the linear terms in the equations more complicated (not just simple permutations), e.g., Klein (Gong et al., 2010), Piccolo (Shibutani et al., 2011) and LED (Guo et al., 2011); (3) increase the number of rounds that is required to associate the leakage bit with all the key bits, e.g., LBlock (Wu and Zhang, 2011).

7. Conclusions

This paper proposes several efficient SCCAs on PRESENT based on the single byte Hamming weight leakage model. We start with the basic HW-SCCA on PRESENT-80. Then, two new approaches are proposed to improve SCCA. The first one leverages non-linear equations and the second is an iterative scheme. All the attacks can be easily extended to PRESENT-128. To the best of our knowledge, the combination of all these advanced techniques results in the most efficient SCCA on PRESENT-80/128. The key search space can be reduced to 2^8 for PRESENT-80 with $2^{8.95}$ chosen plaintexts and to 2^7 for PRESENT-128 with $2^{9.78}$ chosen plaintexts.

Meanwhile, the attacks were verified with experiments on PRESENT implemented on an 8-bit microcontroller, instead of with simulations only. The results demonstrate that HW-SCCA has advantages over CPA in some scenarios, especially when some countermeasures such as random delays and masking are integrated. The results also indicate that for implementations where the HW can be deduced accurately, PRESENT may be broken by HW-SCCA under a black-box setting. We hope the work can deepen our understanding of HW-SCCA, which is critical to the design of ciphers and the evaluation of their physical security.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under the grants 60772082, 61173191, 61272491, and US National Science Foundation under the grant CNS-0644188.

Appendix A.

1. Results of HW-SCCA on PRESENT-80

Tables 7–9.

2. Results of HW-SCCA on PRESENT-128

The techniques in Section 4 can be extended to PRESENT-128. The results on PRESENT-128 are summarized in Table 10. As shown in row 1 and row 3 in Table 10, targeting $L_{1,0}^{3,AK}$ and $L_{2,0}^{3,AK}$ can recover 80 key bits with $2^{9.02}$ and $2^{8.99}$ chosen plaintexts, respectively. To recover more key bits, we can target $L_{1,0}^{4,AK}$ and $L_{2,0}^{4,AK}$. As shown in row 2 and row 4 in Table 10, 38 and 41 additional key bits can be recovered with $2^{8.71}$ and $2^{8.52}$ chosen plaintexts respectively. In total, 118 and and 121 key bits can be deduced with $2^{9.87}$ and $2^{9.78}$ chosen plaintexts, respectively. The remaining bits in the master key can be obtained through a brute-force search.

Tables 11–25.

3. Results of HW-SCCA on PRESENT-80 with masking.

Table 7
Cube indexes and superpolys for $N_k(L_{1,0}^{3,AK})$ (linear superploys,using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
1, 2	$1+k_{16}$	33, 34	$1+k_{48}$	5, 7, 8, 9, 10	$1+k_{20}$	37, 38, 40, 41, 42	k_{52}
0, 1	$k_{18}+k_{19}$	32, 33	$k_{50}+k_{51}$	4, 5, 8, 9, 11	$k_{22}+k_{23}$	36, 37, 40, 42, 43	$k_{54}+k_{55}$
0, 2	$k_{17}+k_{19}$	32, 34	$k_{49}+k_{51}$	4, 6, 8, 9, 10	$k_{21}+k_{23}$	36, 38, 40, 41, 42	$k_{53}+k_{55}$
13, 14	$1+k_{28}$	45, 46	$1+k_{60}$	4, 5, 6, 9, 11	$1+k_{24}$	36, 37, 38, 41, 42	k_{56}
12, 13	$k_{30}+k_{31}$	44, 45	$k_{62}+k_{63}$	4, 5, 6, 8, 9	$k_{26}+k_{27}$	36, 37, 38, 40, 41	$k_{58}+k_{59}$
12, 14	$k_{29}+k_{31}$	44, 46	$k_{61}+k_{63}$	4, 5, 6, 8, 10	$k_{25}+k_{27}$	36, 37, 38, 40, 42	$k_{57}+k_{59}$
17, 18	$1+k_{32}$	49, 50	$1+k_{64}$	21, 22, 24, 25, 26	k_{36}	53, 54, 56, 57, 58	k_{68}
16, 17	$k_{34}+k_{35}$	48, 49	$k_{66}+k_{67}$	20, 21, 24, 26, 27	$k_{38}+k_{39}$	52, 53, 56, 58, 59	$k_{70}+k_{71}$
16, 18	$k_{33}+k_{35}$	48, 50	$k_{65}+k_{67}$	20, 22, 24, 25, 26	$k_{37}+k_{39}$	52, 54, 56, 57, 58	$k_{69}+k_{71}$
29, 30	$1+k_{44}$	61, 62	$1+k_{76}$	20, 21, 22, 25, 26	k_{40}	52, 53, 54, 57, 58	k_{72}
28, 29	$k_{46}+k_{47}$	60, 61	$k_{78}+k_{79}$	20, 21, 22, 24, 25	$k_{42}+k_{43}$	52, 53, 55, 56, 57	$k_{74}+k_{75}$
28, 30	$k_{45}+k_{47}$	60, 62	$k_{77}+k_{79}$	20, 21, 22, 24, 26	$k_{41}+k_{43}$	52, 53, 55, 56, 58	$k_{73}+k_{75}$

Table 8
Cube indexes and superpolys for $N_k(L_{1,0}^{3,AK})$ (non-linear superploys,using y_0).

I	K_{SI}	K_d	I	K_{SI}	K_d
5, 8, 9, 10	$1+k_{23}+k_{20}k_{22}+k_{20}k_{23}$	k_{22} or k_{23}	37, 40, 41, 42	$1+k_{55}+k_{52}k_{54}+k_{52}k_{55}$	k_{54} or k_{55}
4, 5, 6, 9	$1+k_{27}+k_{24}k_{26}+k_{24}k_{27}$	k_{26} or k_{27}	36, 37, 38, 41	$1+k_{59}+k_{56}k_{58}+k_{56}k_{59}$	k_{58} or k_{59}
21, 24, 25, 26	$1+k_{39}+k_{36}k_{38}+k_{36}k_{39}$	k_{38} or k_{39}	53, 56, 57, 58	$1+k_{71}+k_{68}k_{70}+k_{68}k_{71}$	k_{70} or k_{71}
20, 21, 22, 25	$1+k_{43}+k_{40}k_{42}+k_{40}k_{43}$	k_{42} or k_{43}	52, 53, 54, 58	$k_{75}+k_{72}k_{73}+k_{72}k_{75}$	k_{73} or k_{75}

Table 9
Cube indexes and superpolys for $N_k(L_{1,0}^{3,AK})$ (iteration,using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
2, 13, 15	$1+k_{17}$	18, 29, 31	$1+k_{33}$	17, 18, 52	k_{49}	48, 53	k_{65}
1, 3, 14	$1+k_{29}$	17, 19, 30	$1+k_{45}$	32, 37	k_{61}	49, 51, 62	$1+k_{77}$

Table 10
Results of HW-SCCAs on PRESENT-128.

Leakage byte	Refer	N_{cp}	N_{rk}
$N_k(L_{1,0}^{3,AK})$	Tables 11–14	$2^{9.02}$	80 bits
$N_k(L_{1,0}^{3,AK}, L_{1,0}^{4,AK})$	Tables 11–15	$2^{9.02}+2^{8.71}=2^{9.87}$	118 bits
$N_k(L_{2,0}^{3,AK})$	Tables 16–18	$2^{8.99}$	80 bits
$N_k(L_{2,0}^{3,AK}, L_{2,0}^{4,AK})$	Table 16–19	$2^{8.99}+2^{8.52}=2^{9.78}$	121 bits

Table 11
Cube indexes and superpolys for $N_k(L_{1,0}^{3,AK})$ (linear equations,using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
1, 2	$1+k_{64}$	33, 34	$1+k_{96}$	5, 7, 8, 9, 10	$1+k_{68}$	37, 39, 40, 41, 42	k_{100}
0, 1	$k_{66}+k_{67}$	32, 33	$k_{98}+k_{99}$	4, 5, 8, 10, 11	$k_{70}+k_{71}$	36, 37, 40, 42, 43	$k_{102}+k_{103}$
0, 2	$k_{65}+k_{67}$	32, 34	$k_{97}+k_{99}$	4, 6, 8, 9, 10	$k_{69}+k_{71}$	36, 38, 40, 41, 42	$k_{101}+k_{103}$
13, 14	$1+k_{76}$	45, 46	$1+k_{108}$	4, 5, 6, 9, 11	$1+k_{72}$	36, 37, 38, 41, 43	k_{104}
12, 13	$k_{78}+k_{79}$	44, 45	$k_{110}+k_{111}$	4, 5, 7, 8, 9	$k_{74}+k_{75}$	36, 37, 38, 40, 41	$k_{106}+k_{107}$
12, 14	$k_{77}+k_{79}$	44, 46	$k_{109}+k_{111}$	4, 5, 7, 8, 10	$k_{73}+k_{75}$	36, 37, 38, 40, 42	$k_{105}+k_{107}$
17, 18	$1+k_{80}$	49, 50	$1+k_{112}$	21, 23, 24, 25, 26	k_{84}	53, 55, 56, 57, 58	k_{116}
16, 17	$k_{82}+k_{83}$	48, 49	$k_{114}+k_{115}$	20, 21, 24, 26, 27	$k_{86}+k_{87}$	52, 53, 56, 58, 59	$k_{118}+k_{119}$
16, 18	$k_{81}+k_{83}$	48, 50	$k_{113}+k_{115}$	20, 22, 24, 25, 26	$k_{85}+k_{87}$	52, 54, 56, 57, 58	$k_{117}+k_{119}$
29, 30	$1+k_{92}$	61, 62	$1+k_{124}$	20, 21, 22, 25, 27	k_{88}	52, 53, 54, 57, 59	k_{120}
28, 29	$k_{94}+k_{95}$	60, 61	$k_{126}+k_{127}$	20, 21, 22, 24, 25	$k_{90}+k_{91}$	52, 53, 54, 56, 57	$k_{122}+k_{123}$
28, 30	$k_{93}+k_{95}$	60, 62	$k_{125}+k_{127}$	20, 21, 22, 24, 26	$k_{89}+k_{91}$	52, 53, 54, 56, 58	$k_{121}+k_{123}$

Table 12
Cube indexes and superpolys for $N_k(L_{1,0}^{3,AK})$ (non-linear equations, using y_0).

I	K_{SI}	D	I	K_{SI}	D
6, 8, 9, 10	$k_{71}+k_{68}k_{69}+k_{68}k_{71}$	k_{69} or k_{71}	38, 40, 41, 42	$k_{103}+k_{100}k_{101}+k_{100}k_{103}$	k_{101} or k_{103}
4, 5, 6, 9	$1+k_{75}+k_{72}k_{74}+k_{72}k_{75}$	k_{74} or k_{75}	36, 37, 38, 41	$1+k_{107}+k_{104}k_{106}+k_{104}k_{107}$	k_{106} or k_{107}
22, 24, 25, 26	$k_{87}+k_{84}k_{85}+k_{84}k_{87}$	k_{85} or k_{87}	53, 56, 57, 58	$1+k_{119}+k_{116}k_{118}+k_{116}k_{119}$	k_{118} or k_{119}
20, 21, 22, 25	$1+k_{91}+k_{88}k_{90}+k_{88}k_{91}$	k_{90} or k_{91}	52, 54, 55, 58	$k_{123}+k_{120}k_{121}+k_{120}k_{123}$	k_{121} or k_{123}

Table 13
Cube indexes and superpolys for $N_k(L_{1,0}^{3,AK})$ (1st iterative SCCA, using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
8	$1+k_4$	40	$1+k_{12}$	9	$1+k_{20}$	41	$1+k_{28}$	24	$1+k_8$	56	$1+k_{16}$	25	k_{24}	57	k_{32}
4	k_5	36	k_{13}	5,	k_{21}	37	k_{29}	20	k_9	52	k_{17}	21	k_{25}	53	k_{33}

Table 14
Cube indexes and superpolys for $N_k(L_{1,0}^{3,AK})$ (2nd iterative SCCA, using y_1).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
2, 13, 14	k_{67}	18, 29, 30	k_{83}	34, 45, 46	k_{99}	50, 61, 62	k_{115}
1, 2, 14	k_{79}	17, 18, 30	k_{95}	33, 34, 46	k_{111}	49, 50, 62	k_{127}

Table 15
Cube indexes and superpolys for $N_k(L_{1,0}^{4,AK})$ (3rd iterative SCCA, using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}
1, 2	$1+k_3$	44, 45	$k_{49}+k_{50}$	20, 21, 22, 24, 25	$k_{29}+k_{30}$
0, 1	k_5+k_6	44, 46	$k_{48}+k_{50}$	20, 21, 22, 24, 26	$k_{28}+k_{30}$
0, 2	k_4+k_6	49, 50	$1+k_{51}$	37, 39, 40, 41, 42	$1+k_{39}$
13, 14	$1+k_{15}$	48, 49	$k_{53}+k_{54}$	36, 37, 40, 42, 43	$k_{41}+k_{42}$
12, 13	$k_{17}+k_{18}$	48, 50	$k_{52}+k_{54}$	36, 37, 40, 42, 43	$k_{40}+k_{42}$
12, 14	$k_{16}+k_{18}$	61, 62	$1+k_{63}+k_{65}+k_{66}+k_{64}k_{65}$	38, 40, 41, 42	$k_{42}+k_{39}k_{40}+k_{39}k_{42}$
17, 18	$1+k_{19}$	5, 7, 8, 9, 10	$1+k_7$	36, 37, 38, 41, 43	$1+k_{43}$
16, 17	$k_{21}+k_{22}$	4, 5, 8, 10, 11	k_9+k_{10}	36, 37, 38, 40, 41	$k_{45}+k_{46}$
16, 18	$k_{20}+k_{22}$	4, 6, 8, 9, 10	k_8+k_{10}	36, 37, 38, 40, 42	$k_{44}+k_{46}$
29, 30	$1+k_{31}$	4, 5, 6, 9, 11	$1+k_{11}$	36, 37, 38, 42	$k_{46}+k_{43}k_{44}+k_{43}k_{46}$
28, 29	$k_{33}+k_{34}$	4, 5, 6, 8, 9	$k_{13}+k_{14}$	53, 55, 56, 57, 58	$1+k_{55}$
28, 30	$k_{32}+k_{34}$	4, 5, 6, 8, 10	$k_{12}+k_{14}$	52, 53, 56, 58, 59	$k_{57}+k_{58}$
33, 34	$1+k_{35}$	21, 23, 24, 25, 26	$1+k_{23}$	52, 54, 56, 57, 58	$k_{56}+k_{58}$
32, 33	$k_{37}+k_{38}$	20, 21, 24, 26, 27	$k_{25}+k_{26}$	54, 56, 57, 58	$k_{58}+k_{55}k_{56}+k_{55}k_{58}$
32, 34	$k_{36}+k_{38}$	20, 22, 24, 25, 26	$k_{24}+k_{26}$		
45, 46	$1+k_{47}$	20, 21, 22, 25, 27	$1+k_{27}$		

Table 16
Cube indexes and superpolys for $N_k(L_{2,0}^{3,AK})$ (linear equations,using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
1, 2	$1+k_{64}$	32, 33	$1+k_{98}$	4, 5, 6, 9, 10	$1+k_{72}$	36, 37, 40, 41, 43	$1+k_{102}$
0, 2	k_{65}	45, 46	$1+k_{108}$	4, 5, 7, 8, 10	k_{73}	36, 37, 40, 41, 42	$k_{102}+k_{103}$
0, 1	$1+k_{66}$	44, 46	k_{109}	4, 5, 7, 8, 9	$1+k_{74}$	36, 37, 38, 41, 42	$1+k_{104}$
13, 14	$1+k_{76}$	44, 45	$1+k_{110}$	4, 5, 6, 8, 9	$k_{74}+k_{75}$	36, 37, 39, 40, 42	k_{105}
12, 14	k_{77}	49, 50	$1+k_{112}$	21, 22, 24, 25, 26	$1+k_{84}$	36, 37, 39, 40, 41	$1+k_{106}$
12, 13	$1+k_{78}$	48, 50	k_{113}	20, 22, 24, 25, 27	k_{85}	36, 37, 38, 40, 41	$k_{106}+k_{107}$
17, 18	$1+k_{80}$	48, 49	$1+k_{114}$	20, 21, 24, 25, 27	$1+k_{86}$	53, 54, 56, 57, 58	$1+k_{116}$
16, 18	k_{81}	61, 62	$1+k_{124}$	20, 21, 24, 25, 26	$k_{86}+k_{87}$	52, 54, 56, 57, 59	k_{117}
16, 17	$1+k_{82}$	60, 62	k_{125}	20, 21, 23, 25, 26	$1+k_{88}$	52, 53, 56, 57, 59	$1+k_{118}$
29, 30	$1+k_{92}$	60, 61	$1+k_{126}$	20, 22, 23, 24, 26	k_{89}	52, 53, 56, 57, 58	$k_{118}+k_{119}$
28, 30	k_{93}	5, 6, 8, 9, 10	$1+k_{68}$	20, 21, 23, 24, 25	$1+k_{90}$	52, 53, 54, 57, 58	$1+k_{120}$
28, 29	$1+k_{94}$	4, 6, 8, 9, 11	k_{69}	20, 21, 22, 24, 25	$k_{90}+k_{91}$	52, 53, 55, 56, 58	k_{121}
33, 34	$1+k_{96}$	4, 5, 8, 9, 11	$1+k_{70}$	37, 38, 40, 41, 42	$1+k_{100}$	52, 53, 55, 56, 57	$1+k_{122}$
32, 34	k_{97}	4, 5, 8, 9, 10	$k_{70}+k_{71}$	36, 38, 40, 41, 43	k_{101}	52, 53, 54, 56, 57	$k_{122}+k_{123}$

Table 17
Cube indexes and superpolys for $N_k(L_{2,0}^{3,AK})$ (non-linear equations, using y_0).

I	K_{SI}	DI	K_{SI}	D
1	$k_{64}+k_{67}+k_{66}+k_{64}k_{66}$	$k_{67}33$	$k_{96}+k_{99}+k_{98}+k_{96}k_{98}$	k_{99}
13	$k_{76}+k_{79}+k_{78}+k_{76}k_{78}$	$k_{79}45$	$k_{108}+k_{111}+k_{110}+k_{108}k_{110}$	k_{111}
17	$k_{80}+k_{83}+k_{82}+k_{80}k_{82}$	$k_{83}49$	$k_{112}+k_{115}+k_{114}+k_{112}k_{114}$	k_{115}
29	$k_{92}+k_{95}+k_{94}+k_{92}k_{94}$	$k_{95}61$	$k_{124}+k_{127}+k_{126}+k_{124}k_{126}$	k_{127}

Table 18
Cube indexes and superpolys for $N_k(L_{2,0}^{3,AK})$ (1st iterative SCCA, using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}	I	K_{SI}
34	$1+k_{36}$	38	$1+k_{40}$	42	$1+k_{44}$	46	$1+k_{48}$	50	$1+k_{52}$	54	$1+k_{56}$
33	k_{37}	37	k_{41}	41	k_{45}	45	k_{49}	49	k_{53}	53	k_{57}

Table 19
Cube indexes and superpolys for $N_k(L_{2,0}^{4,AK})$ (2nd iterative SCCA, using y_0).

I	K_{SI}	I	K_{SI}	I	K_{SI}
1, 2	$1+k_3$	29	$1+k_{31}+k_{34}+k_{33}+k_{31}k_{33}$	4, 5, 6, 8, 9	$k_{13}+k_{14}$
0, 2	k_4	33, 34	$1+k_{35}$	22, 23, 25, 27	k_{23}
0, 1	$1+k_5$	33	$1+k_{35}+k_{38}+k_{37}+k_{35}k_{37}$	20, 22, 24, 25, 27	k_{24}
0	$1+k_4+k_6+k_4k_5$	45, 46	$1+k_{47}$	20, 21, 24, 25, 27	$1+k_{25}$
13, 14	$1+k_{15}$	45	$1+k_{47}+k_{50}+k_{49}+k_{47}k_{49}$	20, 21, 24, 25, 26	$k_{25}+k_{26}$
12, 14	k_{16}	49, 50	$1+k_{51}$	21, 23, 26, 27	k_{27}
12, 13	$1+k_{17}$	49	$1+k_{51}+k_{54}+k_{53}+k_{51}k_{53}$	20, 22, 23, 24, 26	k_{28}
13	$1+k_{15}+k_{18}+k_{17}+k_{15}k_{17}$	61, 62	$1+k_{63}+k_{65}+k_{66}+k_{64}k_{65}$	20, 21, 23, 24, 25	$1+k_{29}$
17, 18	$1+k_{19}$	6, 7, 9, 11	k_7	20, 21, 22, 24, 25	$k_{29}+k_{30}$
16, 18	k_{20}	7, 8, 9, 11	k_7+k_8	38, 39, 41, 43	k_{39}
16, 17	$1+k_{21}$	4, 5, 8, 9, 11	$1+k_9$	36, 37, 40, 41, 42	$k_{41}+k_{42}$
17	$1+k_{19}+k_{22}+k_{21}+k_{19}k_{21}$	4, 5, 8, 9, 10	k_9+k_{10}	37, 39, 42, 43	k_{43}
29, 30	$1+k_{31}$	5, 7, 10, 11	k_{11}	36, 37, 38, 40, 41	$k_{45}+k_{46}$
28, 30	k_{32}	4, 5, 7, 8, 10	k_{12}	53, 54, 56, 57, 58	$1+k_{55}$
28, 29	$1+k_{33}$	4, 5, 7, 8, 9	$1+k_{13}$	52, 53, 56, 57, 58	$k_{57}+k_{58}$

Table 20
Cube indexes and linear superpolys on $L_{1,0}^{2,AK} \oplus L_{1,0}^{3,AK}$.

I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$
1, 2	k_{16}	33, 34	$1+k_{48}$	5, 7, 8, 9, 10	$1+k_{20}$	37, 38, 40, 41, 42	k_{52}
0, 1	$k_{18}+k_{19}$	32, 33	$k_{50}+k_{51}$	4, 5, 8, 9, 11	$k_{22}+k_{23}$	36, 37, 40, 42, 43	$k_{54}+k_{55}$
0, 2	$k_{17}+k_{19}$	32, 34	$k_{49}+k_{51}$	4, 6, 8, 9, 10	$k_{21}+k_{23}$	36, 38, 40, 41, 42	$k_{53}+k_{55}$
13, 14	k_{28}	45, 46	$1+k_{60}$	4, 5, 6, 9, 11	$1+k_{24}$	36, 37, 38, 41, 42	k_{56}
12, 13	$k_{30}+k_{31}$	44, 45	$k_{62}+k_{63}$	4, 5, 6, 8, 9	$k_{26}+k_{27}$	36, 37, 38, 40, 41	$k_{58}+k_{59}$
12, 14	$k_{29}+k_{31}$	44, 46	$k_{61}+k_{63}$	4, 5, 6, 8, 10	$k_{25}+k_{27}$	36, 37, 38, 40, 42	$k_{57}+k_{59}$
17, 18	k_{32}	49, 50	$1+k_{64}$	21, 22, 24, 25, 26	k_{36}	53, 54, 56, 57, 58	k_{68}
16, 17	$k_{34}+k_{35}$	48, 49	$k_{66}+k_{67}$	20, 21, 24, 26, 27	$k_{38}+k_{39}$	52, 53, 56, 58, 59	$k_{70}+k_{71}$
16, 18	$k_{33}+k_{35}$	48, 50	$k_{65}+k_{67}$	20, 22, 24, 25, 26	$k_{37}+k_{39}$	52, 54, 56, 57, 58	$k_{69}+k_{71}$
29, 30	k_{44}	61, 62	$1+k_{76}$	20, 21, 22, 25, 26	k_{40}	52, 53, 54, 57, 58	k_{72}
28, 29	$k_{46}+k_{47}$	60, 61	$k_{78}+k_{79}$	20, 21, 22, 24, 25	$k_{42}+k_{43}$	52, 53, 55, 56, 57	$k_{74}+k_{75}$
28, 30	$k_{45}+k_{47}$	60, 62	$k_{77}+k_{79}$	20, 21, 22, 24, 26	$k_{41}+k_{43}$	52, 53, 55, 56, 58	$k_{73}+k_{75}$

Table 21
Cube indexes and non-linear superpolys on $L_{1,0}^{2,AK} \oplus L_{1,0}^{3,AK}$.

I	$p_{S(I)}$	K_d	I	$p_{S(I)}$	K_d
5, 8, 9, 10	$1+k_{23}+k_{20}k_{22}+k_{20}k_{23}$	k_{22} or k_{23}	37, 40, 41, 42	$1+k_{55}+k_{52}k_{54}+k_{52}k_{55}$	k_{54} or k_{55}
4, 5, 6, 9	$1+k_{27}+k_{24}k_{26}+k_{24}k_{27}$	k_{26} or k_{27}	36, 37, 38, 41	$1+k_{59}+k_{56}k_{58}+k_{56}k_{59}$	k_{58} or k_{59}
21, 24, 25, 26	$1+k_{39}+k_{36}k_{38}+k_{36}k_{39}$	k_{38} or k_{39}	53, 56, 57, 58	$1+k_{71}+k_{68}k_{70}+k_{68}k_{71}$	k_{70} or k_{71}
20, 21, 22, 25	$1+k_{43}+k_{40}k_{42}+k_{40}k_{43}$	k_{42} or k_{43}	52, 53, 54, 58	$k_{75}+k_{72}k_{73}+k_{72}k_{75}$	k_{73} or k_{75}

Table 22
Cube indexes and superpolys on $L_{1,0}^{2,AK} \oplus L_{1,0}^{3,AK}$ (iterative SCCA).

I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$
1	k_{19}	18	$1+k_{35}$	52	k_{49}	53	k_{65}
14	$1+k_{31}$	30	$1+k_{47}$	37	k_{61}		

Table 23
Cube indexes and linear superpolys on $L_{2,0}^{2,AK} \oplus L_{2,0}^{3,AK}$.

I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$
1, 2	$1+k_{16}$	32, 33	$1+k_{50}$	4, 5, 6, 9, 10	$1+k_{24}$	36, 37, 40, 41, 43	$1+k_{54}$
0, 2	k_{17}	45, 46	k_{60}	4, 5, 7, 8, 10	k_{25}	36, 37, 40, 41, 42	$k_{54}+k_{55}$
0, 1	$1+k_{18}$	44, 46	k_{61}	4, 5, 7, 8, 9	$1+k_{26}$	36, 37, 38, 41, 42	$1+k_{56}$
13, 14	$1+k_{28}$	44, 45	$1+k_{62}$	4, 5, 6, 8, 9	$k_{26}+k_{27}$	36, 37, 39, 40, 42	k_{57}
12, 14	k_{29}	49, 50	k_{64}	21, 22, 24, 25, 26	$1+k_{36}$	36, 37, 39, 40, 41	$1+k_{58}$
12, 13	$1+k_{30}$	48, 50	k_{65}	20, 22, 24, 25, 27	k_{37}	36, 37, 38, 40, 41	$k_{58}+k_{59}$
17, 18	$1+k_{32}$	48, 49	$1+k_{66}$	20, 21, 24, 25, 27	$1+k_{38}$	53, 54, 56, 57, 58	$1+k_{68}$
16, 18	k_{33}	61, 62	k_{76}	20, 21, 24, 25, 26	$k_{38}+k_{39}$	52, 54, 56, 57, 59	k_{69}
16, 17	$1+k_{34}$	60, 62	k_{77}	20, 21, 22, 25, 26	$1+k_{40}$	52, 53, 56, 57, 59	$1+k_{70}$
29, 30	$1+k_{44}$	60, 61	$1+k_{78}$	20, 21, 23, 24, 26	k_{41}	52, 53, 56, 57, 58	$k_{70}+k_{71}$
28, 30	k_{45}	5, 6, 8, 9, 10	$1+k_{20}$	20, 21, 23, 24, 25	$1+k_{42}$	52, 53, 54, 57, 58	$1+k_{72}$
28, 29	$1+k_{46}$	4, 6, 8, 9, 11	k_{21}	20, 21, 22, 24, 25	$k_{42}+k_{43}$	52, 53, 55, 56, 58	k_{73}
33, 34	k_{48}	4, 5, 8, 9, 11	$1+k_{22}$	37, 38, 40, 41, 42	$1+k_{52}$	52, 53, 55, 56, 57	$1+k_{74}$
32, 34	k_{49}	4, 5, 8, 9, 10	$k_{22}+k_{23}$	36, 38, 40, 41, 43	k_{53}	52, 53, 54, 56, 57	$k_{74}+k_{75}$

Table 24
Cube indexes and non-linear superpolys on $L_{2,0}^{2,AK} \oplus L_{2,0}^{3,AK}$.

I	$p_{S(I)}$	K_d	I	$p_{S(I)}$	K_d
1	$1+k_{16}+k_{19}+k_{18}+k_{16}k_{18}$	k_{19}	33	$1+k_{48}+k_{51}+k_{48}k_{50}$	k_{51}
13	$1+k_{28}+k_{31}+k_{30}+k_{28}k_{30}$	k_{31}	45	$1+k_{60}+k_{63}+k_{60}k_{62}$	k_{63}
17	$1+k_{32}+k_{35}+k_{34}+k_{32}k_{34}$	k_{35}	49	$1+k_{64}+k_{67}+k_{64}k_{66}$	k_{67}
29	$1+k_{44}+k_{47}+k_{46}+k_{44}k_{46}$	k_{47}	61	$1+k_{76}+k_{79}+k_{76}k_{78}$	k_{79}

Table 25
Cube indexes and superpolys on $L_{2,0}^{2,AK} \oplus L_{2,0}^{3,AK}$ (iterative SCCA).

I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$	I	$p_{S(I)}$
46	$1+k_0$	50	$1+k_4$	54	$1+k_8$	58	$1+k_{12}$
45	k_1	49	k_5	53	k_9	57	k_{13}

References

Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J., 2010. On the Security of NOEKEON against side channel cube attacks. In: Proceedings of the 5th Information Security Practice and Experience Conference. Future Conference - ISPEC 2010, LNCS, vol. 6047, pp. 45–55.

Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J., 2011. 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. 296–305.

Akkar, M., Giraud, C., 2001. An implementation of DES and AES, secure against some attacks. In: Cryptographic Hardware and Embedded Systems – CHES 2001, LNCS, vol. 2162, pp. 309–318.

Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A., 2009. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Fast Software Encryption - FSE 2009. LNCS, vol. 5665, pp. 1–22.

Bard, G.V., Courtois, N.T., Nakahara, J., Sepehrdad, P., Zhang, B., 2010. Algebraic, AIDA/cube and side channel analysis of KATAN family of block ciphers. In: Progress in Cryptology-indocryt, 2010, LNCS, vol. 6498, pp. 176–196.

Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., et al., 2007. PRESENT: an ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems – CHES 2007. LNCS, vol. 4727, pp. 450–466.

Brier, E., Clavier, C., Olivier, F., 2004. Correlation power analysis with a leakage model. In: Cryptographic Hardware and Embedded Systems – CHES 2004, LNCS, vol. 3156, pp. 16–29.

Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P., 1999. Towards sound approaches to counteract power-analysis attacks. In: Advances in Cryptology – CRYPTO 1999, LNCS, vol. 1666, pp. 398–412.

Coron, J.S., Kizhvatov, I., 2009. An efficient method for random delay generation in embedded software. In: Cryptographic Hardware and Embedded Systems – CHES 2009, LNCS, vol. 5747, pp. 156–170.

Coron, J.S., Kizhvatov, I., 2010. Analysis and improvement of the random delay countermeasure of CHES 2009. In: Cryptographic Hardware and Embedded Systems – CHES 2010, LNCS, vol. 6225, pp. 95–109.

Dinur, I., Shamir, A. Cube attacks on tweakable black-box polynomials. Cryptology ePrint Archive, http://eprint.iacr.org/2008/385.pdf

Dinur, I., Shamir, A., 2009. Cube attacks on tweakable black-box polynomials. In: Advances in Cryptology – EUROCRYT, 2009. LNCS, vol. 5479, pp. 278–299.

Dinur, I., Shamir, A. Side channel cube attacks on block ciphers. Cryptology ePrint Archive, http://eprint.iacr.org/2009/127.pdf

Een, N., Sorensson, N., 2005. MiniSat – a SAT solver with conflict-clause minimization. In: Proceedings of SAT 05.

Fan, X., Gong, G., 2011. On the security of Hummingbird-2 against side channel cube attacks. In: Proceedings of WEWoRC 2011, pp. 100–104.

Gong, Z., Nikova, S.I., Law, Y.W. KLEIN: a new family of lightweight block ciphers. Technical Report TR-CTIT-10-33, Centre for Telematics and Information Technology, University of Twente, Enschede, 2010.

Goubin, L., Patarin, J., 1999. DES and differential power analysis (the "duplication" method). In: Cryptographic Hardware and Embedded Systems – CHES 1999, LNCS, vol. 1717, pp. 158–172.

Guo, J., Peyrin, T., Poschmann, A., Robshaw, M., 2011. The LED block cipher. In: Cryptographic Hardware and Embedded Systems – CHES 2011, LNCS, vol. 6917, pp. 326–341.

Klose, D. PRESENT implementation, http://www.lightweightcrypto.org/implementations.php, 2011.

Kocher, P.C., 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Advances in Cryptology CRYPTO 1996, LNCS, vol. 1109, pp. 104–113.

Kocher, P.C., Jaffe, J., Jun, B., 1999. Differential power analysis. In: Advances in Cryptology CRYPTO 1999, LNCS, vol. 1666, pp. 388–397.

Kocher, P.C., Jaffe, J., Jun, B., Rohtagi, P., 2011. Introduction to differential power analysis. Journal of Cryptographic Engineering 1, 5–27.

Lai, X., 1994. Higher order derivatives and differential cryptanalysis. Communications and Cryptography. Kluwer Academic Publishers, pp. 227–233.

Leander, G., 2010. Small scale variants of the block cipher PRESENT, Cryptology ePrint Archive, http://eprint.iacr.org/2010/143.pdf

Mangard, S., Oswald, E., Popp, T., 2007. Power Analysis Attacks. Springer, Berlin Heidelberg.

Mayer-Sommer, R., 2000. Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In: Cryptographic Hardware and Embedded Systems – CHES 2000, LNCS, vol. 1965, pp. 78–92.

Oswald, E., Schramm, K., 2005. An efficient masking scheme for AES software implementations. In: Proceedings of WISA 2005, LNCS, vol. 3786, pp. 292–305.

Princeton University zChaff. http://www.princeton.edu/chaff/zchaff.html, 2007.

Quisquater, J.J., Samyde, D., 2000. A new tool for NON-intrusive ANALYSIS of smart cards based on electro-magnetic emissions: the SEMA and DEMA methods. In: Eurocrypt Rump Session.

Renauld, M., Standaert, F.-X., 2009. Algebraic side-channel attacks. In: Proceedings of INSCRY, 2009, LNCS, vol. 6151, pp. 393–410.

Renauld, M., Standaert, F., Veyrat-Charvillon, N., 2009. Algebraic side-channel attacks on the AES: why time also matters in DPA. In: Cryptographic Hardware and Embedded Systems – CHES 2009, LNCS, vol. 5747, pp. 97–111.

Shibutani, K., Isobe, T., Hiwatari, H., et al., 2011. Piccolo: an ultra-lightweight block-cipher. In: Cryptographic Hardware and Embedded Systems – CHES 2011, LNCS, vol. 6917, pp. 342–357.

Soos, M., Nohl, K., Castelluccia, C., 2011. Extending SAT solvers to cryptographic problems. In: Proceedings of SAT 09, LNCS 5584, pp. 244–257.

Standaert, F.-X., BernaÖrs, S., Preneel, B., 2004. Power analysis of an FPGA implementation of Rijndael: is pipelining a DPA countermeasure? In: Cryptographic Hardware and Embedded Systems – CHES 2004, LNCS, vol. 3156, pp. 30–44.

Vielhaber, M. 2007. Breaking ONE.FIVIUM by AIDA – an Algebraic IV Differential Attack. Cryptology ePrint Archive, http://eprint.iacr.org/2007/413.pdf

Wu, W., Zhang, L., 2011. LBlock: a lightweight block cipher. In: Proceedings of the 9th International Conference on Applied Cryptography and Network Security – ACNS 2011, LNCS, vol. 6715, pp. 327–C344.

Yang, L., Wang, M., Qiao, S., 2009. Side Channel cube attack on PRESENT. In: Proceedings of the 8th International Conference on Cryptology and Network Security – CANS 2009, LNCS, vol. 5888, pp. 379–391.

Yap, H., Khoo, K., Poschmann, A., Henricksen, M., 2011. EPCBC – A block cipher suitable for electronic product code encryption. In: Proceedings of the 10th International Conference on Cryptology And Network Security – CANS 2011, LNCS, vol. 7092, pp. 67–97.

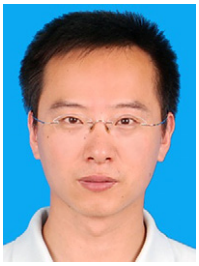
Zhang, A., Lim, C.-W., Khoo, K., Lei, W., Pieprzyk, J. 2009. Extensions of the cube attack based on low degree annihilators. Cryptology ePrint Archive, http://eprint.iacr.org/2009/049.pdf

Zhao,X., Wang, T., Guo, S. 2011. Improved side channel cube attacks on PRESENT. Cryptology ePrint Archive, http://eprint.iacr.org/2011/165.pdf

Zhao, X.J., Zhang, F., Guo, S.Z., et al., 2012. MDASCA: an enhanced algebraic side-channel attack for error tolerance and new leakage model exploitation. In: Constructive Side-Channel Analysis and Secure Design: COSADE 2012, LNCS, vol. 7275, pp. 231–248.

Xinjie Zhao received his B.E. degree and M.E. degree in Department of Information Engineering, Ordnance Engineering College in 2006 and 2009, respectively. He is currently a Ph.D. student in Department of Information Engineering, Ordnance Engineering College. His main research interest includes side channel analysis, fault analysis and combined analysis of block ciphers. He won the best paper in Darmstadt - the 3rd International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2012).

Shize Guo was born in 1964. He received his Ph.D. degree in Harbin Institute of Technology, Harbin, China in 1989. He is currently a researcher in Institute of North Electronic Equipment. His main research interest includes information security and cryptography..

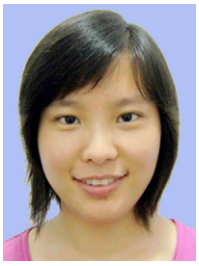


Fan Zhang was born in 1978. He received his Ph.D. degree in Department of Computer Science and Engineering from University of Connecticut in 2012. His research interests include side channel analysis and fault analysis in cryptography, computer architecture, security in wireless sensor network, etc.

Tao Wang was born in 1964. He received his master's degree in computer application from Ordnance Engineering College in 1990 and Ph.D. degree in computer application from Institute of Computing Technology Chinese Academy of Sciences in 1996. He is currently a Professor in Ordnance Engineering College. His research interests include information security and cryptography.

Zhijie Shi is an Assistant Professor of Computer Science and Engineering at the University of Connecticut (UConn). He received his Ph.D. degree from Princeton University in 2004 and his M.S. and B.S. degrees from Tsinghua University, China, in 1996 and 1992, respectively. Professor Shi received US National Science Foundation CAREER award in 2006. His research interests include hardware mechanisms for secure and reliable computing, side-channel attacks and countermeasures, primitives for ultra-efficient cipher designs, sensor network security, application specific instruction-set instruction processor(ASIP) and embedded system designs..

Huiying Liu was born in 1984. He received his master's degree in computer software and theory from Department of Computer Engineering, Ordnance Engineering College in 2010. He is currently a Ph.D. student in Department of Information Engineering, Ordnance Engineering College. His main research interest includes algebraic side channel analysis, and side-channel cube analysis of block ciphers.



Keke Ji was born in 1988. She is currently a Ms.D. student in Department of Information Engineering, Ordnance Engineering College. Her main research interest includes algebraic side channel analysis, and side-channel cube analysis of block ciphers.

Jing Huang was born in 1983. She is currently a Ms.D. student in Department of Computer Science and Technology, School of Information Science and Technology, Tsinghua University. Her main research interest includes: automated test and control of instrument, communication theory and information security.