PATHFINDER: Side Channel Protection through Automatic Leaky Paths Identification and Obfuscation

Haocheng Ma*, Qizhi Zhang*, Ya Gao*, Jiaji He*, Yiqiang Zhao*, and Yier Jin†
*School of Microelectronics, Tianjin University
†Department of Electrical and Computer Engineering, University of Florida
hc_ma@tju.edu.cn, qizhi_zhang@tju.edu.cn, gaoyaya@tju.edu.cn, dochejj@tju.edu.cn, yq_zhao@tju.edu.cn, yier.jin@ece.ufl.edu

***Abstract*—Side-channel analysis (SCA) attacks show an enormous threat to cryptographic integrated circuits (ICs). To address this threat, designers try to adopt various countermeasures during the IC development process. However, many existing solutions are costly in terms of area, power and/or performance, and may require full-custom circuit design for proper implementations. In this paper, we propose a tool, namely PATHFINDER, to automatically identify leaky paths and protect the design, and is compatible with the commercial design flow. The tool first finds out partial logic cells that leak the most information through dynamic correlation analysis. PATHFINDER then exploits static security checking to construct complete leaky paths based on these cells. After leaky paths are identified, PATHFINDER will leverage proper hardware countermeasures, including Boolean masking and random precharge, to eliminate information leakage from these paths. The effectiveness of PATHFINDER is validated both through simulation and physical measurements on FPGA implementations. Results demonstrate more than $1000\times$ improvements on side-channel resistance, with less than $6.53\ \%$ penalty to the power, area and performance.**

***Index Terms*—CAD for Security, Side Channel Analysis, Countermeasure**

I. INTRODUCTION

Cryptographic algorithms provide services of data encryption and identity authentication, playing an important role in modern information security scenarios. Nowadays, cryptographic algorithms are widely deployed in critical applications including autonomous vehicles, electronic banking, mobile communication, and cloud computing. Despite the fact that these algorithms are mathematically secure and are resistant to various attacks targeting the algorithms themselves, their hardware implementations are vulnerable to physical attacks, e.g. side-channel analysis (SCA) [1]. SCA attack is the process of analyzing side-channel leakage through which attackers can extract sensitive internal information. The sensitive internal information may not be available through traditional cryptography analyses. Examples of side channels include power consumption, timing delay, electromagnetic (EM) emanations, heat, optical leakage, etc. Among them, power and EM SCA attacks on cryptographic integrated circuits (ICs) have gained tremendous importance over the last decade.

Numerous countermeasures have been equipped on modern ICs to defend against SCA attacks. For most of the hardware countermeasure schemes in use, the underlying concepts can be classified into two categories, namely hiding and masking techniques. These methods act on those secret-dependent intermediate data, denoted as sensitive variables, which usually are sources of leakage during cipher execution. Hiding techniques represent a group of techniques aiming to breach the correlation between the side-channel information and these sensitive variables. They can be implemented by randomizing switching behaviors, attenuating signatures of sensitive blocks, adding components that generate random noise, etc. While masking techniques try to obfuscate the internal computation and thus isolate the side-channel information from the sensitive variables [2]. On the basis of secret sharing, sensitive variables are split into several shares and imported to circuit operations separately. However, in addition to significant power, area and speed overhead increases, these countermeasures require designers to have sufficient hardware security background to implement them correctly. To address these concerns, designers develop tools to first identify sources of leakage, then apply protections to specific vulnerable modules [3] or logic cells [4].

In this paper, we pay attention to identifying and obfuscating leaky paths at the gate level of the design. Information leakage caused by sensitive variables will propagate through leaky paths within the design. Unlike protecting each vulnerable cell, countermeasures on leaky paths take multiple vulnerable cells as a whole to be protected. Meanwhile, this type of protection can avoid excessive efforts on irrelevant logic cells in a vulnerable module. Hence we can realize sufficient security improvement with less area, power and performance overheads. A novel tool, namely PATHFINDER, is developed for the above purpose. In path identification, PATHFINDER combines dynamic correlation analysis and static security checking to find out all leaky paths quickly. Then in path obfuscation, PATHFINDER exploits hardware solutions like Boolean masking and random precharge to protect leaky paths automatically. The tool can be easily integrated into the existing design flow and not tied to a specific technology node.

The main contributions of the paper are as follows.

- We develop the tool, named PATHFINDER, to support automated side-channel protection. This tool can identify and obfuscate leaky paths automatically and is compatible with the existing EDA design flow.
- PATHFINDER combines dynamic correlation analysis and

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530413

static security checking to identify complete leaky paths. The combination of dynamic and static steps enables accurate and efficient results.
- In path obfuscation, PATHFINDER leverages logic transformation to deploy hardware protection on leaky paths automatically. A well-designed control unit is inserted to balance the security and overheads.
- Both simulated and real experiments are performed on the cryptographic design, respectively. Experimental results validate the efficacy of the proposed tool PATHFINDER in the view of security and cost.

The rest of the paper is organized as follows. Section II goes over existing leakage detection and mitigation against SCA attacks. In Section III, we introduce the overall framework of PATHFINDER. Section IV demonstrates the simulated and actual results of PATHFINDER and compares results with existing works. Finally, we conclude the paper in Section V.

II. BACKGROUND

A. Leakage Detection Approaches

Given hardware design details, leakage detection at the pre-silicon stage has the capability to root-cause the source of leakage. This will facilitate designers to strengthen the insecure design prior to fabrication. For example, RTL-PSC first estimates the power profile of register-transfer level (RTL) design, then combines Kullback-Leibler divergence and success rate to identify vulnerable blocks [5]. PLAN performs trace modeling on the RTL codes and computes side-channel vulnerability factors to pinpoint leaking modules within designs [3]. ACA exploits gate-level simulation to rank cells of designs according to contribution quantities of their leaked information [4]. Further, CASCADE enables evaluating the side-channel vulnerabilities within netlist at every design stages [6].

Previous approaches can narrow sources of side-channel leakage to specific modules or even logic cells. This allows designers to apply countermeasures more accurately and only for those vulnerable parts of a design. However, the emphasis on the vulnerable modules or even cells omits the impact of a leaky path. In a leaky path, information leakage propagates among cascaded logic cells, and thus all cells in the path may be vulnerable to SCA attacks. As will be demonstrated in the paper, countermeasures targeting leaky paths avoid the drawbacks of block-level and cell-level approaches. Hence, in this paper, instead of focusing on vulnerable cells/modules, we try to identify and protect the leaky paths leaking sensitive information.

B. Leakage Mitigation Approaches

Various leakage mitigation solutions have been proposed, covering different hierarchies of hardware descriptions and ranging from the whole design to single logic cells. Solutions such as noise injection [7], integrated voltage regulators [8] and current signature attenuation [9] are often applied to the whole circuit or submodules. They exploit additional circuits to introduce noise parts or suppress signal parts of the side-channel information. Although effective on security, protection of the entire block aggravates the burden with aspects of power, area and performance. Another type of solution is secure logic styles, like sense amplifier-based logic (SABL) and wave dynamic differential logic (WDDL). They try to equalize or randomize the side-channel behavior under different conditions. Traditionally, logic cells of the design are replaced globally with corresponding secure logic. Even in [4], the authors apply them locally on identified vulnerable cells, the additional overhead will be unacceptable as vulnerable cells increase.

Inspired by [3] and [4], we attempt to investigate the leakage propagation due to signal connectivity of vulnerable logic cells. This means that we can separate identified cells as different groups, i.e., leaky paths, and embed generic countermeasures on them. The concept of masking is attractive in terms of obfuscating the side-channel information within leaky paths. However, conventional solutions are always algorithmic dependent and make sophisticated modifications to the entire circuit, resulting in high power and area overheads. To address this, we develop the PATHFINDER to automatically deploy masking solutions on the hardware description of leaky paths.

III. METHODOLOGY

In this section, we will introduce the framework of PATHFINDER. Figure 1 illustrates the overall workflow of PATHFINDER that consists of dynamic correlation analysis, static security checking and path obfuscation. The former two steps complete the path identification together.

A. Dynamic Correlation Analysis

Inspired by [4], dynamic correlation analysis ranks logic cells according to their leaked information. This is realized by quantifying the correlation between dynamic power traces of logic cells and given leakage models. Typically, designers treat those cells that leak information beyond the threshold as vulnerable cells. Then mitigation solutions are applied to them. However, dynamic correlation analysis encounters the following challenges when constructing complete leaky paths. Accurate but efficient collections of dynamic traces may be infeasible under limited experimental environments. Also, predefined thresholds may lose those logic cells with group contributions to side-channel leakage. These limitations result in either excessive cost or incomplete results for path identification. To this end, the dynamic correlation analysis of PATHFINDER is designed to elect partial vulnerable cells with topmost leakage contribution (see Figure 1 (a)). Then complete leaky paths are found by following static security checking.

Power Simulation. PATHFINDER first collects the dynamic power of each logic cell by gate-level power simulations. Given a set of stimuli, we perform function simulation on the post-layout netlist back-annotated accurate delay information. The switch activities of all logic cells are recorded over a period. This file, along with the technology library, design netlist and parasitic data, are imported to Primetime PX for time-based power analysis.

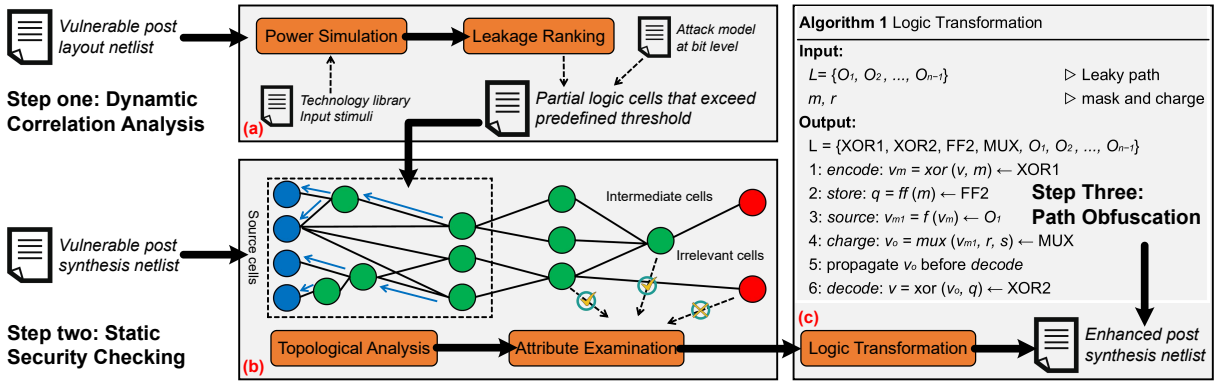

Fig. 1: The overall framework of PATHFINDER.

Leakage Ranking. PATHFINDER then quantifies the information leakage of each logic cell, where the maximum Pearson correlation coefficient ρ serves as the leakage criterion. As shown in Equation (1), this criterion C manifests the dependency between power traces P and the leakage model L. Here the leakage model denotes the Hamming weight or Hamming distance of sensitive variables per bit v. It lies on the designer to select sensitive variables for protection, such as keys of cryptographic algorithms. PATHFINDER ranks logic cells within the netlist from highest leakage criterion to lowest. According to a predefined threshold, we obtain a group of logic cells carrying most information leakage through traversing all data bits.

$$C = max(|\rho(P, L)|) \tag{1}$$

B. Static Security Checking

For one leaky path, the information leakage caused by sensitive variables will start from the source cell, pass through multi-level intermediate cells until irrelevant cells. In general, the source cell leaks the most information. Thus we can locate source cells in partial logic cells obtained from the above step. As illustrated in Figure 1 (b), PATHFINDER uses topological analysis to achieve this goal. While attribute examination expands subsequent intermediate cells for complete leaky paths.

Topological Analysis. As the post-synthesis netlist consists of several modules, the topological analysis first splits it into individual module files. Next, PATHFINDER describes the intrinsic connectivity of every module using two dictionaries. One dictionary, named CELL, lists signal nets passing in and out per logic cell. The other dictionary, named NET, records logic cells that connect before and after per signal net. With the help of CELL and NET, we can quickly stepwise inference antecedent network or consequent network from the certain logic cell. Hereafter, PATHFINDER back annotates partial logic cells into the post-synthesis netlists and builds antecedent networks within the bound of this group. Those cells at the head of antecedent networks are source cells in the leaky paths. Sensitive variables begin to propagate from source cells and thus produce power profiles highly correlated with them.

Attribute Examination. Starting from source cells, PATHFINDER will construct consequent leaky paths by attribute examination. So-called attribute examination aims to check whether certain logic cell inherits the power profile closely related to sensitive variables. Here we define the concept of the leakage attribute during attribute examination.

Take one logic cell as instance, the cell receives the sensitive variable v_c from the antecedent source cell or intermediate cell. Its logic function reads $a_1, a_2, ..., a_{n-1}, v_c$ and then outputs O_c. Considering the sensitive variable changes from v_c to v_p, state transition from O_c to O_p will happen. State transitions $v_c \rightarrow v_p$ and $O_c \rightarrow O_p$ determine the power profile of the logic cell and its antecedent cell, respectively. If the logic cell belongs to leaky paths, it should demonstrate a similar power profile as the known antecedent source cell or intermediate cell. Equation (2) formulates the above relation as the leakage attribute. Here we neglect the slight divergence between power profile of state transition $0 \rightarrow 1$ ($0 \rightarrow 0$) and $1 \rightarrow 0$ ($1 \rightarrow 1$). This means that an intermediate cell will spread state transitions of sensitive variables to its output net, no matter how other input nets change. We exploit the leakage attribute to check all logic cells in the technology library and annotate agreeable cells beforehand. Through examining the gate type, PATHFINDER constructs complete leaky paths step by step for all modules. The final leaky paths are manifested as networks that flow across all data bits of sensitive variables.

$$\forall(a_1, a_2, ...a_{n-1}), \left|O_c - O_p\right| = \left|v_c - v_p\right| \tag{2}$$

C. Path Obfuscation

Once leaky paths have been identified, the PATHFINDER automatically selects simple but efficient solutions such as Boolean masking and random precharge to protect them locally, as illustrated in Figure 1 (c). Logic transformation translates protection solutions on post-synthesis netlist.

Logic Transformation. The final hardware scheme is shown in Figure 2 (a), containing origin logic (black part), Boolean masking (blue part) and random precharge (green part). Take one path, for example, PATHFINDER starts Boolean masking at source cells usually of sequential logic. An XOR gate encodes the sensitive variable v with uniformly random masks m (Line 1). The masked variable v_m propagates throughout the leaky path so that the whole data flow is obfuscated (Lines 3 and 5). The same XOR gate will be added

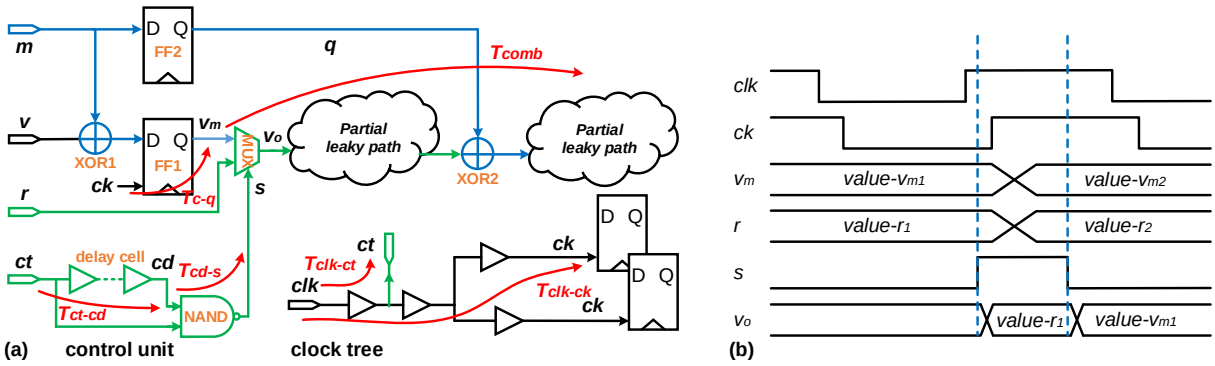

Fig. 2: The hardware scheme of protection solutions, including Boolean masking (blue) and random precharge (green).

before irrelevant cells to decode masked variables (Line 6). Accordingly, a flip-flop (FF) is required to store and deliver the masks. PATHFINDER allows users to define the number of masks and also their paired FFs (Line 2). Then a 2-to-1 MUX gate is inserted between the source cell and intermediate cells (Line 4). The select input s determines which data input (masked variable v_m or random charge r) is presented to the output. For proper implementations, one important issue is that how to control the protection scheme. PATHFINDER inserts a control unit that comprises a NAND gate and some delay cells. It imports control signal ct close to the clock source and outputs the short pulse signal s to the MUX gate. This cell allows the random charge to be transmitted when s remains logic high, otherwise transfers values of the source cell. Figure 2 (b) demonstrates timing diagram of relevant signals. Therefore, settings of the control signal and delay cells have a strong impact on both security and performance, which must satisfy the following conditions:

$$T_{clk-ct} + T_{comb} + T_{cd-s} > T_{clk-ck} + T_{c-q} + T_{hold} \quad (3)$$

$$T_{clk-ct} + T_{cd-s} < T_{clk-ck} + T_{c-q} \quad (4)$$

$$T_{clk-ct} + T_{ct-cd} + T_{cd-s} > T_{clk-ck} + T_{c-q} \quad (5)$$

$$T_{clk-ct}+T_{ct-cd}+T_{cd-s}+T_{comb} < T_{clk-ck}+T_{cycle}-T_{setup} \quad (6)$$

where T_{cycle}, T_{setup} and T_{hold} denote the clock period, setup time and hold time, respectively. Types of T_{a-b} denote the delay from signal a to signal b. T_{comb} is the total delay of combinational logic behind FF1. Equation (3) and Equation (4) define the time range where the positive edge of the pulse signal s arrives. If the positive edge arrives early, random charges will deliver to the next level FFs, which results in wrong results. In contrast, random charges do not take effect on partial leaky paths when the positive edge arrives later. While the arrival time of its negative edge is confined by Equation (5) and Equation (6). It must guarantee that no sampling errors occur when transferring values of source cells. Also, we need to avoid setup time violations due to extra delay. By solving the above equation, we determine the location of ct and the number of delay cells. In turn, PATHFINDER raises new timing demand to the placement and routing stage. The tool updates the timing constraint, increasing the clock uncertainty by $|T_{clk-ct} + T_{cd-s} - T_{clk-ck}|$ delays.

Security Analysis. Considering sensitive variable per bit, state transition $v_1 \rightarrow v_2$ occurs before protection. After applying PATHFINDER, this transition turns into $r \rightarrow (v_1 \oplus m)$ for combinational logic, while $(v_1 \oplus m) \rightarrow (v_2 \oplus m)$ for sequential logic. Since the mask m and charge r are uniformly random at each cycle, PATHFINDER shuffles the state transition so that obfuscates dynamic power. Moreover, due to state transitions, logic cells create currents across metal wires and thus emit EM emanations. Hence obfuscated state transitions also protect the design from EM SCA attacks.

IV. EXPERIMENTAL RESULTS

In this section, we use a 128-bit AES circuit (denoted as AES-128) to validate the efficacy of PATHFINDER on side-channel protection. At first, simulation results are exploited to evaluate the hardened AES-128 in ASIC implementation. Next, we implement the hardened AES-128 using FPGA and carry out actual measurements. Note that security evaluations are performed by correlation analysis attacks (CPA/CEMA) in both experiments.

A. PATHFINDER *Configuration*

We apply logic synthesis to implement the AES-128 design using 180 nm CMOS technology. The post-synthesis netlist consists of 10 modules and 10083 logic cells. Its supply voltage and clock frequency are set as 1.8 V and 25 Mhz, respectively. Then the design goes through placement and routing to obtain the final physical layout. Meanwhile, the required files of PATHFINDER are collected, including post-layout netlist, delay information and parasitic data.

As described in Section III, PATHFINDER will identify and obfuscate leaky paths to harden the AES-128 design. Table I lists configuration parameters of PATHFINDER. During dynamic correlation analysis, we obtain 1082 partial logic cells under 1000 input stimuli and leakage criterion > 0.95. Total 2120 logic cells compose complete leaky paths after static security checking. In path obfuscation, two pseudo-random number generators (PRNG) are inserted to produce 32 random masks and random charges respectively. PATHFINDER takes about 18.54 min, 9.12 s and 0.13 s for the above three steps.

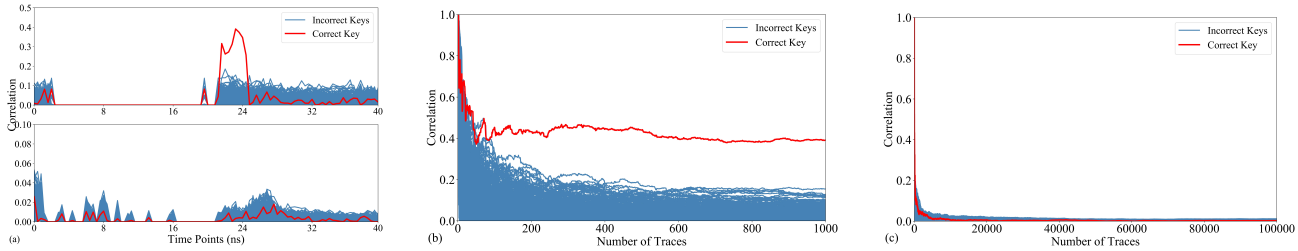

Fig. 3: (a) Correlation traces as a function of time points obtained from unprotected (top) and protected (bottom) design. Correlation traces as a function of stimuli number obtained from (b) unprotected and (c) protected design.

TABLE I: PATHFINDER parameters for the Experiment

Input stimuli	Leakage criterion	Masks	Charges
1000	0.95	32	32

B. Analysis of Simulated Results

To evaluate the security, we perform time-based power simulations on post-layout netlists using the Primetime PX tool. Total 100000 dynamic traces are collected for unprotected and protected designs, respectively. Then CPA attacks are carried out on the time periods where SubBytes operation of first four bytes executes. In CPA attacks, the recovered key with the maximum Pearson correlation coefficient ρ_{max} indicates the most possible correct hypothesis. When the number of traces exceeds a certain value, the ρ_{max} of the correct hypothesis will always be greater than those of wrong guesses. This certain value, often named as measurement to disclosure (MTD), denotes the minimum traces to disclosure the key. Both the above two metrics are used in this paper to represent the resistance of designs against SCA attacks.

Figure 3 (a) shows trends over time of the maximum correlation ρ_{max} for all key candidates. For unprotected design, the maximum correlation of the correct key approximates 0.391 (top sub-figure). Results indicate that its side-channel behaviors highly correlate with leakage models used by the attacker. Hence the attacker recovers the correct key within 84 traces (Figure 3 (b)). While for design equipped with PATHFINDER, hardware protections reduce the maximum correlation to 0.018 (bottom sub-figure). This makes the correlation trace of correct key submerge by those of wrong key candidates. Therefore, the attacker does not steal the correct key even with 100 K traces (Figure 3(c)). PATHFINDER provides at least 1190× improvement in the perspective of power side-channel resistance. Table II lists overheads of PATHFINDER. After hardened by PATHFINDER, the AES-128 requires 10742 logic cells, resulting in an increase of 659 cells. Moreover, PATHFINDER introduces extra 600 μW power, prompting the power consumption to change from 13.3 mW to 13.9 mW. The maximum clock frequency decreases from 45.5 Mhz down to 44.1 Mhz.

C. Analysis of Actual Results

To demonstrate the effectiveness of PATHFINDER against EM SCA attacks, we perform actual measurements on designs deployed in the SAKURA-G platform. Since the PATHFINDER is designed for ASIC chips, the post-synthesis netlists need to be adjusted according to FPGA applications. Here we rewrite underlying modules of every logic cells in the technology library. Figure 4 illustrates this procedure using a logic cell DFFRHQX1, involving its functions and rewritten RTL module. During Xilinx FPGAs development, tools will compile and implement post-synthesis netlists using fundamental elements, such as LUTs, FFs, carry-chain logic and multiplexers. Also, we use hardware primitives to achieve the same functions as the control unit. In these processes, attributes like KEEP and DONT_TOUCH are set on added signals and modules to prevent logic optimizations.

Functions // DFFRHQX1

RN	D	CK	Q[n+1]
0	x	x	0
1	0	↑	0
1	1	↑	1
1	x	↓	Q[n]

```
module DFFRHQX1 (Q, D, CK, RN);
output reg Q;
input  D, CK, RN;

always@(posedge CK or negedge RN)
begin
 if (!RN)
   Q <= 1'b0;
 else
   Q <= D;
end

endmodule // DFFRHQX1
```

Fig. 4: The functions and RTL module of DFFRHQX1.

During silicon measurements, one RF-B LANGER probe is placed on the vicinity of the SAKURA-G board to collect EM traces. We adjust the probe location in advance till the amplitude of EM traces in the oscilloscope reaches the maximum. The fixed key and 100000 random input stimuli are delivered to the AES-128 design for data encryption. Total 800000 traces are recorded for the AES-128 design, under 2.5 GSa/s sampling rate. Then we take the average of 8 measurements (with the same input stimuli) aligned in the time domain through elastic alignment as the final data. Figure 5 demonstrates the CEMA attack results. In the original AES-128, the correlation trace of the correct key stands out compared to those of the wrong keys. After applying PATHFINDER, the maximum correlation of the correct key decreases from 0.19 to 0.02. Therefore, no information leakage can be observed by the attacker. Meanwhile, the CEMA attack recovers the correct key within 92 traces for the original design (Figure 5 (b)). However, the same attack targeting the hardened design does not recover the correct key even after 100 K EM traces (Figure 5 (c)). It can be seen that the tool also improves the EM side-channel resistance, with an increase of MTD at least 1085×.

D. Comparison with Existing Works

Table II summarizes the comparison between our work and existing methods, in terms of MTD improvement, area,

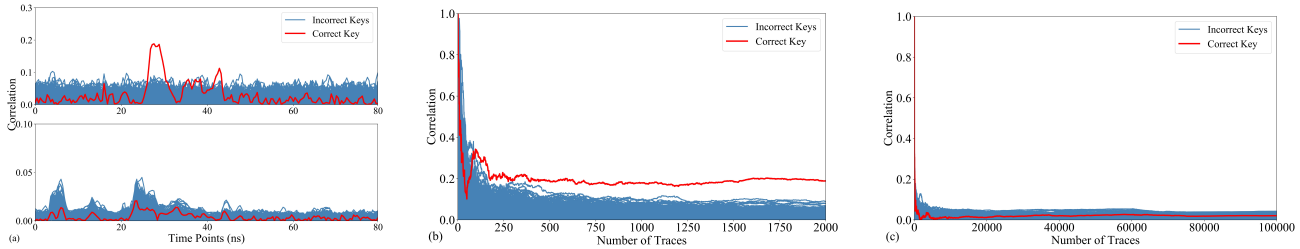

Fig. 5: (a) Correlation traces as a function of time points obtained from unprotected (top) and protected (bottom) design. Correlation traces as a function of stimuli number obtained from (b) unprotected and (c) protected design.

TABLE II: Comparison with existing works.

Works	MTD Improv.		Overheads		
	Power	EM	Area	Power	Perf.
Moradi [2]	100×	–	359 %	262 %	40^a
Moradi [10]	10000×	–	196 %	–	20^a
Yao [4]	$4\times^b$	–	10 %	–	–
SLPSK [11]	107×	–	0 %	0 %	0 %
KF [3]	16×	–	31.9 %	–	31.25 %
Singh [8]	4210×	136×	$96.7\ \%^c$	32 %	10.4 %
Das [12]	–	167×	23 %	49 %	0 %
Das [9]	125000×	83333×	36.7 %	49.8 %	0 %
This Work	$1190\times^d$	$1085\times^d$	6.53 %	4.51 %	3.1 %

– Data has not been reported, ^{a} Increase clock cycles, ^{b} Decrease the maximum correlation, ^{c} Area overhead includes 1.9 nF load capacitor, ^{d} Only 100 K traces are collected limited by experiment conditions.

power and performance overheads. Moradi et al. [2] design the threshold implementation of the AES design, yielding an improvement of 100× in MTD metric. However, the area and power overheads increase by 3× and 2×, respectively. Recently, they optimize the traditional method [10]. Yao et al. [4] replace identified vulnerable cells with WDDL, reducing maximum correlation by 4× with a 10 % area increase. SLPSK et al. [11] propose gate-sizing to improve MTD by 107× without overheads, but bring about increased difficulties in chip fabrication. KF et al. [3] apply a 4-round Feistel structure on leaky modules to obfuscate data. Results report MTD improvement of 16× while sacrificing 31.9 % area and 31.25 % performance. Singh et al. [8] propose a security-aware all-digital low-dropout (DLDO) regulator. This method increases power and EM SCA resistance by 4210× and 136×, respectively. However, the area overhead is about 100 % due to the large load capacitor. The method proposed in [12] has a 167× MTD improvement, with 1.23× area overhead and 1.5× power overhead. An enhanced version of their work can be found in [9]. In our work, PATHFINDER raises the MTD by 1190× and 1085× against power and EM SCA attacks. This hardware protection only incurs 6.53 %, 4.51 % and 3.1 % overheads with respect to area, power and performance.

V. CONCLUSION

In this paper, we propose a novel EDA tool PATHFINDER for automatic side-channel protection. This tool enables designers to identify leaky paths by combining dynamic and static procedures. Then well-designed hardware solutions such as Boolean masking and random precharge are inserted to protect the gate-level netlist. Simulations and actual measurements are performed targeted at the hardened cryptographic design. Experimental results verify that PATHFINDER can enhance the side-channel resistance by at least 1000×, while introducing slight impacts on the area, power and performance.

ACKNOWLEDGMENTS

This work is supported in part by the National Key R&D Program of China (Grant No. 2021YFB3100903), and in part by the National Natural Science Foundation of China (Grant No. 62004112).

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*. Springer, 1999, pp. 388–397.

[2] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of AES," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2011, pp. 69–88.

[3] M. A. KF, V. Ganesan, R. Bodduna, and C. Rebeiro, "PARAM: A microprocessor hardened for power side-channel attack resistance," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 23–34.

[4] Y. Yao, T. Kathuria, B. Ege, and P. Schaumont, "Architecture correlation analysis (ACA): identifying the source of side-channel leakage at gate-level," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 188–196.

[5] M. He, J. Park, A. Nahiyan, A. Vassilev, Y. Jin, and M. Tehranipoor, "RTL-PSC: Automated power side-channel leakage assessment at register-transfer level," in *Proceedings of the 37th IEEE VLSI Test Symposium (VTS)*. Monterey, CA, USA: IEEE, 2019, pp. 1–6.

[6] D. Sijacic, J. Balasch, B. Yang, S. Ghosh, and I. Verbauwhede, "Towards efficient and automated side channel evaluations at design time," *Kalpa Publications in Computing*, pp. 16–31, 2018.

[7] X. Wang, W. Yueh, D. B. Roy, S. Narasimhan, Y. Zheng, S. Mukhopadhyay, D. Mukhopadhyay, and S. Bhunia, "Role of power grid in side channel attack and power-grid-aware secure design," in *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 78.

[8] A. Singh, M. Kar, V. C. K. Chekuri, S. K. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "Enhanced power and electromagnetic SCA resistance of encryption engines via a security-aware integrated all-digital ldo," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 2, pp. 478–493, 2019.

[9] D. Das, J. Danial, A. Golder, N. Modak, S. Maity, B. Chatterjee, D. Seo, M. Chang, A. Varna, H. Krishnamurthy *et al.*, "27.3 EM and power SCA-resilient AES-256 in 65nm cmos through $>$ 350× current-domain signature attenuation," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 424–426.

[10] A. R. Shahmirzadi and A. Moradi, "Re-consolidating first-order masking schemes," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 305–342, 2021.

[11] P. Slpsk, P. K. Vairam, C. Rebeiro, and V. Kamakoti, "Karna: A gate-sizing based security aware EDA flow for improved power side-channel attack protection," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.

[12] D. Das, M. Nath, B. Chatterjee, S. Ghosh, and S. Sen, "STELLAR: A generic EM side-channel attack protection through ground-up root-cause analysis," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2019, pp. 11–20.