Aging Attacks for Key Extraction on Permutation-Based Obfuscation

Zimu Guo, Mark M. Tehranipoor and Domenic Forte
ECE Department, University of Florida
Email: zimuguo@ufl.edu; {tehranipoor,dforte}@ece.ufl.edu

***Abstract*—Permutation-based obfuscation has been exploited to protect hardware against cloning, overproduction, and reverse engineering with a secret key. In order to prevent key extraction from memory, this key is usually stored in volatile memory. Since the key is erased after the system loses power, this scheme is often considered the best way to prevent a key from being stolen since many attacks would require power. However, in this paper, we propose a new attack where the key is determined by exploring path aging within the permutation network used for obfuscation. Both the theoretical analysis and experimental results are provided. A practical procedure to achieve the proposed attack is also discussed in the context of an attacker's capabilities and knowledge. We also present an adjustment scheme to improve the accuracy of the attack. Various aging durations, process variations and measurement conditions are considered in our simulations. The experimental results show the accuracy of identifying the key is as high as 92.4% and more than enough to reduce the number of brute force combinations required by an attacker.**

I. Introduction

Electronic devices, such as integrated circuits (ICs) and printed circuit boards (PCBs), encounter several security/privacy threats, such as cloning, overproduction, reverse engineering, unauthorized operation, etc. [1]. Cloning indicates that the attacker applies either destructive or non-destructive reverse engineering [2] on the IC/PCB to steal the confidential design. With these cloned designs, the attackers can produce their copies, which may be low in quality or even contain hardware Trojans [3]. Overproduction refers to an attack where an untrusted manufacturer produces more devices than its contract with the IP owner allows. Unauthorized operation indicates that devices are used/accessed by the person without permissions. These threats may cause the owner to lose profits and/or reputation. Even more seriously, some critical military installations, such drones/UAVs could be captured and hacked by enemies [4].

Against these threats, hardware obfuscation techniques provide practical solutions. In general, these methods can be classified into three categories based on the abstraction level where they are implemented: gate level, register-transfer (RT) level, and board level. To be specific, gate-level approaches include logic encryption which encrypts gate-level internal connections with XOR gates [5] and logic permutation [6] which permutes these internal connections. RT-level methods modify the high-level design and insert redundant states into the original finite state machine (FSM) [7]. Compared with the logic permutation in gate level, the board-level approach permutes the inter-chip connections [8]. A common feature of these techniques is that they require a secret key to unlock the obfuscated chip or PCB. In order to discover this key, several attacks have been proposed against the encryption [9] and FSM-based obfuscation. However, to the best of our knowledge, no attacks have been found against the permutation-based obfuscation thus far. In this paper, we propose a framework for extracting the key/configuration of the permutation network, which is utilized in permutation based obfuscation.

Our attack should apply to many of the permutation-based approaches already in the literature. Figure 1 [8] illustrates the board-level protection scheme against PCB reverse engineering. In this scheme, part of the actual board-level connections are permuted by a permutation network (a standalone chip). A key which is applied by a trusted party can drive the system to operate correctly by guiding the inputs to the correct outputs. During its normal operation, the inputs of the permutation block must always be routed to the right outputs. It is paramount that these keys or configuring information are kept secured. Cryptography is a widely used approach to secure the confidential information. However, constrained by the resources, some designs are not capable to encrypt/decrypt the key. For these designs, the key are either loaded from the in-system non-volatile memory (NVM) or input externally during each rebooting. The key is vulnerable if it is permutably stored in the system, since the NVM can be easily compromised [10]. In order to achieve a better protection of the key, it should be input externally during each power-up. Then, this key is stored in the volatile memory (e.g. registers) and will be erased after the system loses power.

Consider a powered-off permutation chip which has been working for some time. This permutation chip is designed to load a unique key externally against non-volatile memory attacks [10]. Nonetheless, this chip lacks decryption capability considering the resource constraints. Even if this key is erased after powering off the chip, some "hints" may remain due to the device aging. The term "hints" refers to the information which can be exploited to discover the content of the key. In this paper, we propose an attacking framework utilizing the "hints" from device aging to learn the secret key. Since the degree of transistor aging depends on the inputs, different input patterns result in different path delays. The attacking framework selects the permutation network's paths by manipulating the key. For each selected path, a delay profile is extracted. The secret key is derived by analyzing these delays. Our major contributions in this paper are summarized as follows:

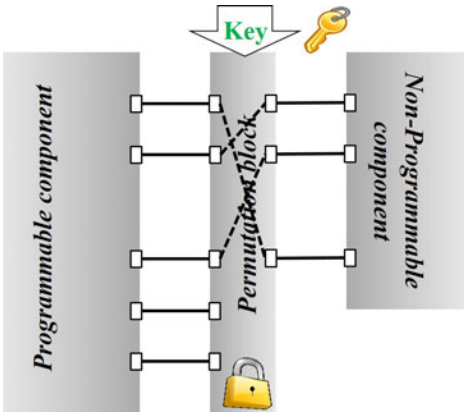


Fig. 1. Schematic of hardware implementation of a board-level permutation based protection scheme [8].

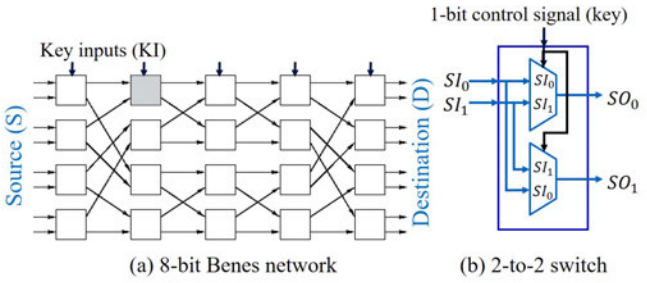


Fig. 2. 8-bit Benes network implementation and the switch-level structure. The network is composed with 2-to-2 switches and each of them consists of two identical multiplexers.

- We propose a comprehensive attack framework for discovering the key/configuration of a used permutation chip. We assume that the key/configuration is not encrypted as it is sent to the chip.
- We provide an aging analysis of multiplexers and present the transistor-level aging models. In these models, the relationship between path aging differences in the permutation network (consisting of multiplexers) and different key/configuration data is studied.
- We analyze the attack effectiveness based on various aging and attacking conditions, such as the aging duration, temperature and measuring resolution, etc. We also verify the proposed framework by implementing Monte Carlo simulations incorporating manufacturer process variations and measurement noise.

The rest of the paper is organized as follows. In Section II, the attacking target and attacker's capabilities are provided. In Section III, we elaborate on the proposed attacking framework. The experimental results and analyses are provided in Section IV. Finally, we conclude this work and provide future directions in Section V.

II. Attack Model

A. Attacking target

As an example of the permutation network, an 8-bit Benes network is shown in Figure 2 (a) [11]. It can achieve any 8-to-8 permutation between the source (S) and destination (D). This permutation is controlled by a 20-bit binary key. Each bit of the key is used as a select input of a 2-to-2 switch, which is the basic unit of the Benes network (Figure 2 (b)). Each of these switches consists of two multiplexers and can operate in two modes: straight mode ($key = 0$) and exchange mode ($key = 1$). In the straight mode, SI_0 is routed to SO_0 and SI_1 is routed to SO_1, while in the exchange mode, the outputs are swapped.

A transistor-level implementation of these multiplexers is provided in Figure 3. This NAND-gate based implementation consists of 3 NAND gates and 1 inverter. The transistors, which form these gates, are marked from M1 to M14. Besides the NAND gates and inverters, other logic gates or transmission/pass gates can also be utilized to build the multiplexers. In this paper, the NAND-based structure is exploited as the attacking target for simplicity. Application of the proposed attack to other multiplexer structure is proposed at the end of this section. In Section III, the theoretical aging analyses on these transistors are provided.

B. Attacker's capabilities

The **attacker's goal** is to learn the key/configuration of a powered-off permutation chip [2]. It is assumed that the key/configuration can be directly controlled or overwritten by the attacker, and encryption/decryption is not used to protect the key. This is often a valid assumption since such crypto hardware might have high overhead.

[2]It is assumed that the attacker only gets hold of a locked system. An unlocked system can be easily probed for the key without the need for the proposed attack.

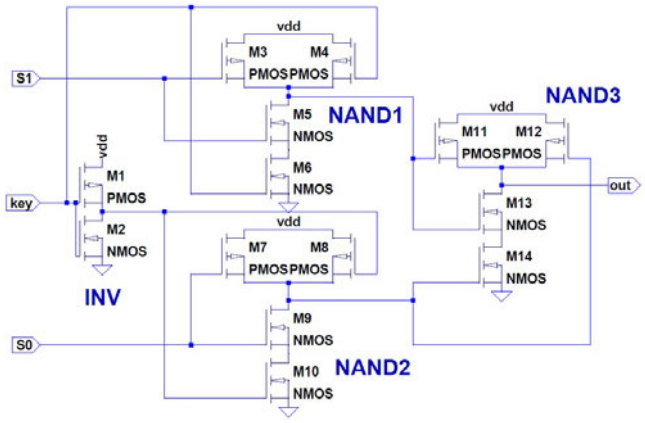


Fig. 3. NAND-based multiplexer transistor-level structure. This type of multiplexer consists of only NAND gates and inverters.

Referring to Figure 2 (a), the **attacker's capabilities/knowledge** are as follows:

(i) The controllability of the source (S) and the key input (KI). The attacker should also have the capability to load a signal into the permutation network via the source and manipulate the key values.

(ii) The knowledge of the key bits' assignment information. This information implies the one-to-one correspondences between the key bits and switches. If this information is not public, the attacker can learn it by simple experiments (see below).

(iii) The observability of permutation network's output (D). The attacker should be able to measure the delay between source and destination.

The **controllability (i)** and **observability (iii)** can be achieved by the attacker since the inputs/outputs of the permutation network are directly connected to the pins of the permutation chip. The key is loaded externally by a user through a dedicated port on the permutation block during each reboot [8]. The attack can inject/read signals to/from the permutation network through these same pins. In order to learn the key bits' **assignment information (ii)** when it is not publicly available, the attacker should proceed an experiment as follows. First, the key bits are initialized as all "0"s. For tracking the input-output relationship, each permutation network's input loads a signal with a unique frequency. Then, the key bits are flipped one after another. After each flipping, one pair of inputs switches their output ports. These switching events are collected after all the key bits are flipped. According to these switching activities, the key bits' assignment information can be obtained.

III. Attack Execution

The attack execution analysis consists of four major parts: (i) NAND-gate based multiplexer aging model, (ii) single switch delay analyses, (iii) common path cancellation and (iv) pre-charging adjustment. We will elaborate on these parts in this section.

A. Multiplexer aging model

The performance of an operational IC slowly but gradually decreases over time due to several phenomena including bias temperature instability (BTI), hot carrier injection (HCI). Although BTI and HCI both degrade pMOS and nMOS, the amount is not the same. NBTI is widely considered as one of the most critical issues for the reliability of pMOS transistors.

While, for nMOS transistors, HCI plays a more significant role in the threshold voltage degeneration [12].

Based on these approximations, the transistor-level aging model of an NAND-gate based multiplexer is presented in Table I. In this table, the transistors labeled M# correspond to the transistors shown in Figure 3. Part of these transistors are marked by HCI or NBTI as the principal sources of the degradations. The rest of them which are characterized as "–" implying that they are not affected by HCI or NBTI. Thus, these transistors possess relatively lesser degradations. This aging model will be utilized to determine the delays in Section III-B.

B. Single switch analysis

In this analysis, a single switch is considered as the attacking target. The sensitized paths within this switch are defined, and their delays are measured. The *switch delays* are computed according to these *path delays*. Based on the switch delays, the attacking framework can reveal the correct key values which were applied during the normal operation.

As shown in Figure 2 (b), each switch consists of two identical multiplexers with inverse inputs. These multiplexers follow the transistor-level structure provided in Figure 3 and the aging model in Table I. The inputs and outputs of any switch are assumed to be accessible during the analyses. More details about this assumption are provided later. As a result, the propagation delays between inputs and outputs for a single switch are measurable. These switch delays are collected in two operating modes: the straight mode and exchange mode.

Apparently, different input-to-output paths will be sensitized by different key values. The term "sensitized" indicates that the signal can pass a path which is defined by an input-output pair. For instance, in Figure 3, setting the key to "0" blocks $NAND1$ and forces $NAND1$ to pass a "1" (logic high) to $NAND3$. Thus, a pulse loaded from input $S0$ sensitizes the following path: $S0$, $NAND2$, $NAND3$ and out. This path is presented in Figure 4 (a). The other path shown in Figure 4 (b) can be sensitized by setting the key to "1" and loading a pulse from $S1$.

When a particular path is sensitized, a rising edge is sent through the switch's input and a rising edge can be received at the output. The time difference between the sent and received rising edges is determined as the path delay. Since these paths pass the same number of NAND gates, their delays are almost the same when the switch is new, and no transistors are aged. The term "almost" implies that slight delay differences may be introduced by the manufacturing process variation. Compared with the new status, an aged switch presents the increased path delay due to the transistors' degradation. This increment depends on how much the transistors are aged along each path.

To investigate the aforementioned path delay, we split it into two parts: new-status propagation delay (d_{base}) and additional delays caused by aging. In any switch operation mode (straight or exchange), all the delay paths contain the same number of NAND gates. This property implies that

TABLE I. NAND-GATE MUX TRANSISTOR LEVEL AGING MODEL

Aging key = 0							
PMOS	M1	M3	M4	M7	M8	M11	M12
	NBTI	NBTI	NBTI	NBTI	–	–	NBTI
NMOS	M2	M5	M6	M9	M10	M13	M14
	–	HCI	–	HCI	–	–	HCI

Aging key = 1							
PMOS	M1	M3	M4	M7	M8	M11	M12
	–	NBTI	–	NBTI	NBTI	NBTI	–
NMOS	M2	M5	M6	M9	M10	M13	M14
	–	HCI	–	HCI	–	–	HCI

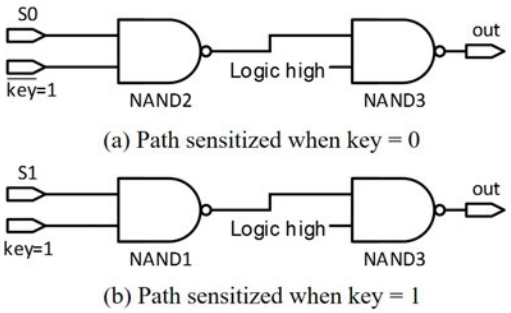


Fig. 4. Single switch delay paths. Different pathes (e.g. $S0$ to out and $S1$ to out) are sensitized by different key values.

TABLE II. AGING INTRODUCED PATH DELAY ANALYSIS.

A | T	0 | 0		0 | 1		1 | 0		1 | 1	
MUX	L	U	L	U	L	U	L	U
NBTI	M12	M12	–	–	–	–	M11	M11
HCI	M9	M9	M5	M5	M9	M9	M5	M5
MUX delay	$d_s^L\|0$	$d_s^U\|0$	$d_e^L\|0$	$d_e^U\|0$	$d_s^L\|1$	$d_s^U\|1$	$d_e^L\|1$	$d_e^U\|1$
Switch delay	$d_s\|0$		$d_e\|0$		$d_s\|1$		$d_e\|1$	

d_{base} is the same among all the paths if process variations are negligible. The amount of aging-induced delay depends on the aging models provided in Table I. Incorporating these two aging models and the path sensitized, four *switch delays* can be estimated and collected as shown in Table II.

In this table, the row "A | T" denotes the aging and testing key values. The aging key indicates the key applied when the device operates normally (before being attacked). The testing key is the one which the attacker applies during the attack. In the row "MUX", the label "L" denotes the lower multiplexer, while "U" denotes the upper one in the same switch (Figure 2 (b)). The transistor's label ($M\#$) indicates that the corresponding transistor contributes to the additional path delay through NBTI or HCI. The marker "–" implies that no significant aging effect occurs. The row "MUX delay" provides the notations of lower and upper *multiplexer delay*, which is the path delay within one multiplexer. For instance, the propagation delay of the lower and upper multiplexer when the aging key is "0" and the testing key is "0" can be expressed as,

$$d_s^L|0 = d_{base} + d_{NBTI} + d_{HCI} \text{ Lower MUX path delay} \quad (1)$$

$$d_s^U|0 = d_{base} + d_{NBTI} + d_{HCI} \text{ Upper MUX path delay} \quad (2)$$

where d_{NBTI} and d_{HCI} denote the additional delays caused by NBTI and HCI respectively. Note that the subscript "s" denotes straight mode and the subscript "e" will denote exchange mode for all delay variables in this paper. The straight switch delay can be computed by summing the lower and upper multiplexer delays (Equation (1) and (2)), such as,

$$d_s|0 = d_s^L|0 + d_s^U|0 = 2 * d_{base} + 2 * d_{NBTI} + 2 * d_{HCI} \quad (3)$$

The same calculation can be applied to the rest of switch delays. The expressions of these switch delays are provided in the following equations.

$$d_s|1 = d_s^L|1 + d_s^U|1 = 2 * d_{base} + 2 * d_{HCI} \quad (4)$$

$$d_e|0 = d_e^L|0 + d_e^U|0 = 2 * d_{base} + 2 * d_{HCI} \quad (5)$$

$$d_e|1 = d_e^L|1 + d_e^U|1 = 2 * d_{base} + 2 * d_{NBTI} + 2 * d_{HCI} \quad (6)$$

By comparing these switch delays computed by Equation (3) to (6), we can see that $d_s|0$ is always greater than $d_e|0$

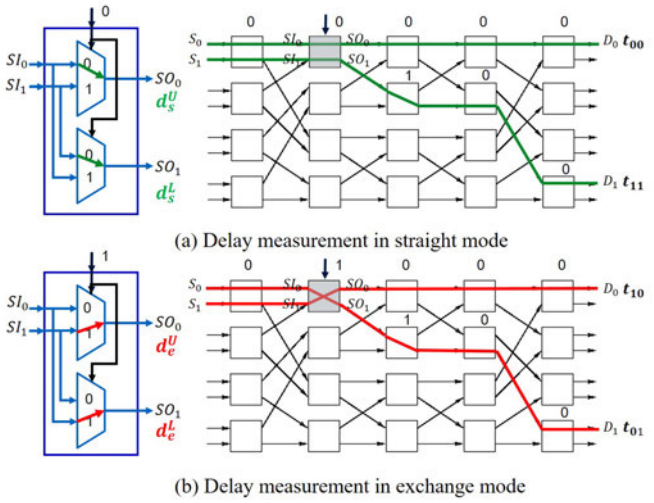


Fig. 5. Common path cancellation illustration. The propagation delays of do-not-care pathes are neutralized.

and $d_e|1$ is always greater than $d_s|1$. Thus,

$$d_s > d_e \Rightarrow \text{Normal operating key value is “0”} \quad (7)$$
$$d_s < d_e \Rightarrow \text{Normal operating key value is “1”} \quad (8)$$

$$d_s = d_s^L + d_s^U \quad \text{and} \quad d_e = d_e^L + d_e^U \quad (9)$$

However, it is impossible for the attackers to access all the inputs/outputs of each single switch directly. The reason is that at least half of the switch's ports are connected internally and cannot be accessed from the network's I/Os. In the next section, a practical solution is proposed to solve this issue.

C. Common path cancellation

In this part, an approach is introduced to solve the aforementioned issue. According to Equations (7) and (8), the proposed approach should be able to achieve the comparison between d_s and d_e by measurable delays. This measurable delay means the *network delay*, which can be directly measured through the permutation network's inputs and outputs.

Considering the attacker's capabilities and knowledge, he can determine any network delay between an arbitrary pair of input and output. In order to achieve this goal, the key is designed to choose a specified network path. Several examples are provided in Figure 5 and the required key configurations are provided (the rest of the key configurations are do-not-care). The chosen paths are marked as green and red. All these paths pass the same shaded switch. The network delay can be determined by applying a rising edge at the inputs (S_0 or S_1), then measuring the propagation delay at the corresponding output (D_0 or D_1). In this case, four network delays can be determined: t_{00} ($S_0 \rightarrow D_0$), t_{11} ($S_1 \rightarrow D_1$), t_{10} ($S_1 \rightarrow D_0$) and t_{01} ($S_0 \rightarrow D_1$).

Assume the attacking target is the shaded switch with inputs SI_0/SI_1 and outputs SO_0/SO_1, shown in Figure 5. The target switch's straight-mode multiplexer delays are provided as the left figure in Figure 5 (a). The left figure in Figure 5 (b) shows the exchange-mode multiplexer delays. These four multiplexer delays are involved in four chosen network delays respectively, such as, $d_s^U \in t_{00}$, $d_s^L \in t_{11}$, $d_e^U \in t_{10}$ and $d_e^L \in t_{01}$. Besides these switch delays, the network delays also consist of other delay fractions shown in Table III with their starting and ending points. These fractions are labeled from d_1 to d_4.

Based on the delay fractions and multiplexer delays, the four network delays in Figure 5 can be expressed as Equations (10), (11), (12) and (13).

TABLE III. FRACTIONS OF THE NETWORK DELAY

Delay fraction label	Starting	Ending
d_1	S_0	SI_0
d_2	S_1	SI_1
d_3	SO_0	D_0
d_4	SO_1	D_1

$$t_{00} = d_1 + d_s^U + d_3 \quad (10) \qquad t_{10} = d_1 + d_e^L + d_4 \quad (12)$$
$$t_{11} = d_2 + d_s^L + d_4 \quad (11) \qquad t_{01} = d_2 + d_e^U + d_3 \quad (13)$$

Then, Equation (10) is subtracted by (12) and Equation (11) is subtracted by (13) for cancelling the common delay fractions d_1 and d_2. By doing this subtraction, we get,

$$d_s^U + d_3 - d_e^L - d_4 = t_{00} - t_{01} \quad (14)$$
$$d_s^L + d_4 - d_e^U - d_3 = t_{11} - t_{10} \quad (15)$$

Common delay fractions d_3 and d_4 can be further cancelled by adding Equation (14) and (15). Combining Equation (9), the following expression can be denoted.

$$\underbrace{d_s^U + d_s^L}_{d_s} - \underbrace{d_e^U + d_e^L}_{d_e} = t_{00} + t_{11} - t_{10} - t_{01}$$
$$\Rightarrow \quad d_s - d_e = t_{00} + t_{11} - t_{10} - t_{01} \quad (16)$$

Finally, the attacking framework can conclude the key bit by comparing between d_s and d_e through checking the sign of $t_{00} + t_{11} - t_{10} - t_{01}$. The decision rules are presented in Equation (17).

$$\begin{aligned} &\text{Check the sign of } (t_{00} + t_{11} - t_{10} - t_{01}): \\ &\text{Positive} \Rightarrow \text{Normal operating key value is “0”} \\ &\text{Negative} \Rightarrow \text{Normal operating key value is “1”} \end{aligned} \quad (17)$$

The presented approach achieves the required comparisons in Equation (7) and (8) by cancelling the common fractions of the delay paths. Hence, we name this approach as *common path cancellation*. For each switch, the above procedure is executed, and four delay values should be measured. Since the number of switches is directly related to the dimension of the network, complexity of the proposed attack follows:

$$\mathcal{O}(\text{n-bit network}) = n \times (4 \times \log_2(n) - 1) \quad (18)$$

Where, n is the number of inputs of the permutation network.

D. Pre-charging adjustment

Besides the process variation discussed earlier, another factor may also affect the switch delays and introduce errors when the attacker makes decisions. This factor is called the pre-charging effect, which is commonly studied as an optimization problem in IC designs. Unlike the process variation which cannot be completely eliminated, the negative effect of this factor can be diminished by applying a proper adjustment. In this section, comprehensive analyses on the effect, and the adjustment is provided.

In Figure 6, both circuits achieve a logic NAND between an input ($Pulse$) and a logic high (vdd). The only difference between these two circuits is the locations of “always-on transistors”. The nMOS M4 in Figure 6 (a) and nMOS M3 in Figure 6 (b) are always kept on.

We analyze the delays of these two circuits when a rising edge presents at the input $Pulse$. Prior to the arrival of this rising edge, the circuit outputs a high voltage at out. In Figure 6 (a), since M4 is always on, the source of M3 is connected to the ground. After the gate voltage of M3 switches to high, M3 turns on. The discharging path is from M3's drain to M3's source. On the other hand, in Figure 6 (b), the discharging path is from M3's drain to M4's source, which is longer. A longer discharging path leads to larger delay. These two

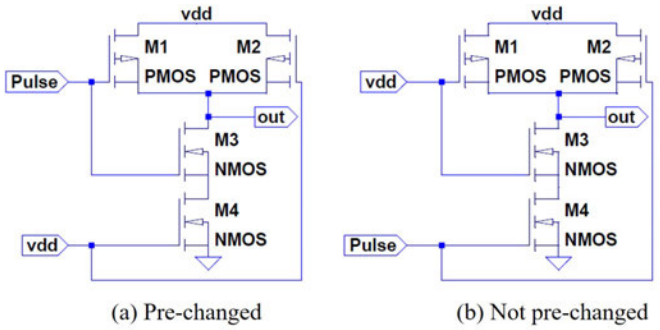


Fig. 6. Pre-charging adjustment. The circuit presented in (a) is pre-charged and performs less delay than the non-pre-charged circuit in (b).

circuits are simulated in SPICE based on the setup shown in Section IV-A. The propagation delay from $Pulse$ to out is measured as d_a for the circuit in Figure 6 (a) and d_b for the circuit in Figure 6 (b). For the proposed attack, the each switch's key controls whether the sensitized path contains any gates which are not pre-charged.

According to the proposed attack, the transistor aging should be the only source which changes the path delay. In other words, the new-status propagation delay (d_{base}) should be the same if the path consists of the same number of gates. The delay should be adjusted if any path contains one or more gates which are not pre-charged. This adjustment can be accomplished by subtracting measured delay with a pre-computed adjustment value. This value is computed as

$$d_{adj} = m * (d_b - d_a) \tag{19}$$

where, m is the number of gates which are not pre-charged. To compute d_{adj}, the attacker only needs to know the technology node of the permutation chip. This adjustment value can be either obtained simulation (e.g. HSPICE) or theoretical calculation.

By analyzing the switch's operating modes (straight and exchange), we conclude that the adjustment should be applied when the testing key equals to "0". The number of non-pre-charged gates is 1. The effect of proposed delay adjustment will be presented in Section IV based on simulation data.

Applications to other permutation networks: The proposed attacking framework can be applied to any other 2-to-2 switch based permutation networks such as Omega network, butterfly network, etc. These switches consist of NAND-based (3) and AND-based multiplexers. The AND-based multiplexer presents similar characteristics as the NAND-based one does in Section III-B. Moreover, the common path cancellation approach (Section III-C) is applicable to these networks. After applying this cancellation approach, attacking any of these networks simplifies to an attack on attack a single switch, although the internal structures of these networks are different.

IV. Experimental Results

A. Experimental Setup

For the transistor library, the Predictive Technology Model (PTM) 45nm transistor library is exploited. The highest level of accuracy of this library is engaged in the simulation. For the standard cell library, we utilize the library from the NanGate. The two inputs of the switch are 50kHz with duty cycles 75% and 25% respectively. The nominal supply voltage for the transistor library is 1.1 volts. The permutation network operates at room temperature ($25°C$) during normal usage. The ambient temperatures during the attack is also set as $25°C$. The durations of this normal operation vary from 1 week to 2 years. The percentage of time which the circuit is on is set as $10\%, 20\%, 40\%$ and 80%. The time domain resolutions of the devices (e.g. an oscilloscope) which are utilized to measure the delay can be $0.1\mu s$, $10ps$ and $1ps$. The signal-to-noise ratio is considered as 10dB in the simulation.

As discussed in Section III-B, the process variations could introduce errors when the attacker makes decisions. In our experiment, the effect of process variations is evaluated by incorporating Monte Carlo simulations. For each aging duration (e.g. 1 week or 1 year), 10,000 Monte Carlo simulations are performed. Each of these simulation refers to the attacking result for each switch (i.e. the key is correctly recovered or not). The attacking accuracy is computed by Equation (20).

$$\text{Attacking accuracy} = \frac{\text{Number of correct recoveries}}{10,000} \tag{20}$$

According to the common path cancellation in Section III-C, the percentage of the permutation network's key bits, which can be correctly recovered, equals to the attacking accuracy computed in Equation (20).

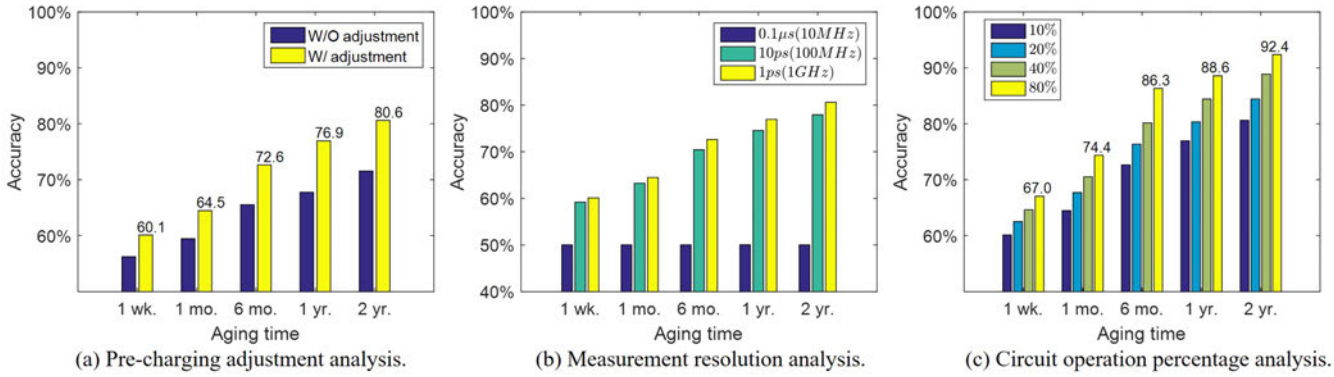
The simulation and aging model are configured based on [13]. The systematic process variations of the transistor's channel length and threshold voltage are expressed by zero-mean Gaussian distributions. The variances are set according to [14]. These variances are doubled for combining local and systematic process variations.

B. Attack Accuracy

In this section, the simulation results of the **attack accuracy** are presented. This criterion implies the percentage of the key bits which can be discovered by proposed the attack framework. The desired attacking accuracy is 100% (i.e., no error in predicting the correct key bits). With a higher percentage of key bits recovered by the attacker, the total strength of the key decreases exponentially. The average attacking accuracy derived from 10,000 Monte Carlo trials (10,000 switches) is provided.

*1) **Pre-charging adjustment**:* We apply the pre-charging adjustment on attacking a permutation network and compare the results in this section. The experimental adjustment value is obtained using HSPICE simulation at room temperature (25°C). According to Equation (19), this value equals to $1.2e-10s$. Besides the temperature, the circuit is assumed to be turned on for 10% of its aging period. The resolution of the delay measurement device is set to be $10ps$. More analyses on the different settings of measurement resolution, etc. are provided in subsequent sections.

In order to examine how this value influences the attacking accuracy, we sweep the adjustment value from 0 to $3e-11s$ and present this analysis in Figure 8. Setting this adjustment value to "0" implies that no adjustment is involved. In this figure, the attacking accuracies when the aging key equals to "0" or "1" are provided in red and blue curves respectively. With the adjustment value increases, the exchange-mode accuracy (blue curve) increase which the straight-mode accuracy (red curve) decreases. Since the key bits are mixed with "0"s and "1"s, the accuracy of the proposed attacking framework should consider both cases. Thus, the average accuracy is computed by averaging the accuracies when the aging key is "0" and "1". The optimal point which marked as black circle points to the highest average accuracy. The experimental adjustment value ($1.2e-10s$) is shown as x-mark in the same figure. It can be seen from the figure that this experimental adjustment value produces similar accuracy as the optimal point. Figure 7 (a) presents the comparison of this improvement among different aging durations. The attacking accuracy improvement can been observed as 5% to 10% when considering different length of aging. The longer the aging is, the more improvement of the attacking accuracy can be obtained.



(a) Pre-charging adjustment analysis. (b) Measurement resolution analysis. (c) Circuit operation percentage analysis.

Fig. 7. Summary of the simulation results.

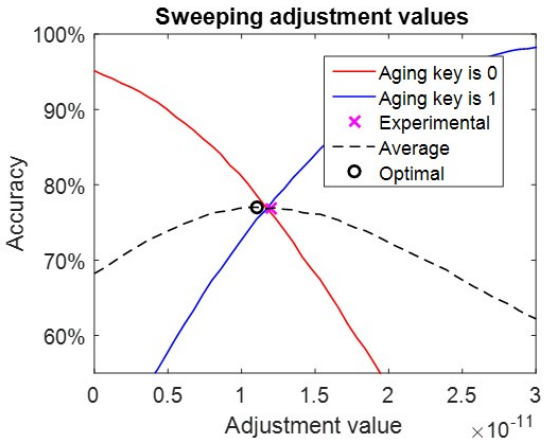


Fig. 8. Pre-charging adjustment analysis.

*2) **Measurement resolution**:* Measurement resolution is another factor which could influence the attacking accuracy. This resolution indicates the smallest time unit of the device used to measure the path delay, such as an oscilloscope. Three levels of resolutions are examined: $0.1\mu s$, $10ps$ and $1ps$. The corresponding oscilloscope bandwidths are $10MHz$, $100MHz$ and $1GHz$. These oscilloscopes can be easily found from the market. According to Figure 7 (b), an oscilloscope with the bandwidth no larger than $10MHz$ results in a 50% attacking accuracy. As a result, $10MHz$ is the lower bound of the oscilloscope bandwidth. If the bandwidth exceeds $1GHz$, the attacking accuracy is barely improved. Thus, the attacker should choose an oscilloscope with bandwidth ranging from $10MHz$ to $1GHz$ (e.g. $100MHz$).

*3) **Circuit operating time**:* The percentage of time which the circuit is on varies among different applications. We studied the effects of the circuit's activation time percentage. Four different percentages are set as follows: $10\%, 20\%, 40\%$ and 80%. For the system (e.g. the industry control system), the permutation network may be kept operating for more than 80% the time. For the consumer electronics, such as the cell phones and personal computers, this percentage may close to 50%. The results are provided in Figure 7 (c). The larger this percentage is, the longer time the permutation network is on during its operation. Thus, the device experienced more degradations. According to Figure 7 (c), with increasing this percentage, the attacking accuracies are increased. The attacking accuracy can be as high as 92% after a 2-year aging with 80% of the time which the circuit is on.

V. Conclusion and Future Work

We proposed a framework for attacking the permutation network from an aged device. This proposed framework can be exploited to break permutation-based obfuscation. Theoretical analyses of the feasibility are provided first through inspecting the transistor-level aging model. Following this feasibility analyses, the common path cancellation approach is introduced as a guide to implement the proposed attack. As for future work, we would focus on applying this attack on other types of multiplexer structures and other types of permutation networks.

VI. Acknowledgement

This project was supported in part by US Army Research Office grant under award number grant W911NF-16-1-0321.

References

[1] M. M. Tehranipoor, U. Guin, and D. Forte, *Counterfeit Integrated Circuits: Detection and Avoidance*. Springer, 2015.

[2] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A survey on chip to system reverse engineering," *ACM JETC*, 2016.

[3] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design and Test of Computers*, 2010.

[4] K. Hartmann and C. Steup, "The vulnerability of uavs to cyber attacks-an approach to the risk assessment," in *CyCon*, 2013.

[5] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Trans. Comput*, 2015.

[6] S. Khaleghi, K. Da Zhao, and W. Rao, "Ic piracy prevention via design withholding and entanglement," in *ASP-DAC*, 2015.

[7] R. S. Chakraborty and S. Bhunia, "Harpoon: an obfuscation-based soc design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2009.

[8] Z. Guo, M. Tehranipoor, J. Di, and D. Forte, "Investigation of obfuscation-based anti-reverse engineering for printed circuit boards," in *DAC*, 2015.

[9] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm arbiter pufs: Accurate modeling poses strict bounds on usability," in *WIFS*, 2012.

[10] H. Jeong, Y. Choi, W. Jeon, F. Yang, Y. Lee, S. Kim, and D. Won, "Vulnerability analysis of secure usb flash drives," in *MTDT 2007*, 2007.

[11] C. Chang and R. Melhem, "Arbitrary size benes networks," *Parallel Processing Letters*, 1997.

[12] E. Maricau and G. Gielen, "Transistor aging-induced degradation of analog circuits: Impact analysis and design guidelines," in *ESSCIRC*, 2011.

[13] Synopsys, "Synopsys MOSRA manual," 2014.

[14] K. Ajayan and N. Bhat, "Device oriented statistical modeling method for process variability in 45nm analog cmos technology," in *IWPSD*, 2012.