# Vulnerability Analysis for Crypto Devices against Probing Attack

Lingxiao Wei†, Jie Zhang†, Feng Yuan†, Yannan Liu†, Junfeng Fan‡ and Qiang Xu†

†CUhk REliable Computing Laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

‡Open Security Research, Shenzhen, China

## ABSTRACT

Probing attack is a severe threat for the security of hardware cryptographic modules (HCMs). In this paper, we make the first step to evaluate the vulnerability of HCMs against probing attack, wherein we investigate the probing complexity and the key candidate reduction capability for probing attack on every signal in the circuit. We also present approximate solutions for the calculation of the proposed metrics to reduce computational complexity. Experimental results demonstrate that the proposed evaluation metric is both effective and efficient.

## 1. INTRODUCTION

Driven by the increasing demand for secure computation and communication in the era of internet of things (IoT), hardware cryptographic modules (HCMs) are widely used nowadays, ranging from dedicated secure applications such as smartcards to all sorts of "smart" devices connected to the Internet. As HCMs provide the "root of trust" in the system, it is essential to ensure its own security. However, while the cryptographic algorithms themselves are extremely difficult (if not impossible) to break mathematically, their implementations are often not. That is, the internal computational information can be eavesdropped either directly or indirectly by side channels, such as power [1] or timing [2]. Among all side-channel attacks (SCAs), probing attack is a simple yet severe threat, which cracks the secret key by directly spying those key-related internal signals with metal needles and then guessing the key according to the observed information.

Probing attack was first introduced by Handschuh *et al.* [3], wherein they successfully found the secret key of various tamper-resistant HCMs including SM-1, DES and RC5 implementations. Later, Schmidt *et al.* [4] adopted probing attack to crack the key of AES implementation. In [5], Ishai *et al.* developed several efficient techniques against probing attacks by building private circuits. However, due to the hardware overhead of private circuits, their approach could only selectively protect some of circuits. This raises an interesting and relevant question: when designing a crypto device, which part of the circuitries are more vulnerable to probing attacks and hence require to be protected.

In the literature, there were some prior works evaluating the vulnerability of SCAs, but they are not applicable for probing attacks. Standaert *et al.* [6] proposed a unified framework for the analysis of SCAs based on mutual information. As a general method, however, this approach incurs quite high computational complexity to quantify the relationship between keys and side channel outputs. Demme *et al.* [7] developed a so-called side-channel vulnerability factor (SVF) quantifying the correlation between attackers' observation and measures. SVF works well for timing-based attacks [8], but it is not applicable for probing attacks because it cannot represent the relationship between certain probed wires and the key.

Motivated by the above, in this paper, we propose a novel vulnerability metric that evaluates the threat of probing attack for HCMs. To be specific, it quantifies the probing complexity and the key candidate reduction capability for every signal in the HCM by considering the cryptographic implementation on the logic gate level. We also present approximate solutions for the calculation of the proposed metrics to reduce computational complexity. Our experimental results show that the proposed vulnerability metric is able to effectively reflect the threat posed by probing attack.

The rest of the paper is organized as follows. Section 2 presents preliminaries on probing attack and motivates this work. In Section 3, we detail the proposed vulnerability metric for probing attack. Section 4 presents the bound of the vulnerability metric and proposes an efficient method to approximate its calculation. Next, we validate the proposed vulnerability metric in Section 5. Finally, Section 6 concludes the paper.

## 2. BACKGROUND AND MOTIVATION

### 2.1 Probing Attack

Before probing any internal signals of interest in a crypto device, the passivation above such signal should be firstly removed. Thereafter, the test pad can be placed on probed signals. Algorithm 1 presents how probing attack operates. It is an iterative procedure, wherein attackers probe internal signals and reduce key candidates, denoted by $K$, to the point that they can crack the entire key for the remaining candidates in a brute-force manner (Line 1-6). The rule to remove a key candidate is to check whether it would output the different value denoted by $f_S(k_i, d)$ compared to the probed value $s$ under data values denoted by $d$, as shown by Line 12. During each probing iteration shown by Line 7-17, attackers would verify all key candidates with the random data input denoted by $d$. Attackers stop probing these signals until key candidates cannot be reduced.

Apart from countermeasures special tailored for probing attacks [5], one common technique against such attacks is to add *meshes*[1] to the device. A so-called "glue logic" design approach helps prevent probing attacker from reverse engineering signals by obfuscating the regular structure of standard components on the layout.

**Algorithm 1:** The Procedure of Probing Attack

```
/* $K$ is the set of the whole key; */
1  repeat
2      Choose one or some signals denoted by $S$;
3      $K_S$ is the associated key candidate set for $S$;
4      ProbingOneSignal($S, K_S$);
5      $K \leftarrow (K - K_S)$;
6  until $K$ is able to be enumerated by brute force;
7  ProbingOneSignal($S, K_S$)
8      repeat
9          $d \leftarrow$ random number;
10         Probe $s = f_S(k_g, d)$;
11         foreach $k_i \in K_S$ do
12             if $f_S(k_i, d) \neq s$ then
13                 Remove $k_i$ from $K_S$;
14             end if
15         end foreach
16     until $K_S$ is unable to be reduced;
17 end
```

## 2.2 Threat Model

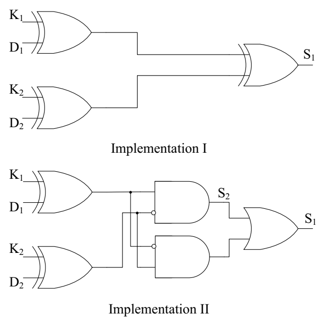
We follow the same threat model of [4]. That is, we assumed that attackers are capable of physically accessing and depackaging the crypto chip. Thereby, a probe is able to be placed onto the chip and retrieve the value of signals during all encrypt and decrypt operations controlled by the attacker. Moreover, we assume that attackers have sufficient knowledge about the implementation of the crypto device and capability to correctly retrieve the value of desired signal, helping them to reduce the key candidates via probed information.

## 2.3 Motivation

In terms of probing attack, not only algorithm-level signals but also gate-level signals are able to leak the key-related information. Take Implementation I shown in Fig. 1 as an example, wherein the signal denoted by $S_1$ is a gate-level signal. It is obvious that $S_1$ would leak the information of two key bits, $K_1$ and $K_2$, by probing its value, as shown in the K-map of $S_1$. Moreover, different implementations for the same Boolean function could leak different amount of information. As shown by Fig. 1, Implementation I and Implementation II are two realizations for the function $S_1 = K_1 \oplus D_1 \oplus K_2 \oplus D_2$. Suppose $S_1$ of Implementation I and $S_2$ of Implementation II are probed. By going to details their K-maps, we find that by probing $S_1$, the key candidate 00 is indistinguishable with 11, while by probing $S_2$, all four key candidates can be distinguished.

However, until now, there is no theory available to evaluate the vulnerability of cryptographic implementation against probing attack. This motivates us to propose a novel vulnerability metric for probing attack by considering the cryptographic implementation.

[1]Meshes are redundant layers of metallization on top of the device itself, which would alarm if triggered by any detection of short circuit or interruption.



K-Map

| $K_1K_2 \backslash D_1D_2$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 |
| 11 | 0 | 1 | 1 | 0 |

$S_1$ value

| $K_1K_2 \backslash D_1D_2$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |

$S_2$ value

Figure 1: An motivational example

## 3. THE PROPOSED VULNERABILITY METRIC

### 3.1 Overview

In practice, a complete vulnerability metric for probing attack should cover probing all signal combinations of HCM, as discussed above. However, it is very complex, if not impossible, to evaluate all signal combinations, especially considering that there are a great number of signals in the today's HCM. Consequently, the purpose of the proposed vulnerability metric is to evaluate the vulnerability of every single signal separately and help designers to make the decision on protecting which signal in the HCM. Intuitively, the reason our vulnerability metric works is that a vulnerable signal is more likely to be the probed target.

Our vulnerability metric evaluates those signals that are able to be utilized for probing attack in practice. These signals should satisfied following two requirements: (i) they are not protected by countermeasures for probing attack; (ii) they are able to help attackers to filter out key candidates within their computational capability. We rule out signals violating the second requirement by the proposed complexity metric that is detailed in Section 3.2.

The proposed vulnerability metric aims to evaluate the ratio between the effort attackers pay and the benefits they obtain during the probing one signal. Therefore, we define it as follows.

Definition 1. *Consider a signal $S$ that is driven by $N_k^S$ key bits. The vulnerability of this signal in terms of probing attack denoted by $V_S$ is give by*

$$V_S = \frac{1}{\sum_{i=0}^{R} |\boldsymbol{K}_i|} \times \frac{2^{N_k^s}}{|\boldsymbol{K}_{final}|}, \tag{1}$$

where $R$ denotes the total number of iterations for probing this signal, $|\mathbf{K}_i|$ denotes the number of key candidates after verifying $i$-th iteration, and $|\mathbf{K}_{final}|$ denotes the final key space size after the probing this signal.

Look at the definition of the vulnerability in Eqn. 1, we observe that it contains two important components. The first one represented by $\sum_{i=0}^{R} |\mathbf{K}_i|$ describes the complexity of probing this signal while the second one represented by $2^{N_k^S}/|\mathbf{K}_{final}|$ illustrates the capability of reducing the key candidates, wherein $|\mathbf{K}_{final}|$ denotes the final key space size.

In the following, we detail the probing complexity and the final key space size separately.

Table 1: List of Notations

| Symbol | Meaning |
|---|---|
| $\mathbf{K}$ | Key candidate set |
| $\mathbf{D}$ | Data set |
| $k_g$ | Genuine key |
| $k, k_i$ | Key candidate |
| $d, d_i$ | Data pattern |
| $K^{k_i}$ | Key candidate containing $k_i$ |
| $\lvert\mathbf{K}\rvert$ | Size of key candidate set |
| $\lvert\mathbf{D}\rvert$ | Size of data set |
| $S$ | Internal signal |
| $N_k^S$ | Number of key bits correlates with signal $S$ |
| $N_d^S$ | Number of data bits correlates with signal $S$ |

## 3.2 Probing Complexity

The probing complexity is closely related to the iteration complexity and the number of iterations. The iteration complexity refers to the number of operations verifying all key candidates in one iteration. The number of iterations relies on the choice of data, because different data leads to different numbers of operations to determine all key bits. As a result, we detail these two factors in the following.

### 3.2.1 Iteration Complexity

Iteration complexity is referred to the operations needed for traversing the key candidate set. Obviously it depends on the size of key candidate set. The attacker at least need to iterate all the elements in the initial key candidate set, which is exponential to the number of correlated key bits. This criteria can serve as a deciding rule for attackers to filter out signals that are beyond their capability. From the side of designers, those signals with large number of correlated key bits, e.g., 60 bits, can be viewed as "safe" ones.

### 3.2.2 Data Revealability

Another important factor that an attacker cares about is on how to choose a specific data that can help to reduce more key candidates, thus reducing the iteration round. Intuitively, different data have different abilities to reveal the key information to the targeted signal. Every data pattern can be viewed as a map function which maps a specific key candidate to logic value 0 and 1, i.e., $S = f(K, D)$ can be viewed as a parametric function $S = f_D(K)$. For example, if one data pattern is more likely to map most elements in the current key candidate set to 1/0 while another data pattern tends to map the candidates evenly to 1/0, obviously the latter one leaks more information to the targeted signal. On the average case, we are able to eliminate more keys on the second condition. Therefore we use the concept of data revealability to represent the ability of one data pattern to eliminate key candidates. It should also be noted that the data revealability does not only relate to data itself, it also depends on the current key candidate set. Here we define the data revealability as follows:

Definition 2. *Suppose we have a key candidate set $\boldsymbol{K}$, and a data pattern $d$ performing probing attacks on it. If the size of $\boldsymbol{K}$ is $N$, and the number of key candidates that are mapped to 1 by $d$ is $N_1$ while the number of key candidates that are mapped to 0 by $d$ is $N_0$. Obviously, $N = N_0 + N_1$. We define the data revealability for $d$ with respect to $\boldsymbol{K}$ as:*

$$R_d^{\boldsymbol{K}} = \frac{2N_1N_0}{N^2}. \tag{2}$$

If every key candidate has same probability to be the genuine key, the data revealability defines the portion of key candidates that can be eliminated by a data pattern on a specific key candidate set in an average case. From the definition, it is obvious that the data revealability obtains its maximum value when $N_0 = N_1 = \frac{1}{2}N$, the $R_d^{\mathbf{K}} = 0.5$. Therefore in the best expected case, we are able to eliminate at most half of the key candidate by probing one signal at a time with one data.

The iteration round may vary according to different choices of data applied. The uncertainty of attackers' performance is not preferable for an evaluation procedure. With the definition of data revealability, designers are able to predict the average best case an attacker can achieve in probing attack. Designer may simulate the probing attack, counting the iteration round. Instead of applying random data at each round, designers would choose the data with the highest data revealability with respect to the current key candidate set. At each round, the key candidate set would shrink at highest speed, thus providing a lower bound of iteration rounds.

## 3.3 Final Average Key Space Size

In general, the ability of reducing overall key space through probing one signal is determined by the correlation between this signal with the driven key bits. However, it is quite possible that these key bits cannot be cracked by probing one signal, and attackers may end up with a set of key candidates. Thus, the *size of final key candidate set* is one important fact evaluating the ability of reducing key candidates by probing one signal.

Before discussing the size of final key candidate set, we first investigate the condition that a key can be distinguished from other keys by observing one particular signal in circuit and then give a metric to denote the final key space size.

The observed signal's relation with data and key input can be modeled as a Boolean function. Suppose $S$ represent the signal to be probed, while $K$ and $D$ represent corresponding key and data inputs with length of $N_k^S$ and $N_d^S$ respectively. Therefore $S$ can be represented as $S = f(K, D)$.

Definition 3. *For $\forall d \in \boldsymbol{D}$, if key candidate $k_a \in \boldsymbol{K}$ and key candidate $k_b \in \boldsymbol{K}$ has the same value on $S$, $S = f(k_a, d) = f(k_b, d)$, we say $k_a$ and $k_b$ are indistinguishable, denoted as $k_a \sim k_b$.*

If the genuine key $k_g$ is indistinguishable from other key candidates, the final key candidate set a probing adversary get must contain all the key candidates indistinguishable from the genuine key.

With the key indistinguishability defined above, for every key candidate $k_i$, it belongs to a key candidate set, denoted as $\mathbf{K}^{k_i}$, in which all elements are indistinguishable with each other. If $k_i$ happens to be the genuine key, key candidate set $\mathbf{K}^{k_i}$ will be the final output of a probing attack algorithm. As the genuine key may be any element in the candidate set, it is impossible to accurately predict the exact final key space when performing a static analysis. Here we gave a metric based on the average case. We can define the final average key space size as follows:

$$\overline{|\mathbf{K}_{final}|} = \sum_{i=0}^{|\mathbf{K}|} \Pr[k_g = k_i] \cdot |\mathbf{K}^{k_i}|. \tag{3}$$

The size of final key candidate set is determined when given a genuine key and an arbitrary signal. We use the probability of the genuine key being every possible key candidate to average the size of key candidate set.

The total key space can be divided into several sets which are non-overlapping and within each set, the keys are indistinguishable from each other. If the genuine key is chosen total randomly on the key candidate set, i.e., all key candidates has the same probability to be the genuine key, The final average key candidate set size can be simplified as:

$$\overline{|\mathbf{K}_{final}|} = \sum_{i=0}^{t} \frac{|\mathbf{K}^i|^2}{|\mathbf{K}|}, \tag{4}$$

in which $t$ represents the total number of final key candidate set which can be distinguished from each other.

## 4. EVALUATING VULNERABILITY ON CRYPTO IMPLEMENTATIONS

In this section, we investigate ways to apply metric proposed above to real crypto implementation. First we analyze the vulnerability under a common crypto assumption and then give concrete and practical solutions to evaluate a netlist.

### 4.1 Vulnerability under *perfectly secret* assumption

Before we dig deep into applying metric on crypto implementations, we start to examine a common assumption, *perfectly secret* assumption [9], in cryptographic algorithm:

Theorem 1. *Suppose $b_C$ is one bit of the ciphertext $C$ for a perfectly secret encryption algorithm, $b_C$ has the same probability to be 0 or 1:*

$$\Pr[b_C = 1] = \Pr[b_C = 0] = 0.5. \tag{5}$$

Though the theorem says the bit of ciphertext has the same probability to be 1 and 0, we can safely assume not only the bits of cipherthext, but also the signals after levels of diffusion and confusion hold the assumption. Moreover, a "good" encryption scheme is required to be resistant to chosen plaintext attack meaning that on every given data input, the probability of targeted signal to be 1 or 0 should also be the same.

Based on the above assumption, if signal $S$ is randomly selected, the probability of each item to be 0 or 1 should be the same, i.e., $\Pr[S = 0|k_i, d_i] = \Pr[S = 1|k_i, d_i]$, in which $S = f(k_i, d_i)$. Given this probability, we can calculate the probability that a genuine key is indistinguishable from any other key candidates in $\mathbf{K}$:

$$\Pr[k_g \sim k_i] = |\mathbf{K}|(0.5)^{|\mathbf{D}|}, \forall k_i \in \mathbf{K}, \tag{6}$$

in which $|\mathbf{K}|$ denotes the size of key candidate set while $|\mathbf{D}|$ denotes the number of all possible data patterns. Usually the size of key candidate set is comparable with the size of data set. If $|\mathbf{K}| = |\mathbf{D}| = 8$, the probability that the genuine key is indistinguishable is 0.03125, which is a very small value.

Similarly, the probability of final average key space size to be 1 is:

$$\Pr[\overline{|\mathbf{K}_{final}|} = 1] = (1 - |\mathbf{K}|(0.5)^{|\mathbf{D}|})^{|\mathbf{K}|}. \tag{7}$$

The probability approaches 1 as the $|\mathbf{K}|$ and $|\mathbf{D}|$ increase.

Then we consider another aspect: the choices of data, which can be represented as the data revealability as shown in Section 3. If the 0 or 1 is evenly distributed, the number of value mapped to 0 or 1 by one data pattern follows binomial distribution, i.e., $N_0/N_1 \sim B(|\mathbf{K}|, 0.5)$, with expectation of $0.5|\mathbf{K}|$ and standard deviation of $0.5\sqrt{|\mathbf{K}|}$. When $|\mathbf{K}|$ is large, the binomial distribution can be approximated with the normal distribution. So we can get:

$$\Pr[R_d^{\mathbf{K}} \geq 0.5 - 2/|\mathbf{K}|] \geq 95\%. \tag{8}$$

When $|\mathbf{K}|$ increases, the data revealability has high probability to reach the neighborhood of its maximum.

With the above analysis, if the one signal conforms with the assumption, it has high probability to have a final key candidate set of size 1 and to be utilized for reducing at a highest rate with arbitrary data patterns, i.e., half at each time, leading to least effort possible to break. Reviewing the formula in Eqn. 1, for signals obeying the assumption, the vulnerability can be directly calculated by:

$$V_S = 0.5 \times \frac{2^{N_k^S+1}}{2^{N_k^S+1} - 1}. \tag{9}$$

The signals' vulnerability decreases as the number of correlated key bits increases and converges to 0.5 eventually. It is also an upper bound of vulnerability for signals if they depend on the same number of key bits. In fact, the probing attack is similar to a brute force cracking algorithm that only focuses on a small number of key bits. Probing Attack is practical as it can divide the total key space into several parts and conquer them separately. The upper bound of vulnerability is the lower bound of complexity an attacker need to break these correlated key bits. The number of correlated key bits increases, the expected best case of efforts to crack all of them by probing attack would approaches to the 2 times to that by breaking them brute-forcely.

The limitation is that "perfectly secret" assumption does not hold for all signals in a crypto implementation. In the assumption, the signals are supposed to be sufficiently confused and diffused, which is true for ciphertext, and signals close to it. But there also exist signals for which this assumption does not hold, requiring a practical tool to calculate vulnerability which is to be shown in next subsection.

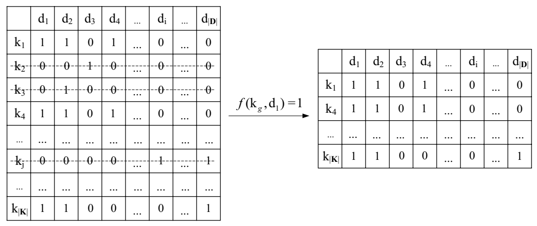
### 4.2 Evaluation Tool and Approximation

We propose to use a table like Fig. 2, denoted as *Key-Data Map*, involving key candidates and data patterns to represent the relationship between probed signal and key/data.

| | $d_1$ | $d_2$ | $d_3$ | $d_4$ | ... | $d_i$ | ... | $d_{\|D\|}$ |
|---|---|---|---|---|---|---|---|---|
| $k_1$ | 1 | 1 | 0 | 1 | ... | 0 | ... | 0 |
| $k_2$ | 0 | 0 | 1 | 0 | ... | 0 | ... | 0 |
| $k_3$ | 0 | 1 | 0 | 0 | ... | 0 | ... | 0 |
| $k_4$ | 1 | 1 | 0 | 1 | ... | 0 | ... | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $k_j$ | 0 | 0 | 0 | 0 | ... | 1 | ... | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $k_{\|K\|}$ | 1 | 1 | 0 | 0 | ... | 0 | ... | 1 |

Figure 2: An Example of a Key-Data Map of one Signal

In Fig. 2, we have a key candidate set $\mathbf{K}$ containing all the key candidates $k_1, k_2, \ldots, k_{|\mathbf{K}|}$, i.e., all the rows in the Key-Data Map, and all the data patterns $d_1, d_2, \ldots, d_{|\mathbf{D}|}$, i.e., all the columns in the Key-Data Map, and the value of targeted signal that mapped by each data and key candidates with a boolean function $f(k, d)$. Upon getting the map, key indistinguishability of two key candidates can be verified by comparing all the values in its own key rows mapped by same data. If all of them are the same, they are indistinguishable by definition. Final average key space size can be calculated from counting the number of indistinguishable key candidate set and calculating the size of them. If the attacker has chosen a data and get the value of targeted signal by probing, then the attacker can shrink the table by deleting all the key candidates like Fig. 3 that do not consistent with the probed result. Given a specific key candidate set, the data revealability can be calculated by counting the number of 1 and 0 of one data column in the Key-Data Map.



|  | $d_1$ | $d_2$ | $d_3$ | $d_4$ | ... | $d_i$ | ... | $d_{\|\mathbf{D}\|}$ |
|---|---|---|---|---|---|---|---|---|
| $k_1$ | 1 | 1 | 0 | 1 | ... | 0 | ... | 0 |
| ~~$k_2$~~ | ~~0~~ | ~~0~~ | ~~1~~ | ~~0~~ | ~~...~~ | ~~0~~ | ~~...~~ | ~~0~~ |
| ~~$k_3$~~ | ~~0~~ | ~~1~~ | ~~0~~ | ~~0~~ | ~~...~~ | ~~0~~ | ~~...~~ | ~~0~~ |
| $k_4$ | 1 | 1 | 0 | 1 | ... | 0 | ... | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ~~$k_j$~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~...~~ | ~~1~~ | ~~...~~ | ~~1~~ |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $k_{\|\mathbf{K}\|}$ | 1 | 1 | 0 | 0 | ... | 0 | ... | 1 |

$\xrightarrow{f(k_g, d_1) = 1}$

|  | $d_1$ | $d_2$ | $d_3$ | $d_4$ | ... | $d_i$ | ... | $d_{\|\mathbf{D}\|}$ |
|---|---|---|---|---|---|---|---|---|
| $k_1$ | 1 | 1 | 0 | 1 | ... | 0 | ... | 0 |
| $k_4$ | 1 | 1 | 0 | 1 | ... | 0 | ... | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $k_{\|\mathbf{K}\|}$ | 1 | 1 | 0 | 0 | ... | 0 | ... | 1 |

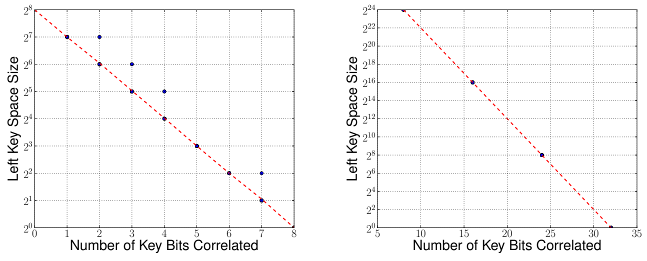
Figure 3: Key-Data Map Shrink

Obviously, the complexity of above calculation procedure is exponentiate to the number of input bits (both key and data). When a signal correlates with large numbers of input bits, it is extremely infeasible to tackle it. We propose to approximate the calculation on a subset of Key-Data Map. This sub-map can be generated by randomly selecting keys and data and calculating the value of targeted signal. To be specific, when we want to determine the final key candidate set, we need to randomly sample key candidates and data patterns. When we want to evaluate the data revealability, only key patterns need to be randomly selected.

## 5. EXPERIMENTAL RESULTS

### 5.1 Experimental Setup

As AES is a widely accepted encryption algorithm, it has been intensively evaluated and analyzed and it is also the often target of various side channel attacks. So we validate the proposed vulnerability metric on an AES design which is obtained from OpenCores website [10]. The AES design is synthesized into netlist by the commercial synthesis tool. Every signal in the netlist would be evaluated in this section. AES [11] is constituted by rounds of transformations to convert plaintext to ciphertext. This AES design is a non-pipeline implementation, and hence the operation of each round adopts the same circuits. As a result, in order to evaluate the vulnerability of signals at the different rounds, we are required to adopt the time-frame expansion technique [12] to construct the circuit that mimics the functionality of previous rounds. For signals that correlates with more than 16 key bits, we use the random sampling technique to approximately calculate them with an constant sample size $2^{16} = 65536$.



(a) Sbox (b) Complete AES (Other than Sbox)

Figure 4: the Size of Left Key Space

We report the results for signals at the first round in the experiment, because signals at the subsequent rounds are driven by a number of key bits and their probing complexities exceed the computation constraint. Moreover, we report results for signals in the Sbox separately, because Sbox is the only non-linear part in AES and it always becomes the target for attackers during the probing attack.

### 5.2 Results and Discussion

First, we present the distribution of signals with different numbers of key bits in Table 2. As can seen from it, signals in Sbox are quite evenly distributed, while most signals in the complete AES are located in the range from 1 to 8, as Sbox is complicated using more logic to implement than other operations.

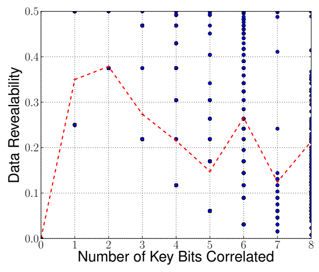
Table 2: The number of signals with different numbers of key bits

| # of Key Bits | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Sbox | 40 | 69 | 102 | 141 | 92 | 119 | 27 | 112 |
| # of Key Bits | $1 \sim 8$ |  | 16 |  | 24 |  | 32 |  |
| Complete AES | 12582 |  | 705 |  | 375 |  | 509 |  |

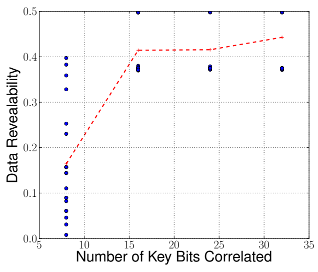
Next, we study the size of key space left after probing a signal, and results are shown in Fig. 4, wherein the point represents the size of key space left while the dash line represents the minimum final key space size for signals with different number of key bits. If one signal is with final key space size of 1, it shall fall on the dashed line. For signals in the Sbox, some lie beyond the dash line, meaning that certain key candidates are indistinguishable with other keys. On the contrary, all other signals in the AES lie on the line, which means their final key space sizes after probing is equal to 1.

Then, we present results of data revealability in Fig. 5. Each point in the figure represents the average data revealability of one signal among all its relevant data patterns. These points are aligned by the number of its correlated key bits. The dash line is the mean value of signals with same number of key bits. The distribution of data revealability indeed varies among signals.

Thereafter, we verify the effectiveness of the proposed approximation technique. We randomly choose 4 signals with 16 key bits and evaluate final key space size and data revealability in different sample size using the approximation shown in Sec. 4. Fig. 6 shows their changes of relative errors. As the sample size increases, both the relative errors of final

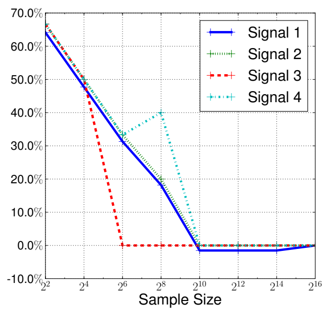


(a) Sbox

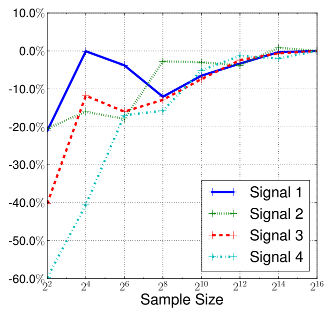


(b) Complete AES (Other than Sbox)

Figure 5: Data Revealability



(a) Final Key Space Size



(b) Data Revealability

Figure 6: Sampling Error

key space size and data revealability converge to zero gradually. As shown, sampling $2^{14}$ is enough to obtain a high accuate approximation for the final key space size and data revealability.

Finally we siumulate the probing attack using Algorithm 1 and validate the proposed vulnerability metric by comparing it with that calculated from simulated probing attack. Fig. 7 shows relationship between the vulnerability calculated by the approximation technique and that calculated from simulated probing attack. As can be seen, the proposed vulnerability has a quite accuracy, which illustrate the effectiveness of the proposed vulnerability metric.

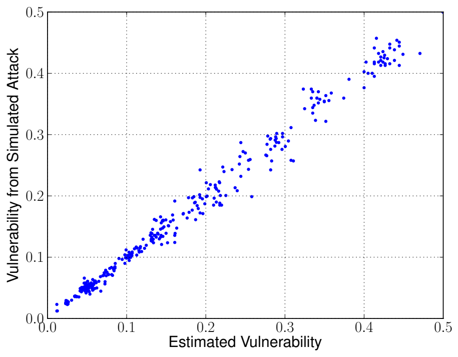


Figure 7: Vulnerability Metric

## 6. CONCLUSION

Probing attack is a severe threat for the security of crypto devices, and it is essential to evaluate its impact at design-time. In this paper, we perform comprehensive study for this problem. To be specific, we a investigate the probing complexity and the key candidate reduction capability for probing attack on every signal in the circuit. We also present approximate solutions for the calculation of the proposed metrics to reduce computational complexity.Finally, our experiments verify the effectiveness of the proposed vulnerability metric on a crypto device.

## 7. ACKNOWLEDGEMENTS

This work was supported in part by NSFC/RGC Joint Research Scheme No. N_CUHK444/12.

## 8. REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology - CRYPTO '99*, pages 388–397, 1999.

[2] O. Aciiçmez. Yet another MicroArchitectural Attack: : exploiting I-Cache. In *Proceedings of the 2007 ACM workshop on Computer Security Architecture*, pages 11–18, 2007.

[3] H. Handschuh, P. Paillier, and J. Stern. Probing Attacks on Tamper-Resistant Devices. In *Cryptographic Hardware and Embedded Systems - CHES*, pages 303–315, 1999.

[4] J. Schmidt and C.-H. Kim. A Probing Attack on AES. In *Information Security Applications, 9th International Workshop, WISA*, pages 256–265, 2009.

[5] Y. Ishai, A. Sahai, and D. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *Advances in Cryptology - CRYPTO 2003*, pages 463–481, 2003.

[6] F.-X. Standaert, T. G. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology - EUROCRYPT 2009*, pages 443–461. 2009.

[7] J. Demme, R. Martin, A. Waksman, and S. Sethumadhavan. Side-channel vulnerability factor: A metric for measuring information leakage. In *39th International Symposium on Computer Architecture ISCA*, pages 106–117, 2012.

[8] T. Zhang, F. Liu, S. Chen, and R. B. Lee. Side channel vulnerability metrics: the promise and the pitfalls. In *The 2nd Workshop on Hardware and Architectural Support for Security and Privacy*, page 2, 2013.

[9] J. Katz and Y. Lindell. *Introduction to Modern Cryptography: Principles and Protocols.* CRC Press, 2007.

[10] R. Usselmann. AES (Rijndael) IP :: Overview :: OpenCores. `http://opencores.org/project,aes_core`, 2013.

[11] J. Daemen and V. Rijmen. AES Proposal: Rijndael. 1999.

[12] F. Fallah. Binary time-frame expansion. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 458–464, 2002.