On Trojan Side Channel Design and Identification

Jie Zhang, Guantong Su†, Yannan Liu, Lingxiao Wei, Feng Yuan, Guoqiang Bai† and Qiang Xu

CUhk REliable Computing Laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong

† Institute of Microelectronics, Tsinghua University

ABSTRACT

Trojan side channels (TSCs) are serious threats to the security of cryptographic systems because they facilitate to leak secret keys to attackers via covert side channels that are unknown to designers. To tackle this problem, we present a new hardware Trojan detection technique for TSCs. To be specific, we first investigate general power-based TSC designs and discuss the tradeoff between their hardware cost and the complexity of the key cracking process. Next, we present our TSC identification technique based on the correlation between the key and the covert physical side channels used by attackers. Experimental results demonstrate the effectiveness of the proposed solution.

1. INTRODUCTION

With increasing demand for secure computation and communication in the era of internet of things (IoT), hardware cryptographic modules are not only widely used in secure applications such as smartcards and set-top boxes, but also proliferate in all sorts of "smart" devices connected to the Internet. As cryptographic hardware provides 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 [1], their implementations suffer from the well-known side-channel attack, i.e., secret information may leak through side channels such as power consumption, timing information and even sound, unless carefully designed and implemented. Side-channel attacks thus become serious industrial concerns and there are significant amount of research efforts spent in designing sophisticated attacks (e.g., differential power attack [2]) and the corresponding countermeasures [3].

Recently, a new type of hardware security threat namely *hardware Trojan* (HT) emerges, which are malicious circuits introduced by adversaries in the design team, third-parties or even foundries to serve as back-doors in the system [4, 5]. Various types of hardware Trojans were presented in the literature with different kinds of malicious functionalities [6]. In particular, for cryptographic hardware, Lin *et al.* [7] proposed the so-called *Trojan side-channel* (TSC) concept that facilitates to leak secret information via covert Trojan-induced side channels and showed that Trojans with size of 14 LUTs can reveal secret keys of an AES core implemented in an FPGA. Liu *et al.* [8] demonstrated a silicon implementation of an AES-based wireless cryptographic chip with embedded TSC and showed it could leak secret keys while passing conventional verification and test procedures.

ICCAD 2014, November 2-6, 2014, San Jose, California, USA.

In [7, 8], the authors also presented potential TSC identification techniques for their specific TSC designs. [7] briefly introduced a potential TSC identification solution for their TSCs, but the details are missing. At the same time, they admitted that the proposed technique would not be able to detect sophisticated TSC designs. In [8], the authors employed existing HT detection techniques based on side channel analysis, such as [9, 10], to differentiate their fabricated TSC-free chips and TSC-infected chips. In practice, however, side channel analysis is often quite difficult, if not impossible, to have known TSC-free chips as golden reference.

The basic idea of TSC design is to embed some circuitries that are closely related to the on-chip secret key, thereby inducing or amplifying the key information leaked via physical side-channels. On the one hand, more direct TSC-induced correlation between the key and the physical side-channels facilitates attackers to extract the secret key easily, but it also enables relatively simple side channel analysis for TSC identification. On the other hand, more sophisticated TSC-induced correlation is hard to detect (without TSC-free chips as golden reference), but the cryptosystem becomes more difficult to break. Consequently, it is interesting and relevant to investigate TSC design tradeoffs and the corresponding identification techniques, which is addressed in this paper.

To be specific, the main contributions of this work include:

- We present a general power-based TSC design methodology and the corresponding key cracking procedure, with arbitrary combination of key bits, plaintext bits and random bits as key information leakage source. We then conduct systematic design tradeoff analysis considering the TSC size, the key cracking complexity via TSC and the TSC stealthiness against the proposed identification technique.
- Leveraging the correlation between TSCs and secret key, we propose a novel TSC identification technique that is applicable to general TSC designs without requiring Trojan-free chips as golden reference, and discuss its detection capability and limitations.

The remainder of this paper is organized as follows. Section 2 presents preliminaries and surveys related work. In Section 3, we present the theoretical study on general TSC designs and the corresponding key extraction procedure. Next, we describe the proposed TSC identification technique in Section 4. Experimental results are then presented in Section 5. Finally, Section 6 concludes this paper.

2. PRELIMINARIES

In this section, we present existing TSC design and identification techniques.

2.1 Trojan Side Channel Design

Side channel attack tries to break a cryptosystem based on information gained from its physical implementation (e.g., power consumption and timing information). For example, a vulnerable smart card

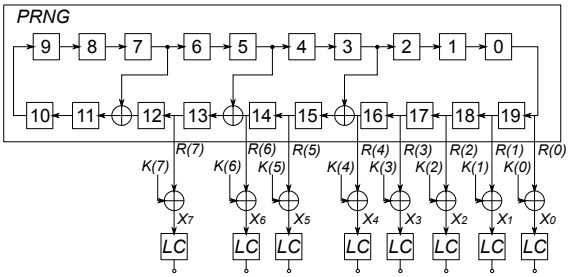

Figure 1: An example TSC design: MOLES [12]

has distinct power consumption when operations are performed with correct secret keys and wrong ones. Differential power attack [2] is able to exploit this property for key extraction even with noisy power traces, thanks to its signal processing and error correction properties. Various types of countermeasures have been proposed to mitigate this severe security threat. For instance, designers could use power analysis-resistant logic (e.g., dual-rail logic) or masking logic when implementing cryptographic hardware [3].

In recent years, hardware Trojans emerge as a serious security threat due to the fact that today's IC designs involve many third-parties during the design and manufacturing process. For example, Skorobogatov *et al.* [11] reported a backdoor found in a military-grade FPGA device. Considering the fact that cryptographic hardware modules are often used as the "root of trust" in a system, they are no doubt the focus of hardware Trojan threats. Lin *et al.* [7] first introduced the TSC concept, which facilitates to leak secret information via covert Trojan-induced side channels. Later, the same authors presented a concrete TSC design namely MOLES in [12]. Gallais *et al.* [13] extended TSC to general-purpose processors on which cryptographic software is executed. Recently, Liu *et al.* presented a silicon implementation of the AES-based wireless cryptographic chip with embedded TSC in [8].

Let us take MOLES [12] introduced in an AES design as an example to illustrate how TSC works (see Fig 1). In this TSC design, the secret key bits, $K(0)$ to $K(7)$, are XORed with a random sequence of $R(0)$ to $R(7)$ generated by a pseudo random number generator (PRNG[1]), represented as $X_i = K(i) \oplus R(i)$. X_i then drives a leakage circuit (LC) to leak the key bit information with additional power consumption P_{0-1} whenever X_i has 0-to-1 transition. Therefore, the total power of MOLES-infected chip, denoted by P_{total}, can be modeled as the sum of the power consumption of MOLES circuitries denoted by P_{MOLES}, the power consumption of AES denoted by P_{AES} and other power consumption that can be modeled as white Gaussian noise (AWGN) denoted by P_{AWGN},

$$P_{total} = P_{MOLES} + P_{AES} + P_{AWGN}. \quad (1)$$

P_{MOLES} can be further represented as:

$$P_{MOLES} = \sum_{i=0}^{7} P_{K(i)} + P_{PRNG}, \quad (2)$$

where $P_{K(i)}$ and P_{PRNG} represent the power consumption of the LC connecting $K(i)$ and that of the PRNG.

During the key cracking process, attackers extract one key bit at a time with differential power analysis (DPA). Let us start by guessing the value of $K(0)$. With the guessed value of $K(0)$ and known $R(0)$, attackers can calculate $X_0 = K(0) \oplus R(0)$. According to the transition of X_0, power traces [2] are grouped into group 0 associated with 0-to-1 transition and group 1 associated with 1-to-0 transition. Suppose there are m_0 power traces in group 0 and m_1 power traces in group 1.

The differential mean power is calculated by the mean power in group 0 minus the mean power in group 1, given by

$$DP \approx \frac{1}{m_0} \sum_{\forall j \in grp0} \sum_{i=0}^{7} P_{(j,K(i))} - \frac{1}{m_1} \sum_{\forall j \in grp1} \sum_{i=0}^{7} P_{(j,K(i))}, \quad (3)$$

where $P_{(j,K(i))}$ denotes the power consumption caused by $K(i)$ in the power trace j. In Eq. 3, P_{PRNG} and P_{AWGN} are canceled with sufficient power traces, since they are not correlated with key bits; P_{AES} can be safely ignored as well because designers would minimize the correlation between key bits with normal side-channel signals (otherwise the key bit can be already cracked even without HTs).

Now, let us first consider the differential mean power caused by $K(0)$, denoted by $DP_{K(0)}$, given by

$$DP_{K(0)} \approx \frac{1}{m_0} \sum_{\forall j \in grp0} P_{(j,K(0))} - \frac{1}{m_1} \sum_{\forall j \in grp1} P_{(j,K(0))}. \quad (4)$$

If $K(0)$ is correctly guessed, each power trace in group 0 consumes the power of P_{0-1} while all power traces in group 1 do not consume any power. Thus, $DP_{K(0)} \approx P_{0-1} > 0$. If $K(0)$ is wrongly guessed, on the contrary, all power traces in group 0 do not consume any power while each power trace in group 1 consume the power of P_{0-1}. Thus, $DP_{K(0)} \approx -P_{0-1} < 0$.

In terms of differential mean power caused by other key bits $K(i)$ (where $i \neq 0$), let m_{i_0} and m_{i_1} be the number of power traces associated with 0-to-1 transition of X_i in group 0 and that in group 1. As a result, $DP_{K(i)}$ is

$$DP_{k_i} = \frac{m_{i_0}}{m_0} P_{0-1} - \frac{m_{i_1}}{m_1} P_{0-1} = (\frac{m_{i_0}}{m_0} - \frac{m_{i_1}}{m_1}) P_{0-1}. \quad (5)$$

Since grouping based on $K(0)$ is uncorrelated with other key bits, we would have $m_{i_0} \approx \frac{1}{2} m_0$ and $m_{i_0} \approx \frac{1}{2} m_1$ with sufficient power traces. As a result, $DP_{K(i)} \approx 0$.

With the above, $DP = \sum_{i=0}^{7} DP_{K(i)} > 0$ if $K(0)$ equals the guessed value; otherwise, $K(0)$ equals the opposite value of the guessed one. Other key bits can be extracted similarly with the above procedure.

2.2 Trojan Side Channel Identification

A number of HT detection techniques have been proposed in the literature and they can be broadly categorized into two categories: trust verification techniques [14, 15] used to detect HTs inserted at the design stage and side channel analysis techniques [9, 10] used to detect HTs inserted during fabrication. Generally speaking, trust verification techniques perform HT detection by identifying the rare trigger signals used in an HT. However, TSC designs are often "always-on" without any dedicated trigger signals and hence cannot be caught by these solutions. Liu *et al.* [8] adopted side channel analysis technique to successfully differentiate their fabricated TSC-free and TSC-infected chips. However, in practice, it is often quite difficult to obtain TSC-free ICs as golden reference.

The above techniques are for general HT detection. In [7], the authors briefly introduced a dedicated TSC identification technique for their proposed TSC designs, but the details were not presented. While effective for their specific TSC designs, it was mentioned in [7] that their method was not applicable for more sophisticated TSC designs that leak secret information via a complex combination of multiple key bits, plaintext bits and random bits.

HT design and HT detection are like arms race, wherein attackers constantly update their tactics to intrude a system while defenders respond with more security measures to protect the system. Existing TSC designs and the corresponding identification techniques such as [7, 8] are case studies that open the horizon of new security concerns for cryptographic hardware, and it is essential to investigate whether there are other forms of TSC designs and how to defend against them, if any. This has motivated the study of general TSC design methodology and the corresponding TSC identification technique in this work.

[1] The PRNG is used here to distribute the leakage of key information to multiple clock cycles, and it is only known by attackers.

[2] Power trace is a sequence of sampled power values.

3. GENERAL POWER-BASED TSC DESIGN

In this section, we provide the theoretical study on the general power-based TSC design methodology as well as the corresponding key cracking process by using TSC.

Before discussing the details, we have the following symbol definitions as listed in Table 1. Let K be the key variable, wherein $K(i)$ is the i^{th} bit of K, and k be the realization of the key variable. Let $\mathbf{K} = \{0,1\}^{n_k}$ be the whole key space where n_k is the number of key bits of K, and hence $k \in \mathbf{K}$. Let K^* be the variable of the subkey composed of certain key bits of K, wherein $K^*(i)$ is the i^{th} bit of K^*, and k^* be the realization of K^*. Let $\mathbf{K}^* = \{0,1\}^{n_{k^*}}$ be the key space of K^* where n_{k^*} is the number of key bits of K^*, and hence $k^* \in \mathbf{K}^*$. Let R be the random number, wherein $R(i)$ is the i^{th} bit of R, and r be the realization of R. Let $\mathbf{R} = \{0,1\}^{n_r}$ be the space of R where n_r is the number of bits of R, and hence $r \in \mathbf{R}$.

3.1 TSC Architecture

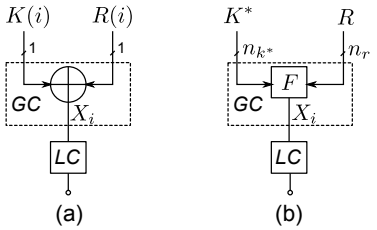

Figure 2: (a) LM used by MOLES; (b) The general LM

Typically, TSC contains multiple leakage modules (LMs) which are used together to crack the whole key. Each of LM, serving as a separate channel to leak key bits via power side channel, is composed of two components, the leakage information generation circuit (GC) and the physical leakage circuit (LC), as shown in Fig. 2.

To be specific, GC generates the actual logic leakage information that is the key masked by a random number. In case of the ease presentation, we model the plaintext bit as a special random bit. Fig. 2 (a) presents the LM of MOLES that leaks one key bit XORed one random bit while Fig. 2 (b) shows the general LM that leaks multiple key bits masked by multiple random bits. Thus, the actual leakage information, denoted by X, can be represented by

$$X = F(K^*, R), \text{ where } X \in \{0,1\}. \tag{6}$$

Previous work treated TSC secure based on the fact that only attackers know how to recover the key with knowledge of F and R. Thus, if LM is driven by multiple key bits and multiple random bits, it becomes much more difficult for designers to determine F and R. In Section 4.3, we discuss the stealthiness of TSC. We assume that LC can output only one power value, denoted by P_{LC}, and whether it consumes the power is determined by the value or the transition of the input of LC, represented by X in Eq. 6, given by

$$P(X) = P(F(K^*, R)). \tag{7}$$

In case of the ease presentation, in the following, we consider that LC consuming the power depends on the value of the input, unless otherwise specified. The proposed general TSC design and corresponding detection technique are easily extended to TSC whose LC consuming the power depends on the transition of the input. The design of LC is out of scope of this paper.

Based on Eq. 6 and Eq. 7, we model LM's behavior as leakage power matrix, which is defined as follow:

DEFINITION 1. *Leakage power matrix for LM, $TP_{2^{n_{k^*}} \times 2^{n_r}}$, describing the leakage power of LM under different values of K^* and R. If LM consumes the power under k^* and r, we set $TP(k^*, r) = 1$; otherwise we set $TP(k^*, r) = 0$.*

For a specific key k^*, we use TP_{k^*}, defined as the row of TP wherein $K^* = k^*$, to denote its leakage power pattern. A simple example for leakage power matrix TP can be found in Fig. 3.

Table 1: List of Notation

Symbol	Meaning
K	The key variable
$K(i)$	The i^{th} bit of K
k	The realization of K
$\mathbf{K}$	The key space $\mathbf{K} = \{0,1\}^{n_k}$, $k \in \mathbf{K}$
n_k	The number of key bits of K
K^*	The subkey variable and all key bits belong to K
$K^*(i)$	The i^{th} bit of K^*
k^*	The realization of K^*
$\mathbf{K}^*$	The subkey space $\mathbf{K}^* = \{0,1\}^{n_{k^*}}$, $k^* \in \mathbf{K}^*$
n_{k^*}	The number of key bits of K^*
R	The random number
$R(i)$	The i^{th} bit of R
r	The realization of R
$\mathbf{R}$	The random number space $\mathbf{R} = \{0,1\}^{n_r}$, $r \in \mathbf{R}$
n_r	The number of random bits of R
$k^\diamond$	The actual key value of K used by the design
$k^{*\diamond}$	The actual key value of K^* driving a LM
n_{lm}	The number of leakage modules

By embedding TSC with n_{lm} LMs, attackers have

$$\begin{cases} F_1(K_1^*, R_1) & = X_1, \\ F_2(K_2^*, R_2) & = X_2, \\ \quad \vdots & \quad \vdots \\ F_{lm}(K_{lm}^*, R_{lm}) & = X_{lm}, \end{cases}, \tag{8}$$

with the knowledge of the implementation of LMs represented by $F(K^*, R)$ and $P(X)$ as well as the random number R generated by PRNG. Next, we detail how to crack the entire key via embedded TSC.

3.2 Key Cracking Procedure

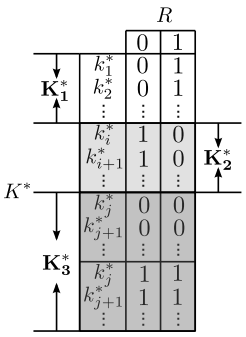

Figure 3: An example to illustrate leakage power matrix for an LM, wherein each row represents TP_{k^*} and $n_r = 1$

The key cracking process is illustrated as follows. First, attackers are assumed to have the ability to measure the power of the chip and collect a large number of power traces under the different plaintexts. Next, with these power traces, attackers calculate one specific set of key candidates for each LM, denoted by $\mathbf{K_1}, \mathbf{K_2}, \ldots, \mathbf{K_m}$, with the knowledge of the implementation of each LM by demodulating techniques, such as DPA. Finally, by intersection of all sets of key candidates, given by

$$\mathbf{K_f} = \mathbf{K_1} \cap \mathbf{K_2} \cap \cdots \cap \mathbf{K_m}, \tag{9}$$

attackers are able to reduce the key candidates into a reasonable size for brute force or directly obtain the genuine key.

For a particular LM (LM_i), the procedure of calculating the key candidate set involves two steps. Let us take the leakage power matrix shown in Fig. 3 to illustrate the details.

The first step is to apply DPA to identify the leakage power pattern, $TP_{k^{*\diamond}}$ where $k^{*\diamond}$ denotes the actual value of K^*. As attackers hold the knowledge of R, the identification process is achieved by grouping the power traces based on R value and calculating the mean power difference between $R=0$ group and $R=1$ group, denoted as DP_{0-1}. According to the value of DP_{0-1}, we can obtain $TP_{k^{*\diamond}}$ as follows.

- If $DP_{0-1}<0$, we have $TP_{k^{*\diamond}}=[0,1]$.
- If $DP_{0-1}>0$, we have $TP_{k^{*\diamond}}=[1,0]$.
- If $DP_{0-1}\approx 0$, we have $TP_{k^{*\diamond}}=\{[0,0],[1,1]\}$.

Conventional DPA is unable to differentiate $TP_{k^*}=[1,1]$ and $TP_{k^{*\diamond}}=[0,0]$, since both of them lead to $DP_{0-1}\approx 0$. Power effects of AES, PRNG and AWGN are assumed to be removed by DPA with sufficient power traces as discussed in Section 2. The power impacts of other LMs can be removed by DPA as well, since all power traces containing the power of other LMs are evenly distributed into two groups associated with $R=0$ and $R=1$ for this LM. This factor is to be illustrated by *Lemma* 1 in Section 3.3.

For the TSC driven by n_r random bits, $TP_{k^{*\diamond}}$ can be determined in the similar way by calculating the differential mean power of power traces with different random numbers. However, DPA is unable to differentiate $TP_{k^{*\diamond}}=[1,1,\ldots,1]$ and $TP_{k^{*\diamond}}=[0,0,\ldots,0]$ as well.

The second step is to extract the key candidate set. With the identified leakage power pattern $TP_{k^{*\diamond}}$. The extraction key process can be done by simply finding the key that leads to the same leakage power pattern. As a result, we obtain the key candidate set for LM_i denoted by $\mathbf{K_i^*}$, wherein $\forall k^*\in\mathbf{K_i^*}$, $TP_{k^*}=TP_{k^{*\diamond}}$, as shown in Fig. 3.

In order to intersect the key candidate sets obtained by LMs, we define $\mathbf{K_i}$ based on $\mathbf{K_i^*}$ and $\mathbf{K_i}\subseteq\mathbf{K}$. For any $k\in\mathbf{K_i}$, the subkey of k, denoted by k^*, belongs to $\mathbf{K_i^*}$, given by $k^*\in\mathbf{K_i^*}$. Correspondingly, we define $\mathbf{K_1}$, $\mathbf{K_2}$, …, and $\mathbf{K_m}$.

3.3 Leakage Module Design

The key issue in a TSC design is the LM design, and hence we discuss its details in this subsection.

During the key cracking process for one LM, the main point is to remove the power impact of other LMs by DPA. Therefore, LM should be designed in such a manner that satisfies the following lemma.

LEMMA 1. *For $\forall k^*\in\mathbf{K}^*$, if we randomly select r where $r\in\mathbf{R}$, we have*

$$\Pr[TP_{k^*}(r)=1]=\Pr[TP_{k^*}(r)=0]=\frac{1}{2}. \tag{10}$$

Lemma 1 enables power trace containing the power of one LM be evenly distributed into groups when power traces for this LM are randomly allocated.

According to the above key cracking procedure, we observe that attackers in fact adopt LM to distribute all key candidates into predefined several key sets. For a general LM, the number of available key sets is given by the following lemma.

LEMMA 2. *Consider an LM driven by n_{k^*} key bits and n_r random bits. This LM is able to distribute the $2^{n_{k^*}}$ key candidates into at most N_{KS} key candidate sets, where N_{KS} is given by*

$$N_{KS}=\text{Min}\,\{2^{(2^{n_r})}-1,2^{n_{k^*}}\}. \tag{11}$$

PROOF. For this LM driven by n_{k^*} key bits and n_r random bits, the size of $TP_{k^{*\diamond}}$ is 2^{n_r}. Therefore, there are at most $2^{(2^{n_r})}$ possible values for $TP_{k^{*\diamond}}$. DPA is able to differentiate $2^{(2^{n_r})}-1$ of them except $TP_{k^{*\diamond}}=[0,0,\ldots,0]$ and $TP_{k^{*\diamond}}=[1,1,\ldots,1]$. Therefore, this LM is able to distribute $2^{n_{k^*}}$ key candidates into Min $\{2^{(2^{n_r})}-1,2^{n_{k^*}}\}$ sets. ∎

Attackers are always expecting to reduce as many key candidates as possible by LM, and the following lemma indicates how to obtain the minimum expected number of key candidates.

LEMMA 3. *Consider an LM driven by n_{k^*} key bits and n_r random bits. The minimum expected number of key candidates after reducing by this LM is given by $\frac{1}{N_{KS}}2^{n_{k^*}}$, where $N_{KS}=\text{Min}\,\{2^{(2^{n_r})}-1,2^{n_{k^*}}\}$.*

PROOF. Let us suppose LM partitions the key space $\mathbf{K}^*$ into N_{KS} subsets, denoted by $\mathbf{K_1^*}$, $\mathbf{K_2^*}$, …, $\mathbf{K_{N_{KS}}^*}$. Let n, n_1, n_2, …, $n_{N_{KS}}$ be the size of $\mathbf{K}^*$, $\mathbf{K_1^*}$, $\mathbf{K_2^*}$, …, $\mathbf{K_{N_{KS}}^*}$, where $n=\sum_{i=1}^{N_{KS}}n_i=2^{n_{k^*}}$. Since the actual key $k^{*\diamond}\in\mathbf{K}^*$ is randomly chosen by designers, the probability of $k^{*\diamond}$ within $\mathbf{K_i^*}$ is given by

$$\Pr[k^{*\diamond}\in\mathbf{K_i^*})]=\frac{n_i}{n}. \tag{12}$$

With the above, the expected number of key candidates left by this LM, denoted by $E(n_{kc})$ is calculated by

$$E(n_{kc})=\sum_{i=1}^{N_{KS}}(\Pr[k^{*\diamond}\in\mathbf{K_i^*}]\cdot n_i)=\sum_{i=1}^{N_{KS}}(\frac{n_i^2}{n}). \tag{13}$$

By minimizing $E(n_{kc})$, we have

$$\text{Min}(E(n_{kc}))=\frac{1}{N_{KS}}n,\ \text{ when } n_1=n_2=\cdots=\frac{1}{N_{KS}}n. \tag{14}$$

Therefore, attackers are required to evenly distribute keys into N_{KS} sets in case of the minimum expected number of key candidates. With the above, *Lemma* 3 is proved. ∎

With *Lemma* 3, it is possible to estimate the number of LMs required to crack the whole key, denoted by n_{lm}. Suppose the acceptable complexity of enumerating all remaining key candidates is 2^{n_b} and assume every LM is perfectly designed to expect to shrink the key space by $\frac{1}{m}$ according to *Lemma* 3, where $m\le N_{KS}=\text{Min}\,\{2^{(2^{n_r})}-1,2^{n_{k^*}}\}$. Therefore, we have

$$2^{n_b}=2^{n_k}\cdot\prod_{i=1}^{n_{lm}}\frac{1}{m_i}. \tag{15}$$

To estimate n_{lm} by Eq. 15, we simply consider $m_i=2^{(2^{n_r})}-1$ and $m_i=2^{n_{k^*}}$, assuming attackers would fully use each LM, and consider n_r and n_{k^*} are constant for all LMs. Therefore, for $m_i=2^{(2^{n_r})}-1$, we have

$$n_{lm}=\frac{n_k-n_b}{log_2(2^{(2^{n_r})}-1)}; \tag{16}$$

for $m_i=2^{n_{k^*}}$, we have

$$n_{lm}=\frac{n_k-n_b}{n_{k^*}}. \tag{17}$$

3.4 Overhead and Complexity

The total area cost of TSC, denoted by C_{TSC}, comes from GC denoted by $C_{GC}$and LC denoted by C_{LC} and is estimated by

$$C_{TSC}=\sum_{i=1}^{n_{lm}}\cdot C_{GC}(n_{k_i^*}+n_{r_i})+C_{LC_i}. \tag{18}$$

$C_{GC}(n_{k_i^*}+n_{r_i})$ approximately increases exponentially with the increase of $(n_{k_i^*}+n_{r_i})$ caused by implementing $F(K^*,R)$. C_{LC} is determined by attackers in the consideration of the ratio between the power of LM and the power of the whole chip. In terms of C_{GC}, the best choice of TSC is to set $n_{k_i^*}=1$ and $n_{r_i}=1$, just like MOLES.

The complexity to build TSC and crack the key in all is given by

$$O(\sum_{i=1}^{n_{lm}}2^{n_{k_i^*}+n_{r_i}})+O(n_{lm}\cdot NPT)+O(\sum_{i=1}^{n_{lm}}2^{n_{r_i}})+O(2^{n_b}). \tag{19}$$

The first part, $O(\sum_{i=1}^{n_{lm}}\cdot 2^{n_{k_i^*}+n_{r_i}})$, denotes the complexity of building n_{lm} LMs, wherein we assume that attackers have to go through all input combinations ($2^{n_{k_i^*}+n_{r_i}}$) to design an efficient LM. The second part, $O(n_{lm}\cdot NPT)$, denotes the complexity to collect sufficient power traces, where NPT is the number of power traces for cracking one

LM. The third part, $O(\sum_{i=1}^{n_{lm}} 2^{n_{r_i}})$, denotes the complexity of identifying $2^{n_{r_i}}$ elements of $TP_{k^{*\circ}}$. The fourth part, $O(2^{n_b})$, denotes the complexity of obtaining the key from the key set in a brute force manner.

With the above, to obtain the minimum complexity building TSC of cracking the key, attackers are suggested to set $n_{k_i^*} = 1$ and $n_{r_i} = 1$, just like MOLES. In other words, MOLES has the lowest hardware cost and the lowest key cracking complexity. However, it is also easy to be detected by our proposed TSC identification technique, as detailed in Section 4.

4. THE PROPOSED TSC IDENTIFICATION SOLUTION

In this section, we present the proposed TSC identification solution. We first illustrate the existence of the correlation between the key and the leaked power via TSC-induced side channels. Next, we present how to identify such correlation for TSC detection. Finally, we estimate the detection capability of our approach and discuss its limitations.

4.1 Observation

The observation used to detect TSC is that all key bits driving one LM are correlated with the leaked power. Next, we describe the existence of this correlation.

Consider an LM that is driven by K^* and R. From the perspective of designers, power traces are generated by the same plaintext but different key values and collected at the same time spot, guaranteeing different keys are masked by the same R value. After collecting power traces, we group them according to the value of K^*, and power traces in the same group are generated by the same value of K^*.

Among all groups of power traces, we choose two groups, group x and group y, and k_x^* and k_y^* denote values of key bits for group x and group y. These two groups should satisfy the requirement of $F(k_x^*, r) \neq F(k_y^*, r)$, and there must exist required group x and group y according to the design of LM discussed in Section 3.

We calculate the differential mean power of these two groups of power traces, given by

$$DP_{(grp_x, grp_y)} = \frac{1}{n_x} \sum_{\forall j \in grp_x} \sum_{i=1}^{n_{lm}} P_{(j,K_i^*)} - \frac{1}{n_y} \sum_{\forall j \in grp_y} \sum_{i=1}^{n_{lm}} P_{(j,K_i^*)}, \quad (20)$$

where n_x and n_y denotes the numbers of power traces in group x and group y, and $P_{(j,K_i^*)}$ denotes the power caused by the LM driven by K_i^* in the power trace j. With the assumption used in TSC, P_{AES}, P_{PRNG} and P_{AWGN} in Eq. 20 are removed by DPA.

To calculate $DP_{(grp_x, grp_y)}$, let us consider the differential mean power caused by each LM separately. For the targeted LM driven by K^*, the differential mean power caused by this LM, denoted by $DP_{(grp_x, grp_y)}(K^*)$, is given by

$$DP_{(grp_x, grp_y)}(K^*) = \frac{1}{n_x} \sum_{\forall j \in grp_x} P_{(j,K^*)} - \frac{1}{n_y} \sum_{\forall j \in grp_y} P_{(j,K^*)}. \quad (21)$$

Since $F(k_x^*, r) \neq F(k_y^*, r)$, $DP_{(grp_x, grp_y)}(K_x^*) \neq 0$.

For any other LM driven by K_z^*, let us define $n_{(x,z_0)}$, $n_{(x,z_1)}$, $n_{(y,z_0)}$ and $n_{(y,z_1)}$ be the number of power traces in group x whose k^* makes $F_z(k^*, r) = 0$, the number of power traces in group x whose k^* makes $F_z(k^*, r) = 1$, the number of power traces in group y whose k^* makes $F_z(k^*, r) = 0$, and the number of power traces in group y whose k^* makes $F_z(k^*, r) = 1$. The differential mean power caused caused by this LM, denoted by $DP_{(grp_x, grp_y)}(K_z^*)$, is given by

$$\begin{aligned} DP_{(grp_x, grp_y)}(K_z^*) &= \frac{1}{n_x}(n_{(x,z_0)}P(0) + n_{(x,z_1)}P(1)) \\ &\quad - \frac{1}{n_y}(n_{(y,z_0)}P(0) + n_{(y,z_1)}P(1)), \quad (22) \\ &= (\frac{n_{(x,z_0)}}{n_x} - \frac{n_{(y,z_0)}}{n_y})P(0) + (\frac{n_{(x,z_1)}}{n_x} - \frac{n_{(y,z_1)}}{n_y})P(1) \end{aligned}$$

where $P(0)$ and $P(1)$ are calculated from Eq. 7. Since the grouping based on K^* is not correlated with the distribution of $n_{(x,z_0)}$, $n_{(x,z_1)}$, $n_{(y,z_0)}$ and $n_{(y,z_1)}$, we would have $n_{(x,z_0)} \approx n_{(x,z_1)} \approx n_{(y,z_0)} \approx n_{(y,z_1)} \approx \frac{1}{2}n_x \approx \frac{1}{2}n_y$ with sufficient power traces. Therefore, we can obtain $DP_{(grp_x, grp_y)}(K_z^*) \approx 0$.

By considering the power caused by all LMs together, we have $DP_{(grp_x, grp_y)} = \sum_{i=1}^{n_{lm}} DP_{(grp_x, grp_y)}(K_i^*) \neq 0$, which indicates the existence of the correlation between key bits (K^*) and leaked power. This observation enables us to detect TSC by identifying such correlation.

4.2 TSC Detection Algorithm

The proposed TSC detection method is based on identifying the correlation between key bits and leaked power. However, the design of TSC discussed above, in fact, is intended to hide such correlation from two aspects. To be specific, on the one hand, the key information is masked by the random number; on the other hand, the correlation between which key bits and the leaked power is unknown to designers.

It is relatively easy to overcome the impact of the random number by collecting power traces masked with the same random number. To achieve this, we can sample the power under the *same* plaintext at the *same* time spot while varying keys. However, without knowledge of which key bits are selected to drive the leakage module, designers are required to try all key bit combinations to identify TSC in the worst case scenario. In the following, we present that designers could have a high detection probability with only a few attempts.

Algorithm 1: TSC Detection Algorithm

```
1  Set N_k*, N_v and N_t;
2  while 1 → N_v do
3  |   Randomly choose K*;
4  |   IdentifyLeakageModule(K*);
5  |   if Any LM detected then
6  |   |   Stop;
7  |   end if
8  end while
9  IdentifyLeakageModule(K*)
10 |   while 1 → N_t do
11 |   |   Randomly choose group x and group y in K*;
12 |   |   if DP_(grp_x,grp_y)(r) ≠ 0 then
13 |   |   |   Detect the leakage module and return;
14 |   |   end if
15 |   end while
16 end
```

4.2.1 Overall Algorithm

Algorithm 1 illustrates the detection algorithm for TSC. We start to set the number of key bits verified, N_{k^*}, and the number of processes to identify LM, N_v. How to set N_{k^*} and N_v is to be detailed in Section 4.3. Then, for each identifying leakage module process, we randomly select key bits K^* from all key bits K to verify whether they drive any LM, denoted by *IdentifyLeakageModule*(K^*), and the number of key bits for K^* is equal to N_{k^*}. Whenever any LM is detected by a selected K^*, the algorithm stops. In the following, we first present the process of identifying LM, and then discuss the detection capability of the proposed detection algorithm as well as the limitations.

4.2.2 Identifying Leakage Module

Based on the observation in Section 4.1, we formulate the problem of identifying LM as identifying the correlation between given key bits and leaked power. The process is illustrated by Line 9-16 in Algorithm 1. We group power traces based on the value of K^*, and we try N_t pairs of groups of power traces. N_t is determined, guaranteeing a high detection probability. For each pair of groups of power traces,

we calculate their differential mean power. If there is any strange differential mean power, we consider that there are LMs embedded in the chip.

However, it is possible that designers choose key bits K^* that drive no LM, one LM or multiple LMs. Let us discuss these cases one by one in the following.

- *K^* does not contain all key bits of any LM.* For this case, we cannot detect any of LMs. This is because that power traces containing the power of LMs are evenly distributed among groups based on the value of K^* according to *Lemma* 1, which makes $DP_{(grp_x,grp_y)} \approx 0$.

- *K^* contains all key bits of one LM.* For this case, We are able to detect this LM whenever we choose two groups of power traces whose keys, k_x^* and k_y^*, satisfies the requirement of $F(k_x^*,r) \neq F(k_y^*,r)$ according to Section 4.1.

  Assume groups are independent, and according to *Lemma* 1, the probability of detecting this leakage module is given by

$$\begin{aligned}\Pr[DP_{(grp_x,grp_y)} \neq 0] &= \Pr[TP_{k_x^*}(r) \neq TP_{k_y^*}(r)] \\ &= \Pr[TP_{k_x^*}(r) = 1 \text{ and } TP_{k_y^*}(r) = 0] + \\ &\quad \Pr[TP_{k_x^*}(r) = 0 \text{ and } TP_{k_y^*}(r) = 1] \\ &= \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{1}{2}.\end{aligned} \tag{23}$$

- *K^* contains all key bits of one LM and some extra key bits.* For this case, we are able to detect this LM and those extra key bits would not have influence on identifying this LM. Let K_1^* be all key bits driving this LM. Thus, this LM will be detected whenever two groups of power traces whose keys, k_{1x}^* and k_{1y}^*, satisfies $TP_{k_{1x}^*}(r) \neq TP_{k_{1y}^*}(r)$. Since key bits are independent, the probability of selecting key bit value from $\mathbf{K}^*$ and $\mathbf{K_1^*}$ belonging to any key set is the same. As a result, we have

$$\Pr[DP_{(grp_x,grp_y)} \neq 0] = \Pr[TP_{k_x^*}(r) \neq TP_{k_y^*}(r)] = \frac{1}{2}. \tag{24}$$

- *K^* contains all key bits of more than one LMs.* For this case, we are able to detect LMs. Suppose there are n_l LMs. To detect them, we have to find two groups of power traces, group x and group y, satisfying $DP_{(grp_x,grp_y)} \neq 0$. Suppose LMs consume the same power P_{LC} when activated. Thus, for any $k^* \in \mathbf{K}^*$, we have n_l possible power values caused by these n_l LMs, $n_l P_{LC}$, $(n_l-1)P_{LC}$, ..., P_{LC} and 0. With the above, there must exist two groups make n_l LMs consume different power.

  Next, let us discuss the probability of n_l LMs consuming different power under k_x and k_y with the same r. For any $k^* \in \mathbf{K}^*$, the probability of n_l LMs consuming tP_0 in all is given by $C_{n_l}^t \frac{1}{2}^{n_l}$. Therefore, the probability of k_x and k_y making n_l LMs consume different power is given by

$$\Pr[DP_{(grp_x,grp_y)} \neq 0] = 1 - \sum_{i=0}^{n_l} (C_{n_l}^i \frac{1}{2}^{n_l})^2 \geq \frac{1}{2}, \tag{25}$$

  which can be proven to be greater than or equal to $\frac{1}{2}$.

- *K^* contains all key bits of more than one LMs and some extra key bits.* For this case, we are able to detect LMs and those extra key bits would not have influence on detecting LMs. This is easily proven by above discussion.

With the above, we can conclude that if K^* contains all key bits of any LMs, our approach is able to detect them whenever two chosen groups of power traces have clear and stable differential mean power.

We try N_t pairs of groups of power traces in order to have a high detection probability. With N_t tries, the probability of detecting LMs is given by

$$1 - (1 - \Pr[DP_{(grp_x,grp_y)} \neq 0])^{N_t} > P_{confidence}. \tag{26}$$

As a result, N_t is determined according to user-defined confidence, denoted by $P_{confidence}$. Since $\Pr[DP_{(grp_x,grp_y)} \neq 0] \geq \frac{1}{2}$ according to Eq. 23, Eq. 24 and Eq. 25, we can have about 99.9% probability to detect LM with $N_t = 10$ if K^* contains all key bits of any of the LMs.

Finally, the complexity of the proposed TSC detection algorithm is approximately given by

$$O(N_v \times N_t \times NPT), \tag{27}$$

where NPT denotes the number of power traces required.

4.3 Detection Capability

To estimate the detection capability of our approach by Algorithm 1, let us first summarize the TSC detection problem as follows. Let n_k be the number of key bits. Attackers would like to build N_{lm} LMs for TSC. Each LM is driven by n_{k^*} different key bits. Designers, on the contrary, expect to detect any of LMs by attackers. The detection flow is like this. Each time, designers randomly select N_{k^*} key bits from n_k key bits. If n_{k^*} key bits of one of LMs are inside of N_{k^*} key bits, TSC is detected by designers. Thus, if designers try above process N_v times, the probability of detecting TSC is given by

$$P_{DeTSC}(N_v, N_{k^*}, n_{lm}, n_{k^*}) = \begin{cases} 1 - \prod_{i=1}^{N_v} \prod_{j=1}^{C_{N_{k^*}}^{n_{k^*}}} (1 - \frac{n_{lm}}{C_{n_k}^{n_{k^*}} - (i-1)C_{N_{k^*}}^{n_{k^*}} - j}), & N_{k^*} \geq n_{k^*} \\ 0, & N_{k^*} < n_{k^*} \end{cases}, \tag{28}$$

where $P_{DeTSC}(N_v, N_{k^*}, n_{lm}, n_{k^*})$ denotes the probability to detect TSC. If $N_{k^*} < n_{k^*}$, our approach definitely misses TSC as discussed. If $N_{k^*} \geq n_{k^*}$, $P_{DeTSC}(N_v, N_{k^*}, n_{lm}, n_{k^*})$ is influenced by N_v, N_{k^*}, n_{lm} and n_{k^*}. Therefore, let us discuss the detection capability of our approach from two aspects, the impact of n_{lm} and n_{k^*} set by attackers and the impact of N_v and N_{k^*} set by designers.

Table 2: Parameters of Cases

	Case 1	Case 2	Case 3	case 4
n_k	128	128	128	128
n_{k^*}	**4-8**	4	4	4
n_{lm}	80	**80-40**	80	80
N_{k^*}	10	10	**10-14**	10
N_v	500	500	500	**500-900**

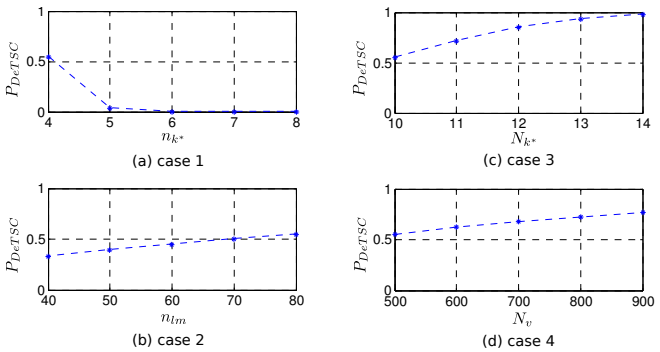

Figure 4: P_{DeTSC} under four cases

We study four cases shown in Table 2 and present the estimated P_{DeTSC} in Fig. 4. From the perspective of attackers, we study cases 1 and case 2 where we vary n_{k^*} and n_{lm}. As can be seen, P_{DeTSC} decreases dramatically by increasing n_{k^*} than by decreasing n_{lm}. This is because that the number of key bit combinations that can be used to build LM ($C_{n_k}^{n_{k^*}}$) increases significantly with the increase of n_{k^*}, which

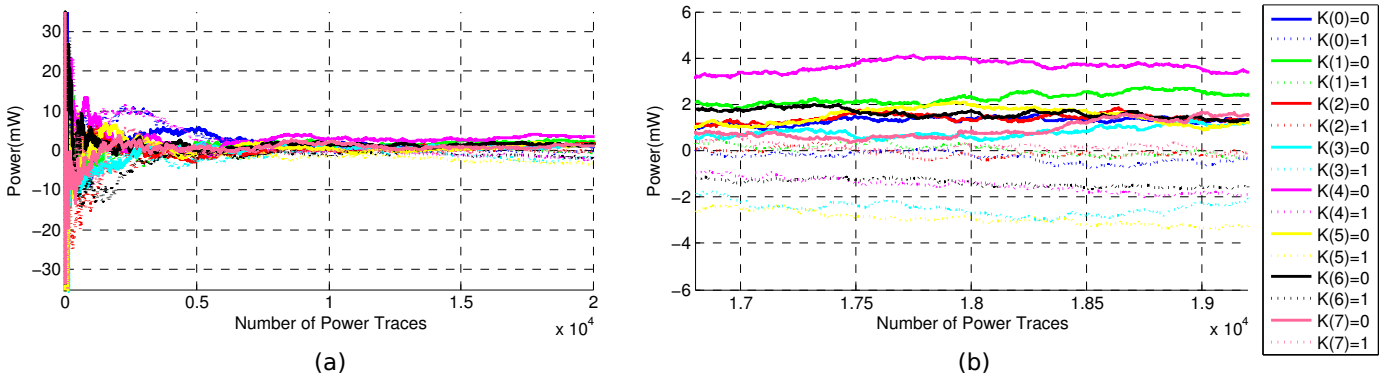

Figure 5: Power curves indicating the detection of TSC_1 with $N_{k^*} = 1$ and the first eight key bits chosen separately

increases the detection complexity. Therefore, the LM driven by multiple key bits is more difficult to be detected, while the hardware cost of LM and the complexity of cracking the key would increase exponentially with n_{k^*}, shown in Eq. 18 and Eq. 19.

From the perspective of designers, we study case 3 and case 4 where we vary N_{k^*} and N_v. We find that it is better to increase N_{k^*} to improve P_{DeTSC} than to increase N_v. This is because more key bit combinations ($C_{N_{k^*}}^{n_{k^*}}$) are verified once with larger N_{k^*}. Moreover, increasing N_{k^*} would not introduce any detection complexity shown in Eq. 27.

As a result, the key of our approach is how to set N_{k^*}, making designers have a high probability to choose K^* that contains all key bits of one of LMs. If $N_{k^*} \gg n_{k^*}$, our approach could detect TSC quickly as shown by Eq. 28. Therefore, if $n_k = 128$, we could set $N_{k^*} = 80$, considering the fact that the number of key bits driven by LM is usually limited in the consideration of hardware cost of implementing LM. If $N_k^* = n_k$, in this extreme case, the algorithm would detect TSC extremely fast. However, for this ideal case, the assumption that P_{AES} and P_{AWGN} can be removed by sufficient power traces cannot hold. This is because that we are not be unable to generate sufficient power traces with the same random number by varying the value of key bits. If the random number does not depend on the plaintexts, designers are able to obtain power traces by varying plaintexts. This is the reason why the plaintext bit is recommended to be used by TSC.

4.4 Discussion

The proposed solution for TSC identification has the following two advantages. Compared to existing HT detection techniques using side-channel analysis, our approach does not require TSC-free chips as golden reference and hence are practical for TSC identification. Compared to the method in [7], our approach works for power-based TSCs with any GC designs that use a combination of arbitrary key bits, plaintext bits and random bits, while their method cannot handle such sophisticated TSCs.

Our approach works well against the always-on TSCs but may fail for trigger-based TSCs. This is because, our approach requires sufficient number of power traces with leaked information from Trojan-induced side channels, which may not be available if the trigger condition is not satisfied. At the same time, however, designers can use existing HT detection solutions such as [15] to identify HT triggers and they are compatible with our TSC identification technique.

5. EXPERIMENTAL RESULTS

5.1 Experimental Setup

We validate the proposed TSC identification technique by conducting experiments on FPGA. We adopt two TSCs in the experiments. One is MOLES whose architecture is shown in Fig. 1, denoted by TSC_1. Each LM of TSC_1 is implemented by XORing one key bit and one random bit, given by $X_i = K(i) \oplus R(i)$. The other one, denoted by TSC_2, is designed as follows. Each LM of TSC_2 is implemented by XORing two key bits and two random bits, given by $X_i = K(i) \oplus K(i+1) \oplus R(i) \oplus R(i+1)$. For both TSC_1 and TSC_2, the random number is generated by a linear feedback shift register, while the LC is realized by flip-flops and the load of LC is controlled by the number of flip-flops. We transplant TSC_1 and TSC_2 into an AES cryptosystem implemented on the FPGA. We then measure the transient power of the entire AES cryptosystem with the oscilloscope.

Table 3: Experimental parameter setting (EPS)

	EPS 1	EPS 2	EPS 3	EPS 4
TSC	TSC_1	TSC_2	TSC_1	TSC_2
n_{k^*}	1	2	1	2
n_r	1	2	1	2
N_{k^*}	1	2	2	1
Fig.	5	6	7	8

The experimental parameter settings are shown in Table 3. All power values shown in Fig. 5-8 remove the bias power, 1.188W, in order to clearly show the power difference between power curves. Thus, the '0' value on the y coordinate illustrate '1.188W'.

5.2 Results and Discussion

We validate the performance of our proposed TSC identification technique from three aspects: (i). TSC detection capability; (ii) the impact of the number of key bits (N_{k^*}) chosen for TSC identification; and (iii) the impact of LC load.

5.2.1 *Detection Capability*

Let us consider the detection of TSC_1 first. Fig. 5 plots the mean power curves with increasing number of power traces (NPT). For every LM in TSC_1, there are only two groups based on the value of $K(i)$. As can be observed in Fig. 5 (a), all the power curves gradually become stable with the increase of NPT. Then, we detail the mean power of all groups when NPT ranges from 1.7×10^4 to 1.9×10^4 in Fig. 5 (b). From this figure, we can observe that the difference between two groups of one LM is clear and stable. Such power gap is the evidence of the correlation between the key bit and power consumption, and hence our approach is able to detect all eight LMs in TSC_1.

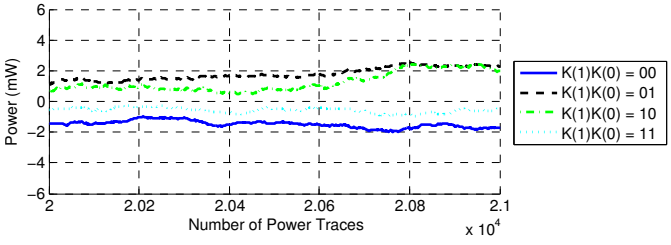

Figure 6: Power curves indicating the detection of TSC_2 with $N_{k^*} = 2$ and $K(0)K(1)$ chosen

Next, let us validate our TSC identification technique for TSC_2. Fig. 6 shows the mean power curves of four groups when $K(0)K(1)$ are chosen during Trojan detection and they drive one LM. A close examination of Fig. 6 reveals the following two observations. First, TSC_2 is successfully detected by our approach since there exists the pair of groups with clear power gap (e.g., $K(0)K(1) = 00$ and $K(0)K(1) = 01$), indicating the correlation between $K(0)K(1)$ and the power consumption. Second, we obtain the similar mean power when the pair of groups associated with 00 and 11 of $K(0)K(1)$ or the pair of groups associated with 01 and 10 are chosen. This is because that the value of X_i is the same for these pairs of groups, which leads to the same leakage power caused by LM.

While we only validate the proposed solution for TSC_1 and TSC_2, we believe it is insensitive to any power-based TSC designs according to the theoretical analysis in Section 4.

5.2.2 *Impact of* N_{k^*}

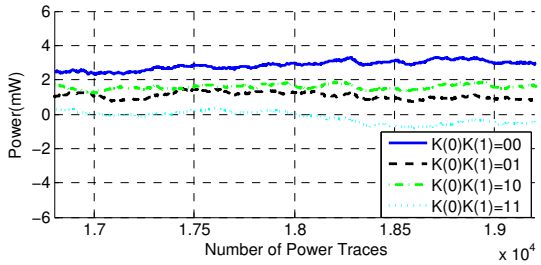

Figure 7: Power curves indicating the detection of TSC_2 with $N_{k^*} = 1$ and $K(0)K(1)$ chosen

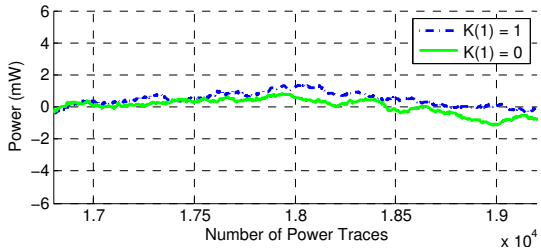

Figure 8: Power curves indicating the detection of TSC_2 with $N_{k^*} = 1$ and $K(1)$ chosen

In the above experiments, we have shown that whenever $N_{k^*} = n_{k^*}$, our approach is able to detect TSC when all key bits driven one LM are chosen during TSC detection. In this experiment, we study the cases when $N_{k^*} > n_{k^*}$ and $N_{k^*} < n_{k^*}$.

For $N_{k^*} > n_{k^*}$, we set $N_{k^*} = 2$ which is larger than $n_{k^*} = 1$ during the detection of TSC_1. Fig. 7 shows four power curves when $K(0)$ and $K(1)$ are chosen and either of them drives one LM. We observe that the mean power associated with 00 of $K(0)K(1)$ is the largest; the mean powers associated with 01 and 10 are close and are a little bit smaller; the mean power associated with 11 is the smallest. This is because that both LMs leaking power when the driven key bit is 0. Our approach is able to detect TSC_1 whenever any of the two groups with clear power gap are chosen.

For $N_{k^*} < n_{k^*}$, we set $N_{k^*} = 1$ which is smaller than $n_{k^*} = 2$ during the detection of TSC_2. Fig. 8 present the case when $K(1)$ is chosen for TSC identification. As can be observed, we cannot obtain a reliable power gap between the two mean power curves associated with $K(1) = 0$ and $K(1) = 1$. This is because the power of LMs is randomly distributed into the chosen groups and cannot be differentiated.

5.2.3 *Impact of LC load*

Finally, we study the impact of the LC size on the number of power traces required for TSC identification, and the results are shown in Table 4. It is clear that fewer power traces are needed with higher capacitive load of the leakage source (implemented as flip-flops in this work). This result shows that, while a large leakage source could help attackers to extract keys easily, it also makes TSC easy to be detected by our proposed TSC identification technique.

Table 4: Impact of load of LC

	# of flip-flops		
	16	128	1024
NPT	15231	8723	1864

6. CONCLUSION

In this paper, we first investigate the general architecture of power-based TSCs and introduce the corresponding key cracking process. Next, we present a novel TSC identification technique by exploiting the correlation between key and power side-channel. Experimental results for two TSC designs inserted into an AES cryptosystem implemented on FPGA proved the effectiveness of our proposed solution.

7. ACKNOWLEDGMENT

This work was supported in part by the Hong Kong SAR Research Grants Council (RGC) under General Research Fund No. CUHK418111 and No. CUHK418112, in part by National Natural Science Foundation of China (NSFC) No. 61073169, and in part by NSFC/RGC Joint Research Scheme No. N_CUHK444/12.

8. REFERENCES

[1] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[2] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology-CRYPTO*, pages 388–397, 1999.

[3] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, 2011.

[4] M. Tehranipoor and F. Koushanfar . A Survey of hardware Trojan taxonomy and detection. *IEEE Design & Test of Computers*, pages 10–25, January/February 2010.

[5] J. Zhang, F. Yuan, and Q. Xu. DeTrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware Trojans. In *Proc. ACM Conference on Computer and Communications Security (CCS)*, to appear, 2014.

[6] *Trust-Hub Website*. http://www.trust-hub.org/resources/benchmarks.

[7] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson. Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In *Proc. IEEE Cryptographic Hardware and Embedded Systems (CHES)*, pages 382–395. 2009.

[8] Y. Liu, Y. Jin, and Y. Makris. Hardware trojans in wireless cryptographic ics: silicon demonstration & detection method evaluation. In *Proc IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 399–404, 2013.

[9] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan detection using ic fingerprinting. In *Proc. IEEE Symposium on Security and Privacy (SP)*, pages 296–310, 2007.

[10] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey. Hardware trojan horse detection using gate-level characterization. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 688–693, 2009.

[11] S. Skorobogatov and C. Woods. Breakthrough silicon scanning discovers backdoor in military chip. In *Proc. International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 23–40, 2012.

[12] L. Lin, W. Burleson, and C. Paar. MOLES: malicious off-chip leakage enabled by side-channels. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 117–122, 2009.

[13] J. F. Gallais, J. Großschädl, N. Hanley, M. Kasper, M. Medwed, F. Regazzoni, J. M. Schmidt, S. Tillich, and M. Wójcik. Hardware trojans for inducing or amplifying side-channel leakage of cryptographic software. In *Trusted Systems*, pages 253–270. 2011.

[14] M. Hicks, M. Finnicum, S. T. King, M. Martin, and J. M. Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *Proc. IEEE Symposium on Security and Privacy (SP)*, pages 159–172, 2010.

[15] J. Zhang, F. Yuan, L. Wei, Z. Sun, and Q. Xu. Veritrust: verification for hardware trust. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, Article No. 61, 2013.