Double Sieve Collision Attack Based on Bitwise Detection

Yanting Ren, Liji Wu and An Wang
Institute of Microelectronics, Tsinghua University
Beijing 100084 – China
[e-mail: ryt10@mails.tsinghua.edu.cn; lijiwu@mail.tsinghua.edu.cn; wanganl@tsinghua.edu.cn]
*Corresponding author: An Wang

Received April 1, 2014; accepted December 13, 2014; published January 31, 2015

Abstract

Advanced Encryption Standard (AES) is widely used for protecting wireless sensor network (WSN). At the Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2012, Gérard *et al*. proposed an optimized collision attack and break a practical implementation of AES. However, the attack needs at least 256 averaged power traces and has a high computational complexity because of its byte wise operation. In this paper, we propose a novel double sieve collision attack based on bitwise collision detection, and an improved version with an error-tolerant mechanism. Practical attacks are successfully conducted on a software implementation of AES in a low-power chip which can be used in wireless sensor node. Simulation results show that our attack needs 90% less time than the work published by Gérard *et al*. to reach a success rate of 0.9.

Keywords: AES, wireless sensor network, collision attack, power analysis, side-channel attack

This research was supported by the Major Program "Core of Electronic Devices, High-End General Chips, and Basis of Software Products" of the Ministry of Industry and Information Technology of China (Nos. 2014ZX01032205, 2014ZX01032401-001-Z05, 2014ZX01032401-001-Z07), the Foundation of Science and Technology on Information Assurance Laboratory (Nos. KJ-13-101, KJ-14-006), the National Natural Science Foundation of China (No. 61402252), and "12th Five-Year Plan" The National Development Foundation for Cryptological Research (No. MMJJ201401009).

http://dx.doi.org/10.3837/tiis.2015.01.017 **ISSN : 1976-7277**

1. Introduction

Wireless sensor networks (WSN) provide promising solutions in a wide range of applications, such as military, healthy care, industrial monitoring, traget localization and tracing. Sensor nodes that consist the WSN are usually placed in protentially hostile enviremont and face various kinds of challenges [1]-[3]. For the needs of security, cryptographic algorithms are used to implement authentications and encrypted communications in WSN. Advanced Encryption Standard (AES), adopted by USA government in 2002 [4], is ideal for the resources-constrained sensor nodes because of its high speed and low cost. As a standard encryption algorithm in wireless communication [5], AES is widely used in current WSN platforms.

However, wireless sensor nodes are vulnerable to side-channel attacks. Since proposed by Kocher *et al*. [6], side-channel attacks have been known as efficient to recover the key by eavesdropping the physical information (e.g., power consumption, electro-magnetic radiation) leaked by target devices [7]-[10]. These attacks are more efficient than traditional cryptanalysis. They do not interrupt operations of the target device, so they can be conducted stealthily on wireless sensor nodes without being detected [11],

Side-channel collision attack, as a combination of side-channel attack and cryptanalysis, was proposed in 2003 by Schramm *et al*. against DES [12], and was applied to AES [13] soon after that. Improved collision attacks were presented subsequently [14]-[17]. Most collision attacks are highly sensitive to errors, namely false positives of collision detections [18], which usually happen when the noise level is high. Gérard *et al*. [18] introduced Low Density Parity Check (LDPC) decoding approach to deal with errors, which made their work more efficient than previous methods. However, there are two problems for LDPC method. First, the computational complexity of the offline stage is high, due to its framework. Second, the online stage (power acquisition stage) is very time-consuming, because all the acquired power traces need to be saved.

In this paper, we propose an efficient and error robust collision attack. The new framework of our approach is based on a double sieve model, which ensures the efficiency and success rate of attacks. A bitwise collision detection method is proposed, which greatly reduces the time for online stage by reducing the number of saved traces. The computational complexity of the framework is low, so the key can be recovered very fast. Practical attacks on AES and experimental results show that our approach is more efficient than previous methods.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the notations and recall previous collision attacks. In Section 3, we propose the framework of our new collision attack. In Section 4, we describe the bitwise collision detection method. In Section 5, we present the experiments of our new attack. An error-tolerant version of our attack is presented in Section 6, and the efficiency is analyzed in Section 7. Finally, Section 8 concludes the paper.

2. Preliminary

2.1 Notations

The cryptographic algorithm we focus on in this paper is AES. The 16-bytes plaintext and first-round sub key are denoted as $P = \{p_1, \cdots, p_{16}\}$ and $K = \{k_1, \cdots, k_{16}\}$. Plaintexts and power

traces are numbered by superscript, and the *i*th plaintext and power trace are written as P^i and T^i. The operations of 16 S-Boxes are handled sequentially, so a power trace can be cut into 16 sections, each of which is composed of l points. The section corresponding to the *a*th S-Box is denoted as $T_a = \{t_{a,1}, t_{a,2} \cdots, t_{a,l}\}$. Averaged power traces are used in our attack, denoted as $\overline{T} = \{\overline{T}_1, \cdots, \overline{T}_{16}\}$.

2.2 Linear Collision Attack

Collision attack proposed by Schramm *et al*. [12] is based on the concept of internal collision, where a function produces the same output for two inputs: $\phi(x_1) = \phi(x_2) = y$. Linear collision attack [17] describes how to recover the key from internal collisions based on linear equations. In AES, if a collision between the computations of S-Box *a* and *b* in the first round is detected, the attackers will have the following relation:

$$Sbox(p_a \oplus k_a) = Sbox(p_b \oplus k_b). \tag{1}$$

A linear equation can be deduced:

$$k_a \oplus k_b = p_a \oplus p_b = \Delta k_{a,b}. \tag{2}$$

A series of linear equations can be built with more collisions detected. Eventually, once 1 key byte is determined, the other 15 key bytes can be decided immediately. As a result, the size of key space is reduced to 2^8.

2.3 Correlation-Enhanced Collision Attack

Two main approaches have been proposed to detect side-channel collisions [18]: the binary test and the correlation-enhanced method [19]. The former one computes the distance between two power traces. Euclidean distance and absolute deviation [20] are usually used here. A Collision is confirmed if the distance is less than a threshold. However, this technique is a byte wise operation. A collision only indicates $HW(p_a \oplus k_a) = HW(p_b \oplus k_b)$, and (2) is not necessarily established. This method is also sensitive to false detections of collisions. The correlation-enhanced technique compares two series of (instead of two) power traces with correlation coefficient, and returns a score list of all the guessed value of $\Delta k_{a,b}$. This allows an improvement: By testing several highest-scored candidates instead of only the first one, the probability of finding a correct collision can be increased. But this approach needs to compute correlation coefficient for every guessed value of $\Delta k_{a,b}$, so the efficient is a problem.

2.4 LDPC Decoding Problem in Collision Attack

Gérard *et al*. [18] pointed out that the linear collision attack can be re-written as a LDPC decoding problem, since there exists a relationship:

$$\Delta k_{a,b} \oplus \Delta k_{b,c} = \Delta k_{a,c}, \quad \forall 1 \le a \ne b \ne c \le 16 \tag{3}$$

The set $\Delta K = \{\Delta k_{a,b} \mid 1 \le a \ne b \le 16\}$ can be regarded as a LDPC code, and (2) as parity-check nodes. Finding the correct ΔK is equivalent to decode the LDPC code.

It is noteworthy that (3) provides a solution to find out errors in collision detections. In this paper, we exploit (3) as an error-checking criterion, and detail the procedure in the next section.

3. A Novel Framework for Linear Collision Attack

In this section, we propose a new framework of collision attack. As shown in **Fig. 1**, the main body of the framework is a loop. Each iteration, called a partial attack, is based on a double sieve model, and contributes a part of information of the key. The loop iterates and accumulates the information until all the bytes of ΔK are determined. Finally 2^8 candidate keys that are compatible to the set ΔK are tested.

The double sieve model includes two screenings: 1) Collision detection sieves the probable candidate of the ΔK, and saves the result in a 1×120 array *DeltaKey1*. 2) Error detection screens out the false part of *DeltaKey1*, and saves the survivals in a 1×120 array *DeltaKey2*. Accumulated information of ΔK is kept in *DeltaKey3*.

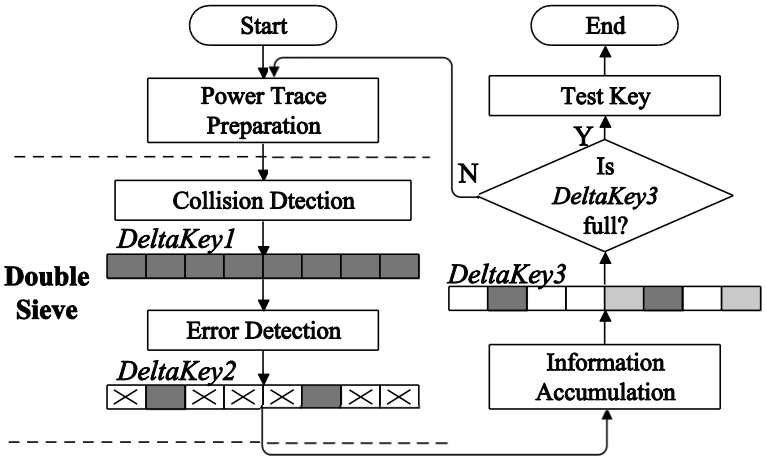

Fig. 1. Work flow of the new framework.

The work flow of our framework is showed in Algorithm 1. Each partial attack consists of 4 steps. First, power traces are collected and preprocessed in the **PreparePowerTraces** step. Then in **DetectCollision** step, these power traces are used to detect collisions. These two steps will be detailed in Section 4.

Algorithm 1 New Framework for Linear Collision Attack

Input: 8 power traces $\bar{T}^1,\cdots,\bar{T}^8$ for each partial attack.
Output: The guessed key K.

```
Clear DeltaKey3
while (DeltaKey3 is not full)
        (T̄^1,⋯,T̄^8) ← PreparePowerTraces ( )
        DeltaKey1 ← DetectCollision (T̄^1,⋯,T̄^8)
        DeltaKey2 ← DetectError (DeltaKey1)
        DeltaKey3 ← Accumulate (DeltaKey2, DeltaKey3)
end
for every candidate key K compatible to DeltaKey3
        if (TestKey ( K ))      return ( K )
end
```

In the **DetectError** step, as described in Algorithm 2, we use (3) to check every elements of *DeltaKey1*. A 1×120 array *Errorlist* is used to record how many times (3) is not satisfied for every $\Delta k_{a,b}$. If an equation of (3) is not satisfied, the three involved Δk 's will be marked. If $Errorlist((a,b))$ is larger than a threshold Th_{EL}, $\Delta k_{a,b}$ will be erased. Because there are 14 possible values of c which satisfy the condition $(a \neq c \neq b) \wedge (1 \leq c \leq 16)$, every $\Delta k_{a,b}$ has 14 relative equations in (3), so the maximum of $Errorlist((a,b))$ is 14. A wrong guess of $\Delta k_{a,b}$ tends to fail in most of the checks, whereas a correct guess has few failures. So we set Th_{EL} as a middle value, for example 7.

Algorithm 2 DetectError ( )

Input: *DeltaKey1*, a set of ΔK to be checked.
Output: The error-eliminated *DeltaKey1*.

```
1:   Errorlist ((a,b)) ← 0 (1 ≤ a ≠ b ≤ 16)
2:   for (each (a, b, c), 1 ≤ a ≠ b ≠ c ≤ 16)
3:       if ( DeltaKey1((a,b)) ⊕ DeltaKey1((b,c)) ≠ DeltaKey1((a,c)))
4:           Errorlist ((a,b)) ← Errorlist ((a,b)) + 1
5:           Errorlist ((a,c)) ← Errorlist ((a,c)) + 1
6:           Errorlist ((b,c)) ← Errorlist ((b,c)) + 1
7:       end
8:   end
9:   for (each (a, b), 1 ≤ a ≠ b ≤ 16)
10:      if (Errorlist ((a,b)) > Th_EL)      erase DeltaKey1((a,b))
11:  end
12:  return (DeltaKey1)
```

Accumulate step compares the newly obtained information in *DeltaKey2* and the accumulated information in *DeltaKey3*. Then *DeltaKey3* is refreshed with the union of *DeltaKey2* and *DeltaKey3*. An exception is that for some $\Delta k_{a,b}$, the corresponding values kept in *DeltaKey2* and *DeltaKey3* are different. Then they should be erased.

4. Bitwise Collision Detection

Here we detail the bitwise collision detection and the related **PreparePowerTraces** step. The essential idea is to find the 1-bit collision between two bytes, and to treat other bits as noise by choosing plaintexts and acquiring power traces properly. The input of S-Box, (i.e., $K \oplus P$), is chosen as the attack target, and Hamming Weight is used as the power model. We denote the bits in a plaintext byte and a key byte with u and v :

$$p_j = u_{j,8} \| u_{j,7} \| u_{j,6} \| u_{j,5} \| u_{j,4} \| u_{j,3} \| u_{j,2} \| u_{j,1}$$
$$k_j = v_{j,8} \| v_{j,7} \| v_{j,6} \| v_{j,5} \| v_{j,4} \| v_{j,3} \| v_{j,2} \| v_{j,1}$$

4.1 Preparation of Power Traces

For a partial attack, 8 power traces will be prepared. We use the most significant bit (i.e., bit 8) as an example to illustrate the process flow of the preparation of power traces:

1) Generate plaintexts of which the 8th bits of all the bytes are fixed (to zero for example). Other bits are random (e.g., $p_j = 0 \| u_{j,7} \| u_{j,6} \| u_{j,5} \| u_{j,4} \| u_{j,3} \| u_{j,2} \| u_{j,1}$ $(1 \le j \le 16)$).
2) Acquiring power traces with an oscilloscope that can average power traces in a real-time mode. Then only 1 power trace will be saved, denoted as T^8.
3) To further reduce the noise, the newly acquired power trace will be averaged along with those acquired in previous partial attacks. For the *n*th partial attack we have:

$$\overline{T}^{8,n} = \left(T^8 + \overline{T}^{8,n-1}(n-1)\right)/n \tag{4}$$

4) Cut $\overline{T}^{8,n}$ into 16 sections: $\overline{T}^8 = \left\{\overline{T}_1^8, \cdots \overline{T}_{16}^8\right\}$. (The superscript *n* is omitted for simplicity.)

4.2 Bitwise Collision Detection

As presented in Algorithm 3, we use $\overline{T}_a^i$ and $\overline{T}_b^i$ to detect the *i*th-bit collision between k_a and k_b $(1 \le a \ne b \le 16)$. Function **Distance** is the Euclidean distance of $\overline{T}_a^i$ and $\overline{T}_b^i$:

$$\textbf{Distance}\left(\overline{T}_a^i, \overline{T}_b^i\right) = \sum_{j=1}^{l}\left(\overline{t}_{a,j}^i - \overline{t}_{b,j}^i\right)^2 \tag{5}$$

If (5) is less than Th_{CD}, we can guess that $u_{a,i} \oplus v_{a,i} = u_{b,i} \oplus v_{b,i}$. Since $u_{a,i} = u_{b,i} = 0$, we have $D(i) = v_{a,i} \oplus v_{b,i} = 0$.

Algorithm 3 Bitwise Collision Detection

Input: 8 power traces $\overline{T}^1, \cdots, \overline{T}^8$
Output: *DeltaKey1*, the most probable candidate of ΔK.

```
for (each (a, b), 1 ≤ a ≠ b ≤ 16)
    for (i=1, … 8)
        if (Distance (T̄_a^i, T̄_b^i) < Th_CD)    D (i) = 0
        else    D (i) = 1
        end
    end
    DeltaKey1 ((a,b)) = D
end
return (DeltaKey1)
```

Th_{CD} should be chosen carefully to ensure the accuracy of collision detection. Since the result of **Distance** follows a chi-square distribution, Th_{CD} is relevant to the noise level. Here is an adaptively strategy to determine it: the median of the results of **Distance** can be chosen as Th_{CD} to make sure that they are divided into two groups with the similar sizes.

5. Experiments

5.1 Measurement Setup

As shown in **Fig. 2**, we built an experimental environment to mount the side-channel collision attacks. The target AES is implemented in a low-power, high-performance microcontroller AT89S52, which is suitable for various applications of WSN. A resistance of 10 Ohm is put in the power supply path of the microcontroller. An Agilent MSO-X 3054A oscilloscope with a differential probe is employed to acquire the voltage difference over the resistance which is related to the current consumed by the AT89S52. In our case, each raw power traces contains 10 000 points. For each partial attack, one power trace is averaged from m raw power traces. Here we set m to be 300.

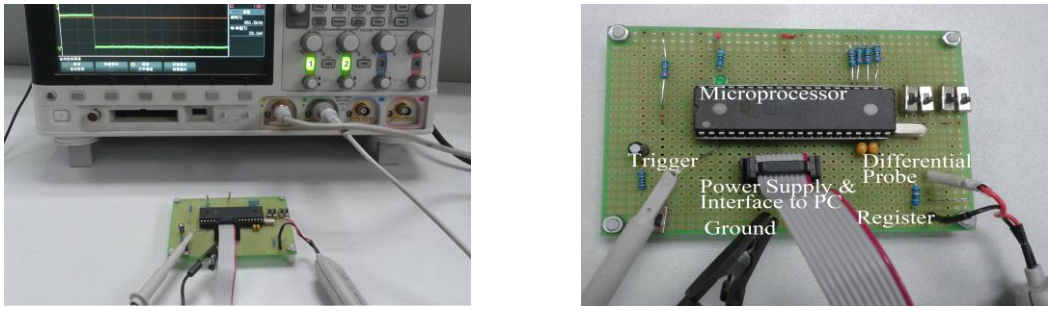

Fig. 2. Measurement setup of collision attacks on AT89S52.

5.2 Bitwise Collision Detection

For each pair of key bytes (k_a, k_b), we use bitwise collision detection method to find out $\Delta k_{a,b}$. Here we use (k_1, k_2) as an example to detail how it works, where

$$k_1 = \alpha_8 \| \alpha_7 \| \alpha_6 \| \alpha_5 \| \alpha_4 \| \alpha_3 \| \alpha_2 \| \alpha_1 = 01110010$$
$$k_2 = \beta_8 \| \beta_7 \| \beta_6 \| \beta_5 \| \beta_4 \| \beta_3 \| \beta_2 \| \beta_1 = 10011011 .$$

Fig. 3. shows the detection results. **Fig. 3(a)-(h)** correspond to the collision detection results of bit 8-1. For example, **Fig. 3(a)** shows $\left(\bar{T}_1^8 - \bar{T}_2^8\right)^2$. There exist obvious peaks, because $\alpha_8 \neq \beta_8$. In **Fig. 3(d)**, the curve $\left(\bar{T}_1^5 - \bar{T}_2^5\right)^2$ is close to zero, suggesting that $\alpha_5 = \beta_5$. Finally, the Euclidean distances of 8 bits and a threshold line (Th_{CD}) are plotted in **Fig. 3(i)**. Here we have $\Delta k_{1,2} = 11101001$.

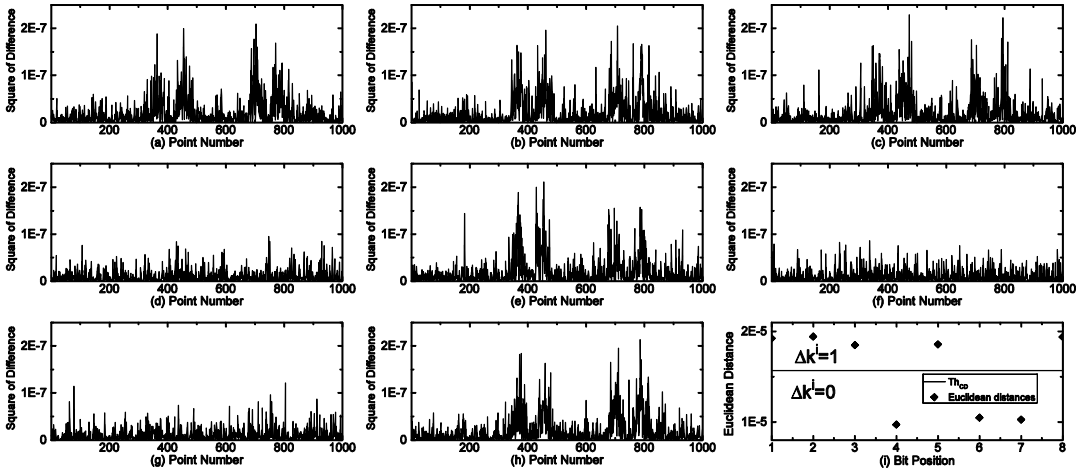

Fig. 3. The result of bitwise collision detection between k_1 and k_2.

5.3 Error Detection

After collision detection, a guessed ΔK is produced and kept in *DeltaKey1*. Then the error detection method is used to screen out the wrong part of ΔK. For the sake of simplicity, we focus on the triplet $\Delta k_{1,2}$, $\Delta k_{1,3}$, and $\Delta k_{2,3}$ to illustrate the workflow of our framework, where

$$\Delta k_{1,2} = 11101001 \quad \Delta k_{1,3} = 11101000 \quad \Delta k_{2,3} = 00000001.$$

As shown in **Fig. 4**, the triplet $\Delta k_{1,2}$, $\Delta k_{1,3}$, and $\Delta k_{2,3}$ are recovered within 2 partial attacks. In partial attack 1, the wrong guessed $\Delta k_{2,3}$ is discarded after error detection, and $\Delta k_{1,2}$, $\Delta k_{1,3}$ are delivered to *DeltaKey2*. Similarly, in partial attack 2, the false guess of $\Delta k_{1,3}$ is discarded. In the **Accumulate** step, the information of two partial attacks is merged together, then $\Delta k_{1,2}$, $\Delta k_{1,3}$, and $\Delta k_{2,3}$ are revealed. All the other parts of ΔK are recovered in the same way, and finally the key of AES is recovered.

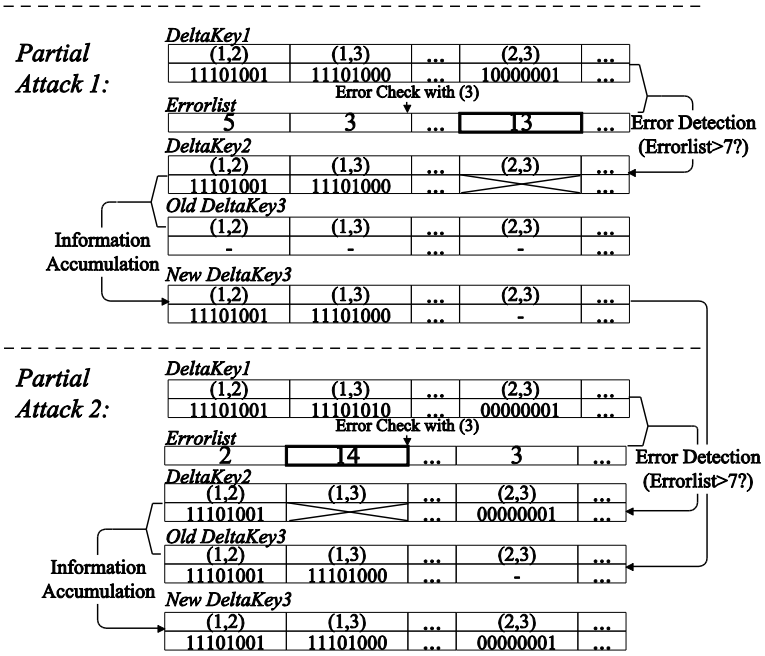

Fig. 4. Recovery process of $\Delta k_{1,2}$, $\Delta k_{1,3}$, and $\Delta k_{2,3}$.

6. Improved Framework with Error-Tolerant Mechanism

Here we propose an error-tolerant version of our approach. Our original approach is a binary test and has a low cost of computation. The correlation-enhanced method produces *a list of* candidates of $\Delta k_{a,b}$, and increases the success rate at the expense of larger computation amount. Our improved approach uses the concept of list to realize an error-tolerant mechanism and keeps the computation amount at a reasonable level. Improvements are mainly made in **DetectCollision** and **DetectError** steps:

6.1 Modified Bitwise Collision Detection

To increase the number of candidates of each ΔK, the most straight forward idea is to include the 8 hypothetic values which are one-bit different from the most probable one. The output of

the **DetectCollision** step, *DeltaKey1*, then becomes a 9×120 matrix. The elements of the matrix are denoted as $DeltaKey1\left(i,\left(a,b\right)\right)$ $\left(1\le i\le 9,1\le a\ne b\le 16\right)$.

6.2 Modified Error Detection

There are two main changes in the modified error detection method as presented in Algorithm 4: 1)The input *DeltaKey1* becomes a 9×120 matrix, and the first row of *DeltaKey1* is checked in loop 1 (step 2-6); 2) In loop 2, if $Errorlist\left(\left(a,b\right)\right) > Th_{EL}$, other 8 candidates will be checked in turn, as shown in Algorithm 5. Only if no candidate passes the check will $DeltaKey1\left(1,\left(a,b\right)\right)$ be erased, or it will be replaced by the candidate which leads to the minimum $Errorlist\left(\left(a,b\right)\right)$.

Algorithm 4 DetectError ( ) (Modified)

Input: *DeltaKey1*, a 9×120 matrix, including 9 candidates of ΔK.
Output: The error-eliminated *DeltaKey1* (1×120).

1: $Errorlist\left(\left(a,b\right)\right) \leftarrow 0$ $\left(1\le a\ne b\le 16\right)$
2: **for** (each $(a, b, c), 1\le a\ne b\ne c\le 16$)
3: **if** ($DeltaKey1\left(1,\left(a,b\right)\right)\oplus DeltaKey1\left(1,\left(b,c\right)\right)\ne DeltaKey1\left(1,\left(a,c\right)\right)$)
4: $Errorlist\left(\left(a,b\right)\right) \leftarrow Errorlist\left(\left(a,b\right)\right) + 1$
$Errorlist\left(\left(a,c\right)\right) \leftarrow Errorlist\left(\left(a,c\right)\right) + 1$
$Errorlist\left(\left(b,c\right)\right) \leftarrow Errorlist\left(\left(b,c\right)\right) + 1$
5: **end**
6: **end**
7: **for** (each $(a, b), 1\le a\ne b\le 16$)
8: **if**($Errorlist\left(\left(a,b\right)\right) > Th_{EL}$) **EnumerateCheck**($DeltaKey1\left(i,\left(a,b\right)\right), \left(1\le i\le 9\right)$)
9: **end**
10: **return** (the first row of *DeltaKey1*)

Algorithm 5 EnumerateCheck ( )

Input: $DeltaKey1\left(i,\left(a,b\right)\right)$ $\left(1\le i\le 9\right)$, candidates of $\Delta k_{a,b}$

1: *temp1*← *Errorlist* (*a*, *b*), *temp2*← 1
2: **for** (i = 2, …, 9)
3: *Errorlist* (*a*, *b*) ← 0
4: **for** $(1\le c\le 16, c\ne a, c\ne b)$
5: **if** ($DeltaKey1\left(i,\left(a,b\right)\right)\oplus DeltaKey1\left(1,\left(b,c\right)\right)\ne DeltaKey1\left(1,\left(a,c\right)\right)$)
6: *Errorlist* (*a*, *b*) ← *Errorlist* (*a*, *b*) + 1
7: **end**
8: **end**
9: **if** ($Errorlist\left(\left(a,b\right)\right)$ <*temp1*) *temp1*← *Errorlist* $\left(\left(a,b\right)\right)$, *temp2*← *i*
10: **end**
11: **if** ($temp1 < Th_{EL}$) $DeltaKey1\left(1,\left(a,b\right)\right) \leftarrow DeltaKey1\left(temp2,\left(a,b\right)\right)$
12: **else** erase $DeltaKey1\left(1,\left(a,b\right)\right)$
13: **end**
14: **return**

7. Efficiency Comparison

We do simulations in MATLAB to compare the efficiency of our attack and the LDPC method. The comparisons include the success rate and time (online time and offline time).

7.1 Online Time *vs.* Success Rate

We denote the time for the oscilloscope to capture and average a power trace as τ_A, and the time to save a trace as τ_S. In our case where one power trace contains 10 000 sample points, τ_S is roughly 50 times of τ_A. Let m be the number of power traces to be averaged, n be the number of partial attacks, the total online time τ_{OL} is

$$\tau_{OL} = 8n\left(m\,\tau_A + \tau_S\right) = 8n\left(0.02m + 1\right)\tau_S . \tag{6}$$

In Our original version, we fix $m = 300$, so

$$\tau_{OL} = 56n\,\tau_S . \tag{7}$$

In the error-tolerant version, we set $m = 50$, so

$$\tau_{OL} = 16n\,\tau_S . \tag{8}$$

In order to assess the efficiency, we plot the success rates of attacks as a function of online time τ_{OL}, rather than the number of raw power traces, by sweeping the upper limit of n. As shown in **Fig. 5**, our original approach (denoted as DS) is more efficient than the LDPC method when the online time is less than $900\tau_S$ .The error-tolerant version (denoted as DS-ET) achieves an obvious gain in efficiency. The online time needed to reach a success rate of 0.9 is 90% less than that of LDPC method.

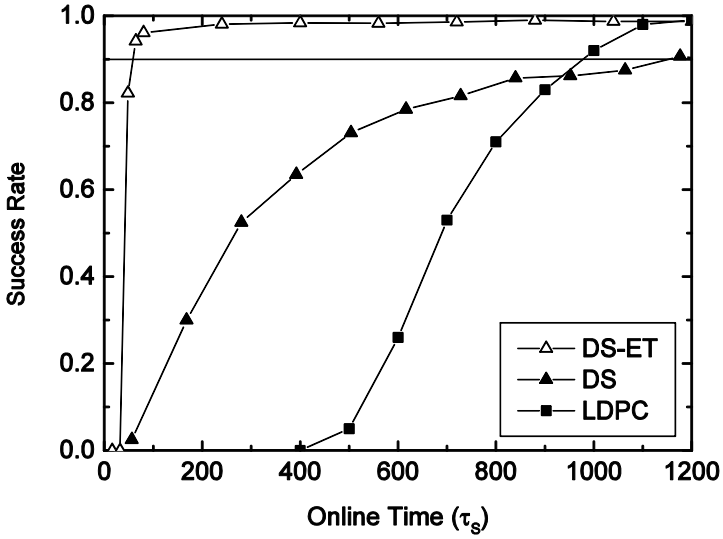

Fig. 5. The success rates as a function of online time τ_{OL} .

7.2 Online Time *vs.* Offline Time

Offline time reflects the computational complexity of the attack algorithm. We examined the operation time for three algorithms (LDPC, DS, and DS-ET) to carry out 1000 attacks. As shown in **Fig. 6**, the computational complexity of LDPC method is significantly higher than our approaches. The error-tolerant version of our attack needs least offline time to recover the key.

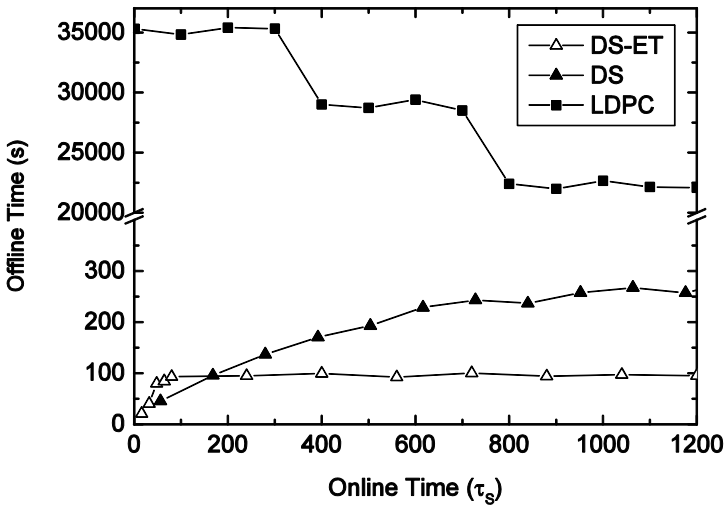

Fig. 6. The offline time as a function of online time τ_{OL}.

7.3 Selection of Parameters

The former experiments are done with a fixed m. Here we focus on the impacts of m on the success rates of attacks.

As shown in **Fig. 7**, the success rate curves reach to a larger upper limit with lager m. However, increasing m also increases the time for each partial attack. So there is no need to increase m once the upper limit of success rate is close to 1. In our case (the error-tolerant version), $m = 50$ is reasonable.

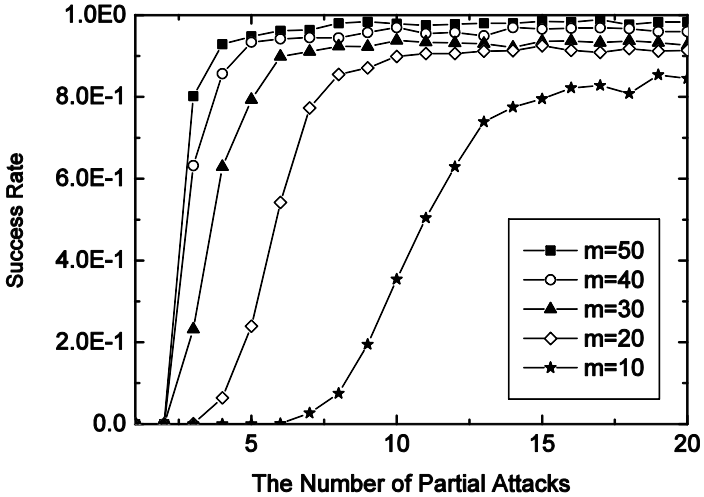

Fig. 7. The success rates with different average times m as a function of n (the number of partial attacks).

8. Conclusion

We propose a double sieve collision attack based on bitwise collision detection in this paper, and give an error-tolerant version which significantly reduces the time of online stage. Practical attacks are successfully mounted on AES implemented in a real chip which can be used in WSN. We also compare the efficiency of our attack with the work published by Gérard *et al*. [18]. The experiment result shows our attack saves 90% of time to reach a success rate of 0.9.

Although AES is the target algorithm in this paper, our work can be extended to other symmetry cryptography algorithms that are vulnerable to collision attack.

References

[1] J. Zheng and A. Jamalipour, *Wireless Sensor Networks: A Networking Perspective*, Wiley, New York, 2009. Article (CrossRef Link)

[2] G. Padmavathi and M. Shanmugapriya, "A survey of attacks, security mechanisms and challenges in wireless sensor networks," *arXiv preprint arXiv: 0909.0576*, 2009. http://arxiv.org/abs/0909.0576

[3] C. Krauß, M. Schneider and C. Eckert, "On handling insider attacks in wireless sensor networks," *Information Security Technical Report*, vol. 13, no. 3, pp. 165-172, August, 2008. Article (CrossRef Link)

[4] NIST Std. 197, "Announcing the Advanced encryption standard (AES)," 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[5] IEEE Std. 802.15.4, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) speci-fications for Low Rate Wireless Personal Area Networks (LR-WPANS)," 2003. http://ieeexplore.ieee.org/iel5/8762/27762/01237559.pdf

[6] P. C. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Proc. of CRYPTO' 1996*, LNCS, vol. 1109, pp. 104–113, Springer, Heidelberg, 1996. Article (CrossRef Link)

[7] P. C. Kocher, J. Jaffe and B. Jun, "Differential power analysis," in *Proc. of CRYPTO' 1999*, LNCS, vol. 1666, pp. 388–397, Springer, Heidelberg, 1999. Article (CrossRef Link)

[8] S. Chair, J. R. Rao and P. Rohatgi, "Template attacks," in *Proc. of CHES 2002*, LNCS, vol. 2523, pp. 13–28, Springer, Heidelberg, 2003. Article (CrossRef Link)

[9] E. Brier, C. Clavier and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. of CHES 2004*, LNCS, vol. 3156, pp. 16–29, Springer, Heidelberg, 2004. Article (CrossRef Link)

[10] B. Gierlichs, L. Batina, P. Tuyls and B. Preneel, "Mutual information analysis," in *Proc. of CHES 2008*, LNCS, vol. 5154, pp. 426-442, Springer, Heidelberg, 2008. Article (CrossRef Link)

[11] G. de Meulenaer and F. X. Standaert, "Stealthy compromise of wireless sensor nodes with power analysis attacks," *Mobile Lightweight Wireless Systems*, LNICST, vol. 45, pp. 229-242, Springer, Heidelberg, 2010. Article (CrossRef Link)

[12] K. Schramm, T. J. Wollinger and C. Paar, "A new class of collision attacks and its application to DES," *in Proc. of 10th Int. Workshop on Fast Software Encryption*, LNCS, vol. 2887, pp. 206–222, Springer, Heidelberg, 2003. Article (CrossRef Link)

[13] K. Schramm, G. Leander, P. Felke and C. Parr, "A collision-attack on AES: Combining side channel and differential attack," in *Proc. of CHES 2004*, LNCS, vol. 3156, pp. 163-175, Springer, Heidelberg, 2004. Article (CrossRef Link)

[14] H. Ledig, F. Muller and F. Valette, "Enhancing collision attacks," in *Proc. of CHES 2004*, LNCS, vol. 3156, pp. 176–190, Springer, Heidelberg, 2004. Article (CrossRef Link)

[15] A. Bogdanov, "Improved side-channel collision attacks on AES," in *Proc. of 14th Int. Workshop on Selected Areas in Cryptography*, LNCS, vol. 4876, pp. 84–95, Springer, Heidelberg, 2007. Article (CrossRef Link)

[16] .A. Bogdanov, "Multiple-differential side channel collision attacks on AES," in *Proc. of CHES*

2008, LNCS, vol. 5154, pp. 30–44, Springer, Heidelberg, 2008. Article (CrossRef Link)
[17] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet and V. Verneuil, "Improved collision-correlation power analysis on first order protected AES," in *Proc. of CHES 2011*, LNCS, vol. 6917, pp. 49–62, Springer, Heidelberg, 2011. Article (CrossRef Link)
[18] B. Gérard and F. X. Standaert, "Unified and optimized linear collision attacks and their application in a non-profiled setting," in *Proc. of CHES 2012*, LNCS, vol. 7428, pp. 175–192, Springer, Heidelberg, 2012. Article (CrossRef Link)
[19] A. Moradi, O. Mischke and T. Eisenbarth, "Correlation-enhanced power analysis collision attack," in *Proc. of CHES 2010*, LNCS, vol. 6225, pp. 125–139, Springer, Heidelberg, 2010. Article (CrossRef Link)
[20] A. A. Sveshnikov and B. R. Gelbaum (Eds.), *Problems in Probability Theory, Mathematical Statistics and Theory of Random Functions*, Courier Dover Publications, New York, 1968.

Yanting Ren was born in 1987. She received her B.S. degree in Tsinghua University in 2010. She is currently a Ph.D. student in Institute of Microelectronics, Tsinghua University. Her main research interests include side-channel attack, and side-channel attack resistant implementation in cryptography.

Liji Wu was born in 1965. He received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1988, 1991 and 1997, respectively. He currently works as a professor in Tsinghua University. His main research interests include Circuits and Systems in Commercial Information Security and Automotive Electronics.

An Wang was born in 1983. He received his Ph.D. degree in Shangdong University in 2011. He currently works as a post doctor in Tsinghua University. His main research interests include side-channel attack, embedded system, and fast implementation in cryptography. He is also the webmaster of a cryptographic website (http://www.mathmagic.cn).