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