SCARE: Side Channel Attack on In-Memory Computing for Reverse Engineering

Sina Sayyah Ensan, Karthikeyan Nagarajan, Mohammad Nasim Imtiaz Khan, and Swaroop Ghosh, *Senior Member, IEEE*

***Abstract*— In-memory computing (IMC) architectures provide a much needed solution to energy-efficiency barriers posed by Von-Neumann computing. The functions implemented in such in-memory architectures are often proprietary and constitute confidential intellectual property (IP). Our studies indicate that IMC architectures implemented using resistive RAM (RRAM) are susceptible to side channel attack (SCA). Unlike the conventional SCAs that are aimed to leak private keys from cryptographic implementations, SCA on IMC for reverse engineering (SCARE) can reveal the sensitive IP implemented within the memory through power/timing side channels. Therefore, the adversary does not need to perform invasive reverse engineering (RE) to unlock the functionality. We demonstrate SCARE by taking recent IMC architectures, such as dynamic computing in memory (DCIM) and memristor-aided logic (MAGIC) as test cases. Simulation results indicate that AND, OR, and NOR gates (which are the building blocks of complex functions) yield distinct power and timing signatures based on the number of inputs, making them vulnerable to SCA. We show that adversary can use templates (using foundry-calibrated simulations or fabricating known functions in test chips) and analysis to identify the structure of the implemented function by testing a limited number of patterns. We also propose countermeasures, such as redundant inputs and expansion of literals. Redundant inputs can mask the IP with 25% area and 20% power overhead. However, functions can be found at higher RE effort. Expansion of literals incurs 36% power overhead. However, it imposes a brute force search increasing the adversarial RE effort by 3.04×.**

***Index Terms*— In-memory computing (IMC), side channel attack (SCA).**

I. Introduction

IN-MEMORY computing (IMC) is a promising compute model to minimize data transfer between processor and memory by confining data within compute-capable memory [1]–[3]. Several works have been proposed to move compute logic closer to the main memory or infuse the processing capability into memory cells [4]–[9]. IMC can perform specific tasks, such as dot-products [6] and search [10], or support a wide range of logic [11] and arithmetic operations (e.g., matrix multiplication [12]). It is true that designing a random intellectual property (IP) using memory devices on-chip offers no benefit over CMOS gates. However, IMC will benefit data-intense applications such as machine learning and/or cryptographic operation such as hashing that may also contain IP.

The compute capabilities of conventional memories, such as static RAM (SRAM) and dynamic RAM (DRAM) have been heavily studied [1], [13]–[15]. Furthermore, IMC is achievable using emerging nonvolatile memories (NVMs), e.g., resistive RAM (RRAM), spin-transfer torque (STT) magnetic RAM, phase change memory, etc. RRAM-based IMC architectures in particular, have exhibited significant promise due to low-power consumption, fast operation, and high integration density ($4F^2$ footprint in crossbar architecture [16]). RRAM provides higher sense margin and many analog states compared to other NVMs, e.g., STTRAM. Although RRAM may not be preferred for storage applications due to poor write endurance, it is useful for IMC since write operation is only needed once (to program the function).

Existing IMC architectures, such as dynamic computing in memory (DCIM) [5], memristor-aided logic (MAGIC) [17], material implication (IMPLY) [18], etc. are only capable of implementing functions with a limited number of inputs. In this article, we show that RRAM-based IMC can reveal the implemented IP through power/timing side channel. Specifically, we propose SCARE (SCA on IMC for reverse engineering taking RRAM-based IMC architectures as test case. Specifically, we show that the number of minterms and the number of inputs per minterm can be revealed by power/timing side channel attack (SCA).

A. Example of SCARE Attack

SCARE considers that the IMC operation is carried out in two cycles to compute a function in the sum-of-product (SOP) form (see Fig. 1). The first cycle computes the AND and the second cycle computes the OR [5]. In this example, SCARE involves the following steps.

1) Extraction of current profile during the compute cycles.
2) Matching the 2nd cycle current profile with one of the template current profiles to determine the number of minterms implemented within the OR array. Here, the minterms represent the fan-in of the OR gate.

Manuscript received February 7, 2021; revised June 7, 2021 and July 26, 2021; accepted August 11, 2021. Date of publication November 3, 2021; date of current version November 30, 2021. This work was supported in part by the Semiconductor Research Corporation (SRC) under Grant 2847.001; and in part by NSF under Grant CNS-1722557, Grant CCF-1718474, Grant CNS-1814710, Grant DGE-1723687, and Grant DGE-1821766. *(Corresponding author: Sina Sayyah Ensan.)*

The authors are with the School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, PA 16802 USA (e-mail: sxs2541@psu.edu; kxn287@psu.edu; muk392@psu.edu; szg212@psu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3110744.

Digital Object Identifier 10.1109/TVLSI.2021.3110744

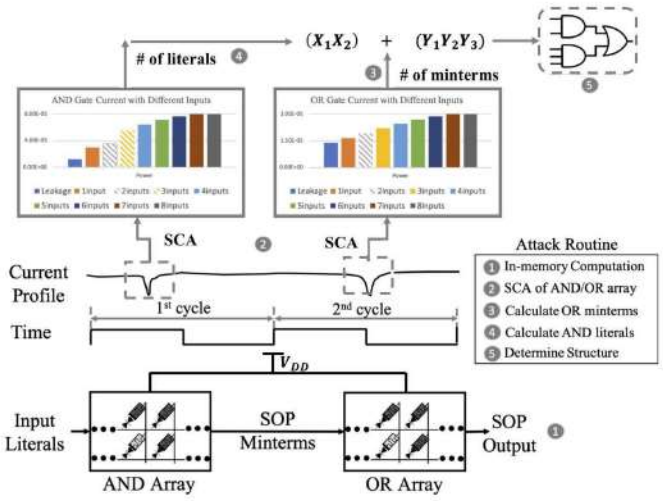


Fig. 1. Example of side channel attack (SCA) to extract the structure of implemented function in IMC.

3) Matching the 1st cycle current profile with one of the template current profiles to determine the number of literals in the AND array, i.e., the fan-in of the AND gates. Knowing the number of minterms from the 2nd cycle allows the adversary to determine the number of input literals per SOP minterm.
4) After finding the function structure, extracting the implemented function by applying a limited number of patterns to the chip and validating using a golden chip.

B. Baseline Attack Model

We assume that adversary can obtain template current profiles developed through foundry-calibrated simulations or by fabricating small known functions in a test chip (more details in Section II-D) to aid in the RE process. For brevity, this paper focuses on two RRAM-based IMC architectures, namely DCIM [5] and MAGIC [17] to demonstrate SCARE. However, the attack can also be implemented on other emerging IMC with minor changes. It is noted that the process variation (PV) can lead to a large variation in power/timing profile of implemented gates. This, in turn, can make RE challenging due to the overlap of signatures from two different gates (e.g., two-input AND gate power can overlap with three-input AND due to variations). However, the adversary can use statistical analysis of power/delay to filter the correct function (details in Sections III and IV).

C. Distinctions From Memory SCA

SCA on traditional memories have been studied in the past. Side channel leakage of ASIC components [19] shows that SRAM's leakage closely follows a generic hamming distance. Differential Power Analysis on STTRAM has been investigated [20] to decode the memory contents. It is noted that the write current-based SCA will not work for all IMC architectures due to the absence of write operation (e.g., in DCIM) during computing. Furthermore, IMC computing is different than the read operation due to the absence of static read current during computing. Therefore, read current-based SCA is not directly applicable to IMC. Furthermore, the objective of SCA-based RE is to extract the implemented function in contrast to SCA-based key extraction which only identifies the hamming weight of the data.

D. Distinctions From Conventional RE

RE is generally an invasive and destructive form of analyzing integrated circuits (ICs) where an adversary grinds away each layer of an IC and captures optical images. The base layer provides gate types and the upper layers provide their connectivity. By combining the information, the IP could be unlocked. In contrast, SCARE is a noninvasive RE approach that exploits SCA to extract IP implemented in an emerging memory technology-based IMC. With SCARE, the extracted structure reveals the number of input literals and the number of SoP minterms. The extracted structure can be further used to craft a limited number of input patterns to extract the overall SoP function (not covered in this article). Based on our literature survey, SCARE is the first work on RE of IMC-based IP.

E. Contributions

In this article, we:

1) Investigate SCA on IMC architectures for noninvasive RE of IP;
2) Exploit side channel current templates to identify the gate structures of the implemented functions;
3) Propose two attack models for DCIM and MAGIC, respectively. One works for true inputs only and other one works for both, true and complementary inputs;
4) Conduct PV analysis of the IMC architectures to develop an SCA comparison model;
5) Propose countermeasures, such as redundant inputs and expansion of literals to protect from SCARE.

II. Background and Related Work

A. Basics of RRAM

RRAM is a two terminal device with a resistive switching layer sandwiched between two electrodes. Switching from low resistance state (LRS) to high resistance state (HRS) is called "reset" process and switching from HRS to LRS is called "set" process. The resistance of the insulator layer changes from LRS (HRS) to HRS (LRS) depending on the voltage polarity between the two terminals [21].

B. Basics of IMC

1) Dynamic Computing in Memory (DCIM): We have implemented the DCIM-based [5] logic function $AB + CD$ as an example [see Fig. 2(a)]. The corresponding timing waveform is shown in Fig. 2(b). Each memory cell consists of an RRAM connected in series with a selector diode. Functions are implemented in SoP form using pre-programmed memory arrays. Separate arrays for AND and OR operations are needed to implement the logical functions. Inputs are given

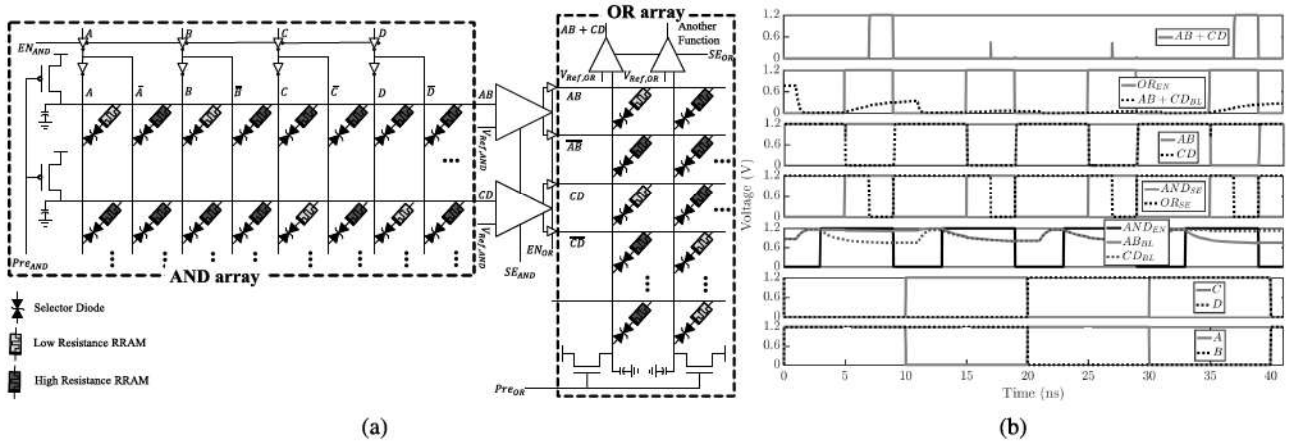


Fig. 2. (a) Implementation of $AB + CD$ using DCIM [5] in RRAM crossbar and (b) timing diagram of the function implementation.

to the arrays through Wordlines (WLs). Final Bitline (BL) voltages are considered as outputs (AND array BLs implement minterms, e.g., AB and CD, and, OR array BLs implement functions). Any LRS RRAM in AND array's BLs is considered as a literal to the respective BL and any LRS RRAM in OR array's BLs serves as a minterm for the implemented function.

Initially, AND array's BLs are precharged to V_{DD} by asserting Pre_{AND}. Then, inputs are applied to the RRAMs by activating EN_{AND} signal. If one of the literals of AND array's BLs is "0," the respective BL gets discharged and its voltage drops below the sense amplifier (SA) reference voltage ($V_{\text{Ref-AND}}$) and the minterm is considered as "0." If all the inputs are logically "1," the BL holds its precharged value which is higher than the reference voltage and is considered as "1."

The BLs of OR array are initially predischarged to "0." After activating OR_{EN} signal if one of the input literals in a BL is logically "1" it can charge up the BL to a value higher than OR array's reference voltage ($V_{\text{Ref-OR}}$). Finally, the voltage of OR array's BLs are compared against $V_{\text{Ref-OR}}$ at the edge of SE_{OR} and an output is generated.

2) Memristor-Aided Logic (MAGIC): We have also implemented MAGIC architecture [17] that employs memristors (RRAM in this article) to implement logic gates. A number of memristors serve as inputs to previously stored data while an additional memristor acts as the output. Gates including MAGIC-NOR, AND, OR, and NAND are shown in Fig. 3(a)–(d), respectively. MAGIC's logical state is represented as a resistance, where the HRS and LRS represent logical "0" and "1" respectively. Fig. 3(e) shows the implementation of $AB + CD$ as an example. It consists of two 2-fan-in AND gates and one 2-fan-in OR gate. Here, the input RRAMs, A, B and D, are initialized to logical "1" (LRS) and RRAM C is initialized to logical "0" (HRS). All output RRAMs are initialized to "0" (HRS). In the first cycle, AB is computed by asserting its bitline driver (BL_{AB}) using the enable (EN_{AB}) signal. Since $AB = 1$, AB's output RRAM switches from "0" (HRS) $\rightarrow$ "1" (LRS). Similarly, during the second cycle, when BL_{CD} is asserted using the EN_{CD} signal, CD's output RRAM remains at "0" (HRS). During the final cycle, the BL driver for the OR_2 operation (BL_{AB+CD}) is asserted using the EN_{AB+CD} signal. Fig. 3(e) shows that the final output RRAM switches from "0"$\rightarrow$"1" and reflects the correct output of $AB + CD = 1.1 + 0.1 = 1$.

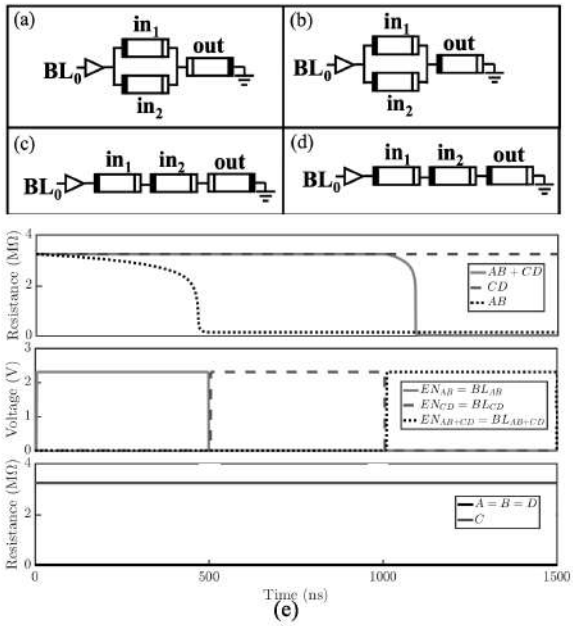


Fig. 3. Implementation of MAGIC gates, (a) NOR; (b) OR; (c) NAND; and (d) AND. (e) Timing diagram of $AB + CD$ implementation on MAGIC architecture.

C. Background on Side Channel Attack

SCA [19] exploits the unintentional signature observed in physical channels such as timing [22], power consumption [19], electromagnetic emanation [23], etc. to recover the sensitive data being processed, e.g., cryptographic keys. Since different data bits exhibit different physical signatures (power consumption, delay), SCA can unveil the data.

In [24], authors noticed that supply noise is related to write and read operations (based on the previous data and new data) and cause a read and write failure. In [25], authors state that SRAM is prone to power attacks.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
MOSFET Gate Length	65 nm
NMOS/PMOS Threshold Voltage	423/-365 mV
BL Capacitance	100 fF
RRAM Gap Min/Max/Oxide Thickness	0.1/1.7/5 nm
Atomic Energy: Vacancy Generation/Recombination	1.501/1.5 eV
RRAM Write Latency	25 ns
RRAM HRS/LRS at 1.2V	6.7M/58.9K Ω

TABLE II
MONTE CARLO SIMULATION PARAMETERS

Parameter	Real Value	Variation	STD. Deviation
RRAM LRS Gap	0.1 nm	7%	3σ
RRAM HRS Gap	1.7 nm	7%	3σ
MOS Oxide Thickness	1.2 nm	10%	3σ
MOS Gate Length	65nm	10%	3σ

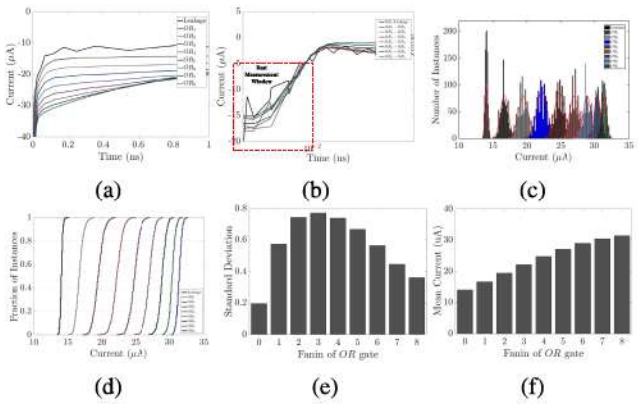


Fig. 4. (a) Analysis of OR gate current profile with various fan-ins; (b) current difference with fan-in of n and $n+1$ to find the measurement window; (c) PDF, (d) CDF, (e) STD and, (f) mean current of OR gates.

D. Adversarial Templates

To correlate the IMC power/timing (extracted using SCA) with the appropriate gate type and fan-in value, the adversary requires a precalculated power and timing template which may be easily developed if the adversary has access to the foundry calibrated device models. If not, a well-funded adversary can order a limited number of test chips that implement multiple small known functions using IMC. Such an opportunity is available through shuttle programs of vendors such as CMP and MOSIS. The adversary can then proceed to develop templates based on the power and timing distributions calculated for different gates and input sizes.

III. ATTACK ON DCIM ARCHITECTURE

Adversary can distinguish the power drawn by the OR and AND arrays of DCIM by exploiting their power signature. Two attack models are proposed to find the structure of the implemented functions on DCIM. Attack model 1 works if the inputs of the implemented function are only true inputs (a, b, c, etc.). Attack model 2 works on any implemented function however it needs more RE effort to find the base condition of the attack.

A. Simulation Setup

The simulations are performed in HSPICE with 65-nm PTM technology [26], ASU RRAM model [27], and bidirectional selector diode model [28]. Detailed parameters of the devices employed for simulations are shown in Table I. To mimic the PV in test chip obtained by adversary, we introduce PVs to the SCARE power profile/operation-time modeling by performing Monte Carlo simulations with the parameters listed in Table II.

B. Attack Model 1

1) Leveraging the Power Drawn by OR *and* AND *Array:* In DCIM, computations are performed in two cycles between EN signal and SE signal activation. SCARE especially considers the peaks in the power profile for three reasons: 1) EN signal activates large buffers (that are upsized to charge/discharge the BLs quickly); 2) SE signal activates SAs (significant capacitive component); and 3) short-circuit current from V_{DD} to ground through buffers/SAs (when both pull up/down networks are ON).

Initially, the adversary chooses two consecutive time periods between EN and SE in each cycle to launch the attack. This is followed by asserting all the input signals to logical "1"s and recording the power profile. The recorded power during the second cycle (i.e., the OR function) is matched with the power template. This step allows the adversary to determine the number of SoP minterms processed in the 2nd cycle (OR function input literals). By setting all the input literals to logical "1," the adversary ensures that each SoP minterm in the function also equals a logical "1." The BLs in the DCIM OR array are pre-discharged to "0." Thus, by applying the logical "1" to all SoP minterms, SCARE ensures that the BLs are charged as fast as possible.

Once the number of SoP minterms is determined, the adversary analyzes the power profile of the 1st cycle to determine the number of input literals in each minterm. Each of the AND array input literals are set to logical "0" to ensure that each SoP minterm equals to "0." By applying "0" to each minterm, the adversary ensures that the BL, which is precharged to V_{DD}, discharges at the highest possible rate. Finally, by analyzing the power profile of the operation, the adversary determines the number of input literals for each AND gate (see Fig. 1). Note, the above attack model works only with true inputs. If the function consists of complementary inputs, the attack will fail since adversary cannot confirm if all the minterms are "1" by forcing all inputs to "1."

2) Simulation Results:

a) DCIM OR *array:* The current profiles of the OR gate with fan-in ranging between 0 to 8 is shown in Fig. 4(a) (time offset is chosen when EN signal is activated). It indicates that OR gates with various fan-ins charge the BLs at different rates. Since the observed resistance of each BL decreases with the number of LRS inputs (more number of resistors in parallel), current through the array increases with fan-in. However, the separation between the current profiles of each gate decreases

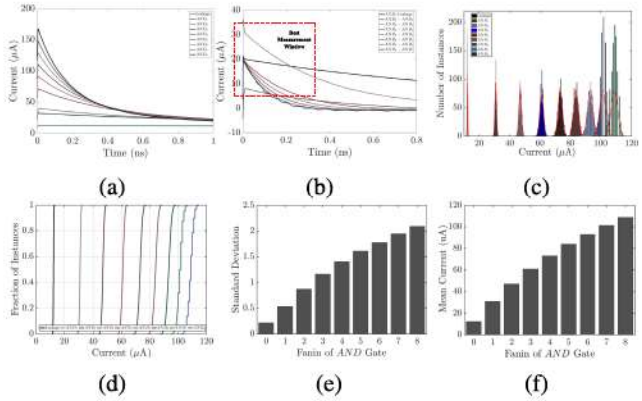


Fig. 5. (a) Analysis of AND gate current profile with various fan-ins; (b) current difference with fan-in of n and $n+1$ to find the measurement window; (c) PDF, (d) CDF, (e) STD and, (f) mean current of AND gates.

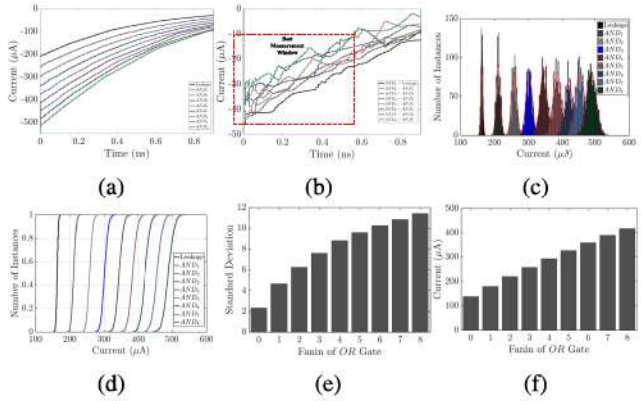


Fig. 6. (a) Analysis of AND gate current profile during precharge phase; (b) current difference of AND gates with fan-in of n and $n+1$ to find the measurement window; (c) PDF, (d) CDF, (e) STD, and (f) mean current of AND array's BLs currents.

with fan-in. With time, the current profiles of various fan-in gates become indistinguishable. The current profile merges when the BL is charged up to $V_{DD} - V_{th,Diode}$. Therefore, selection of measurement window is critical.

Shorter measurement windows increase the resolution for distinguishing fan-ins of OR gates [the best measurement window is shown in Fig. 4(b)]. In the suggested measurement window, the currents of OR_N and OR_{N+1} gates differ by at least 5 μA. This is evident from the probability density function (PDF) of the current profiles of OR gates [see Fig. 4(c)]. It is noted that the PDF of OR_7 and OR_8 have a noticeable overlap. If the recorded current profile (from SCA) falls within this overlap, the adversary might need to consider both possibilities. In such cases, the cumulative distribution function (CDF) may be useful to predict the number of inputs [see Fig. 4(d)]. Additional properties of the distribution, such as the mean and standard deviation (STD) may be used to accurately determine the gate fan-in. The STD of scor gates with different fan-ins [see Fig. 4(e)] does not follow a monotonically increasing or decreasing trend. In contrast, the mean values [see Fig. 4(f)] exhibit a monotonically increasing trend with fan-in. Thus, the adversary can leverage the CDF, PDF, and mean distributions of the OR gate current profile templates collected from the test chips/simulation to analyze the current profile recorded by SCA.

b) DCIM AND array: In this case, the adversary monitors the ground node to extract the current profiles. The current profile of the AND gate with various fan-in is shown in Fig. 5(a). Fig. 5(b) suggests the best measurement window to maximize the difference between power profiles of various fan-ins. The PDF and CDF of current distribution are shown in Fig. 5(c) and (d), respectively. It can be noted that the current increases with fan-in. Unlike the OR array, both the STD and mean value graphs of the AND array, as shown in Fig. 5(e) and (f), respectively, increase monotonically with fan-in. Therefore, the PDF, CDF, STD, and mean value distributions collected from the test chips/simulations can be leveraged by the adversary to identify the fan-in of the gate from the recorded SCA current profile.

C. Attack Model 2

1) Leveraging the Precharge Power of AND Array: Adversary can identify gate fan-in by analyzing the current drawn during the precharge (pre-discharge) phase of AND (OR) array [see Fig. 6(a)]. The adversary forces all minterms to "0" by trying multiple patterns of inputs and analyzing OR array current. If OR array current is in the range of leakage current, the adversary concludes that as all the minterms are "0." Every active BL (i.e., BLs participating in the implemented function) is discharged to $V_{DD} - V_{th,Diode}$ if adversary forces all minterms to "0." This will lead to the maximum current drawn during AND array precharge phase. In the next precharge phase of AND operation cycle, DCIM has to charge all active BLs up to V_{DD} and based on the capacitor energy ($E = (1/2)CV^2$) adversary can find C which is the addition of all active BLs capacitance. The adversary will know the capacitance value on each BL by modeling BLs using accurate tools. Subsequently, adversary can find the number of inputs per minterm by the analysis proposed in Section III-B.

To find the precharge phase in the extracted power profile, adversary will examine a large peak without a short circuit. All large current peaks range from very large positive values to negative values due to switching of CMOS gates which consume short circuit (when both pull-up and pull-down networks are ON). This attack model works for all the functions implemented using DCIM.

2) Simulation Results: In this case, the adversary examines the power signature during the precharge phase instead of the computation window. The AND array's BLs are charged during the precharge phase. The power profiles observed differ based on the number of BLs. The adversary uses the distinct power measured during the precharge phase to determine the number BLs, which represents the number of minterms. The current profiles during the precharge phase of AND gates with various fan-ins are shown in Fig. 6(a). Fig. 6(b) suggests the best measurement window to maximize the difference between these current profiles. Additionally, the PDF and CDF distributions of precharge currents are shown in Fig. 6(c) and (d), respectively. It is noted that the precharge current increases

with fan-in. Both the STD and mean value graphs of the AND array [see Fig. 6(e) and (f)] increase monotonically with fan-in. Therefore, the PDF, CDF, STD, and mean value distributions collected from the test chips/simulations can be leveraged by the adversary to identify the fan-in of the gate from the recorded SCA current profile.

D. Impact of Supply Voltage Magnitude

We have swept the magnitude of supply voltage from 0.75 to 2 V in 50-mV increments (for DCIM) to analyze its impact on the effectiveness of SCARE. The result is summarized in Fig. 7(a). It is evident that mean value of current of the gates with different fan-ins increases at higher voltages. The differences in the mean values of currents are increased (higher slope) at higher voltages too, which helps the adversary to distinguish more accurately between different fan-ins. Sigma analysis of current profiles shows that current distributions are wider at higher voltages and sharper at lower voltages, which means sigma increases with voltage [see Fig. 7(b)]. As shown in Fig. 7(c), CDF slope decreases at higher voltages indicating that sigma increases at higher voltages. Note that, under PV, cases which overlap into adjacent fan-ins' distribution are important and hard to distinguish. SCARE calculates the overlaps between gates with fan-ins n and $n+1$ [see Fig. 7(d)] to gain insight on the overlap percentage at various different voltage nodes. Fig. 7(d) shows that the overlap is at its lowest for voltages near the nominal V_{DD} and it increases when the supply voltage magnitude increases or decreases. The worst case is for very low-supply voltages when the current magnitude is very small and a slight variation in the resistance values of RRAMs can lead to ambiguity of the gates' fan-ins.

E. Finding the Function Implemented on Memory

We can match the measured current profile against the precalculated current profiles to reveal the structure of the implemented function. A reduced RE effort will be needed to find the exact function. For example, 12.5% reduced RE effort is needed to find implementation of a 4-b adder where adversary should try AND_2 patterns which will involve 224 possible patterns (out of $2^8 = 256$ choices for eight total inputs for two numbers). Since adversary knows that function has AND_2 gate from SCARE, he/she will choose the first input out of 16 (including the complementary inputs). After this, he/she will choose the second input out of 14 (since the first choice and its complimentary are eliminated). Therefore, total number of choices equal to 224 (i.e., 16×14). It is noted that this reduction in the number of pattern is possible since adversary is aware of AND_2 gates using SCARE attack. The amount of RE effort reduction for adders of various widths are shown in Table III. The RE effort reduction increases for wider adders. However, the current profiles might overlap for wider adders due to the process voltage and temperature (PVT) variations leading to RE of incorrect structure. Other statistic analysis (mean value and STD) can be used to find the correct structure.

TABLE III
RE EFFORT REDUCTION OF ADDERS USING SCARE

Adder size	Original # of patterns	SCARE patterns	RE reduction
4b	256	224	12.5%
8b	65536	960	98.53%
16b	4294967296	3968	99.99%

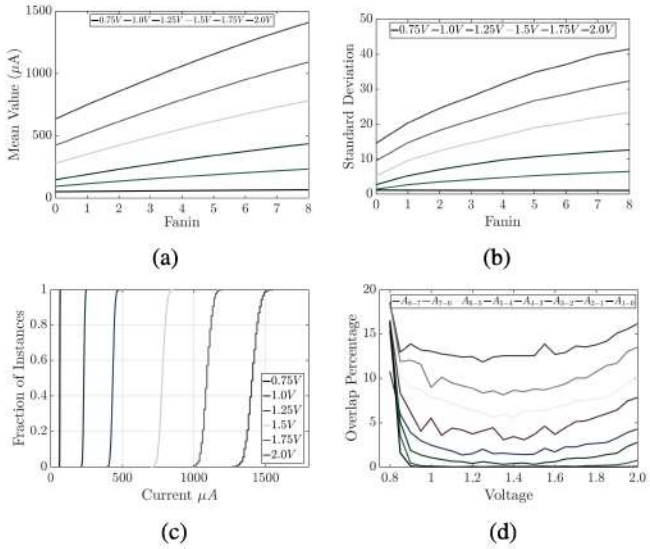


Fig. 7. Analysis of current mean value, STD deviations, percentage of distribution overlaps in PDF, and AND_8 CDF under different voltage nodes, (a) mean values; (b) STD; (c) percentage of overlaps; and (d) CDF of AND_8.

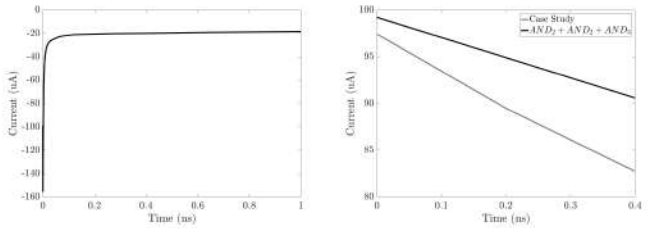
F. Case Study

1) Attack Model 1: We have implemented $a.c + a.b.d + d.c$ on DCIM. First we set all the inputs to logical "1" and measure the power profile of second cycle after EN signal is fired. Power profile matches the power profile of OR_3 in Fig. 4(a) which reveals that the implemented function has three minterms. Next, we set all inputs to logical "0" and ensure that all the BLs are discharging at the highest possible rate. Then, we measure the power profile and match the current drawn with a combination of 3 (number of minterms) of extracted power profiles. The closest combination to the extracted power profile is $AND_2 + AND_2 + AND_3$ (see Fig. 8). Now we have the structure of the function. Various input combinations are needed to find the implemented function. Note, we did not consider any combination of AND gates with four inputs or higher because the power consumption of AND_4 alone is higher than the power consumption of extracted three BLs combination.

2) Attack Model 2: We implemented $\overline{a}.\overline{b}.\overline{c}.\overline{d} + a.b + c.d$ using DCIM. Due to the presence of complementary inputs in the function, setting all inputs to logical "1" does not set all the minterms to "1." Therefore, the next option is to set all the minterms (BLs) to "0" to discharge all the AND array BLs after the EN signal is activated. To achieve this, we start with an arbitrary input pattern (say, 4'b0000). We change inputs one at a time and measure/compare the power profile. If power increases it means that the changed input has been able to set one minterm (BL) to "0." By performing the same action

TABLE IV
POWER CONSUMPTION TO SET ALL MINTERMS TO "0"

Input (dcba)	Minterms	Avg. power consumption
0000	1,0,0	220 u
0001	0,0,0	270 u
0011	0,1,0	220 u
0101	0,0,0	270 u
1101	0,0,1	220 u



(a) (b)

Fig. 8. (a) Current drawn in second cycle which matches the current profile of OR_3; (b) current drawn in first cycle which is very close to the power profile of $AND_2 + AND_2 + AND_3$.

for all the inputs we can find one or multiple patterns that sets all the minterms "0." However, sometimes changing an input may be neural, i.e., it may not change the power profile. This may occur if the change does not affect any minterm or it resets one minterm ("0") and sets another minterm ("1"). If an input is neutral, after setting other inputs we can go back to that particular input and recheck. In the worst case, at each iteration we can set only one input and the complexity of algorithm will be $O(N^2)$ (since we need to iterate over N inputs N-times) which is lower compared to $O(2^N)$ (i.e., brute-force complexity).

In this particular case, our starting pattern is "0000" for which we measure the power during the precharge phase (see Table IV). When we change "a" from "0" to "1," power consumption increases so we decide to set "a" to "0." Next, when we change "b" from "0" to "1" power consumption decreases which means one of the BLs has been set to "1" from "*0*." This in turn indicates that "b" should be kept "0." When we change "c" from "0" to "1," the power profile stays the same which means that "c" can be a neutral input. We proceed to "d." When we change it from "0" to "1," the power consumption reduces which means "d" should stay to "0." Finally, we go back to "c" again. Changing "c" does not change the power consumption. Now that we have an input pattern that sets all the minterms to "0," we can find the number of minterms by matching this power profile with extracted the power profiles (here it matches with power profile of three BLs). Next, we can measure the power profile during the computation phase (as same as attack model 1) to find the structure of the implemented function. Finally, we can find the implemented function by trying the possible patterns.

IV. ATTACK ON MAGIC ARCHITECTURE

An adversary can distinguish between the power drawn by the OR and AND array of MAGIC by examining the operation time determined using the spike in the power signature that is created during operation. The difference between OR and AND operation times ranges from 10× to 100× (see Fig. 9). Two attack models are proposed to find the structure of the implemented functions on MAGIC. Attack model 1 works if the inputs of the implemented function are only true inputs (a, b, c, etc.). Attack model 2 works on any implemented function however it needs more RE effort.

A. Attack Model 1

1) Leveraging the OR *and* AND *Array Power Signature/Operation Time:* Unlike DCIM, MAGIC's computation time depends on the type of gate and the fan-in. An adversary can RE the MAGIC functions using the computation time extracted from the power profile. MAGIC writes the result of a computation into a designated output RRAM by altering its resistance (HRS → "0" and LRS → "1"). We observe a significant change in the power profile when the resistance of the output RRAM changes. This sharp change in the power profile (during writing the output to RRAM) signifies the end of one MAGIC operation (e.g., 3-input AND operation). It is noted that the adversary can find the computation times for different gates and inputs by implementing known functions in MAGIC test chips and/or simulations and recording their power profiles.

Alternatively, the adversary can observe the constant current passing through output RRAM. In this approach, each of the input literals is set to a logical "0" (MAGIC initializes all the input RRAMs to HRS, the output RRAMs of OR and AND gates to HRS, and the output RRAMs of NOR gate to LRS). By measuring the V_{DD} current (minus the leakage current), the adversary can determine the current passing through the output RRAM. This allows the adversary to determine the gate implemented in a particular clock cycle (e.g., differentiate between the AND, NOR, and OR gates). The observed constant current (I) and the fan-in value (n) can be attributed to each gate (AND, OR, and NOR) based on the following rules (considering a function with eight inputs):

NOR:

$$\frac{V_{\text{write}}}{R_{\text{LRS}} + \frac{R_{\text{LRS}}}{\text{Max}_{\text{in}}}} \geq I \geq \frac{V_{\text{write}}}{0.5R_{\text{HRS}}}, \quad n = \frac{R_{\text{HRS}} \times I}{V_{\text{write}}}.$$

OR:

$$\frac{V_{\text{write}}}{R_{\text{HRS}}} \geq I \geq \frac{V\text{write}}{1.5R_{\text{HRS}}}, \quad \frac{n}{n+1} = \frac{R_{\text{HRS}} \times I}{V_{\text{write}}}.$$

AND:

$$\frac{V_{\text{write}}}{R_{\text{HRS}}} > I > \frac{V_{\text{write}}}{(\text{Max}_{\text{in}} + 1)R_{\text{HRS}}}, \quad n = \frac{V_{\text{write}}}{R_{\text{HRS}} \times I} - 1.$$

It is noted that due to the PVs, n might not be an integer and should be used to approximate to the nearest positive whole number (since fan-in should be an integer). The proposed attack model works only with true inputs. If the function consists of complementary inputs, the attack will fail since the adversary cannot confirm if all the minterms are set to "1" by forcing all inputs to "1."

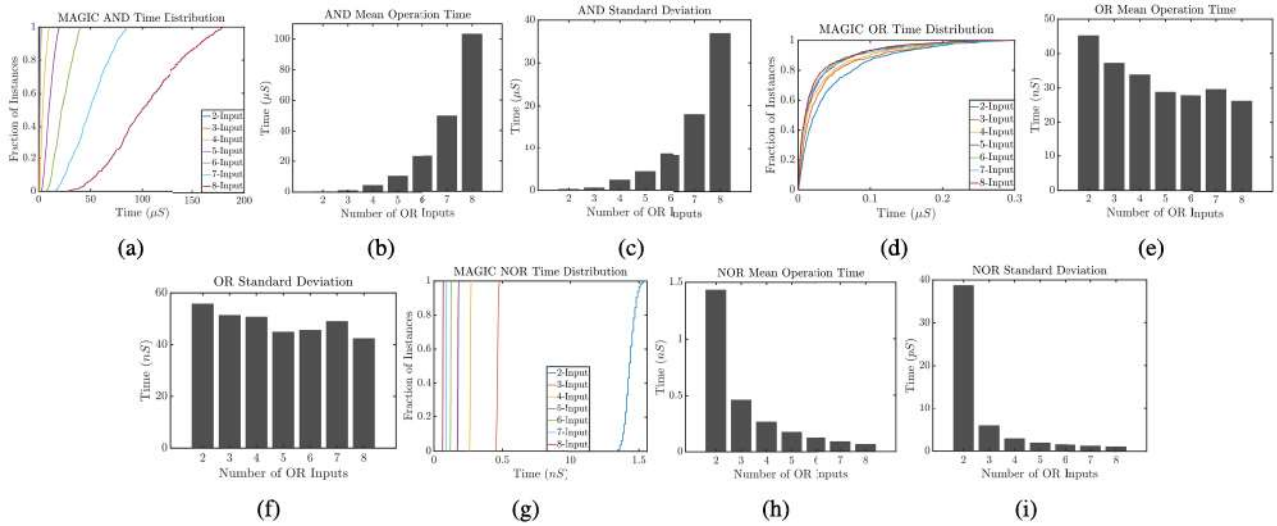


Fig. 9. Analysis of current profiles with different MAGIC fan-ins under PV to obtain CDF distribution of operation times, mean operation times, and STD of operation times for NOR gates [(a), (b), and (c)], OR gates [(d), (e), and (f)], and AND gates [(g), (h), and (i)], respectively.

2) Simulation Results: To evaluate the above-mentioned attack model for MAGIC gates, IMC computations are performed for AND, OR, and NOR gates. It is noted that MAGIC simulations for NAND are not performed since it requires each of the input RRAMs to be initialized to HRS. Therefore, the output RRAM does not get enough voltage headroom to get written since the input RRAMs (in HRS) consume a high voltage across them. Further study is required to ensure the validity of the MAGIC NAND design. For each of the remaining gates, an increasing (AND gate) or decreasing (OR and NOR gates) trend of computation time is observed when the fan-in of the logic array increases from 2 to 8. The computation completion is determined by a sharp change in the current profile during the switching of the output RRAM.

a) MAGIC AND *array:* The distribution of AND operation computation times for various fan-in is shown in Fig. 9(a). The computation times for each of the cases is represented by a distinct distribution. Furthermore, the mean and STD of the distributions [see Fig. 9(b) and (c)] show a monotonically increasing trend as the fan-in increases. Each of these graphs can be used to accurately determine the fan-in of the AND gates.

b) MAGIC OR *array:* The distribution of OR operation computation times for various fan-in is shown in Fig. 9(d). Unlike MAGIC AND, the computation time distribution for each of the fan-in overlaps. Therefore, the PDF alone cannot be reliably used by an adversary to determine the OR gate fan-in. It is noted that the mean and STD of each of the distributions [see Fig. 9(e) and (f)] show a non-monotonic decreasing trend with the fan-in. We note that the MAGIC OR implementation is comparatively resilient against SCA compared to AND and NOR.

c) MAGIC NOR *array:* The distribution of computation times for the NOR operation with various fan-in is shown in Fig. 9(g). Similar to AND, the completion times for each case are represented by distinct CDFs. The mean and STD of each of the distributions [see Fig. 9(h) and (i)] show a distinctly decreasing trend with increase in fan-in. Each of these graphs can be reliably used to accurately determine the fan-in of NOR gate.

B. Attack Model 2

1) Leveraging Pre-Compute RRAM Write Operation Times of AND *or* OR *Arrays:* Since inputs are stored as resistance values in RRAMs, their write operations are asymmetric and the adversary can find the values which are stored in the RRAMs by examining the power profile. The adversary can force each of the inputs to logical "1" (LRS) and examine the RRAM write currents. Based on the write-current observed, the adversary can determine the number of RRAM cells switched to "0" (HRS) and the number of cells that remain at "1" (LRS). IMC of any function through MAGIC occurs only after the RRAM cells corresponding to inputs and output are initialized to HRS or LRS values depending on the function and the array operation (i.e., NOR, OR, etc). Assuming a representative example function with eight input literals, the MAGIC architecture will employ eight input RRAMs and one output RRAM. Furthermore, each of these are preset to "0" (HRS) state. To execute a particular function, some or all of these nine RRAMs resistances are switched to "1" (LRS state). It is noted that the power consumed for switching different numbers of RRAMs (0 to 9 in this case) is distinct and can be extracted through SCA.

In addition to the case-by-case method explained here, the adversary can determine the number of HRS RRAMs (n) using the following equations:

NOR:

$$\frac{V_{\text{write}}}{R_{\text{LRS}} + \frac{R_{\text{LRS}}}{\text{Max}_{\text{in}}}} \geq I \geq \frac{V_{\text{write}}}{0.5R_{\text{HRS}}}, \quad n = \frac{R_{\text{HRS}} \times I}{V_{\text{write}}}.$$

OR:

$$\frac{V_{\text{write}}}{R_{\text{HRS}}} \geq I \geq \frac{V\text{write}}{1.5R_{\text{HRS}}}, \quad \frac{n}{n+1} = \frac{R_{\text{HRS}} \times I}{V_{\text{write}}}.$$

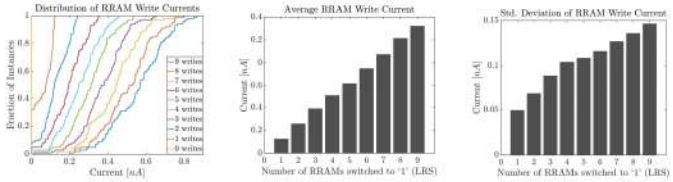


Fig. 10. Simulation results showing (a) write current distribution; (b) average write current; and (c) STD of the distribution for switching 1 to 9 RRAM cells (8 input + 1 output) from "0" to "1."

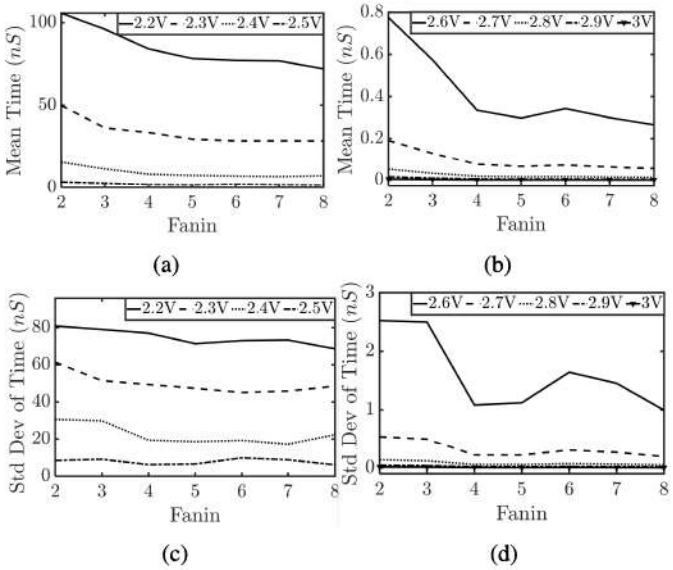


Fig. 11. Analysis of operation time mean values, STD deviations, and OR CDF under different voltage nodes, (a) and (b) mean values, and (c) (d) STD deviations of OR8.

AND:

$$\frac{V_{\text{write}}}{R_{\text{HRS}}} \geq I \geq \frac{V_{\text{write}}}{(\text{Max}_{\text{in}}+1)R_{\text{HRS}}}, \quad n = \frac{V_{\text{write}}}{R_{\text{HRS}} \times I} - 1.$$

2) Simulation Results: Compared to previous attack model, this model requires an added step (i.e., exploring RRAM initialization) to reveal the complementary inputs. A 100-point MC analysis is performed on RRAM write current with the setting shown in Table II. The resulting current distribution, average current, and STD of the distribution are shown in Fig. 10(a)–(c), respectively. It is evident that the number of RRAMs initialized to "1" and "0" can be found. To determine the inputs whose complementary values are used, each input is flipped from its original value ("1") one at a time. If a change in an input value (1→ 0) leads to an increase in the number of "0"s (HRS RRAMs), which is determined by re-examining the power signature, we can deduce that the original value of the input is used in the function. Alternatively, if the number of "0"s decreases, the input's complementary value is used in the function. In this way, the adversary can extract the structure of the function with true and complementary inputs.

C. Impact of Supply Voltage Magnitude

In case of MAGIC, the V_{DD} value is swept from 2.2 to 3 V in 100-mV increments. As shown in Fig. 11(a) and (b), the mean operation time decreases as the V_{DD} value increases. Furthermore, it is also noted that the STD value decreases under all V_{DD}s as the fan-in value increases. Fig. 11(c) and (d) show that the STD of the operation also decreases with the increase in V_{DD} and shows a mostly negative trend with the change in fan-in. Fig. 12 shows that the slope increases as the V_{DD} value increases and shows a decrease in sigma value. The slope of $V_{\text{DD}} > 2.6$ V is extremely high and is therefore not shown in Fig. 12 since they overlap with the CDF at $V_{\text{DD}} = 2.6$ V.

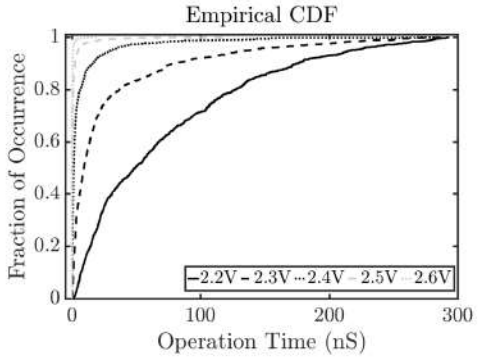


Fig. 12. Operation time CDF of MAGIC OR8 with voltage.

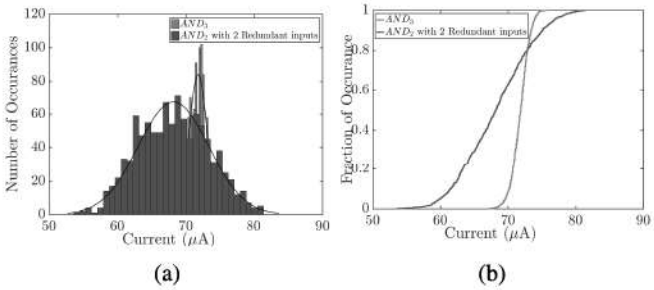


Fig. 13. Analysis of redundant inputs as countermeasure, (a) PDF and (b) CDF power profile of AND2 with two redundant inputs and AND3 without any redundant inputs. There is an overlap between them which obfuscate the power profile.

V. Countermeasures

A. Redundant Inputs

1) Dynamic Computing in Memory (DCIM): Few redundant LRS RRAMs on each BL can be implemented, which are biased with a fixed voltage. For instance, function $a + bc$ can be implemented as: $a.1.1.1.1 + b.c.1.1.1 + 0 + 0 + 0 + 0 + 0$. In this method, the area increases and the overhead is based on the number of maximum redundant inputs and number of inputs (e.g., four redundant inputs for a function with eight inputs increase the area by 0.25%). As long as the number of redundant LRS RRAMs in each BL is less than 8, the SM stays relatively constant. Based on our simulations, eight redundant inputs are enough to mask an array with 64 inputs increasing the area by 6%. The number of LRS RRAMs on each BL can be randomly distributed to further obfuscate the structure of the implemented function. Power overhead is completely dependant on the number of redundant inputs in

each BL. An example of masking AND_2 with two redundant inputs is shown in Fig. 13. It can be noted that the AND_2 power profile with two redundant inputs completely overlaps with AND_3 power profile with no redundant inputs. Therefore, the power profile signature is obfuscated. In this example, the power overhead is 21%.

Since the selector diode turns off when the voltage across its two terminals is less than V_{th}, the AND (OR) array's redundant inputs should not be driven by V_{DD} ("0") instead they should be driven by $(V_{\text{DD}}/3)$ for AND array and $(2V_{\text{DD}}/3)$ for OR array for better obfuscation.

2) Memristor-Aided Logic (MAGIC): In the case of MAGIC, redundant inputs increases the fan-in, and thus increases the number of input RRAM bitcells. As previously shown in Section IV, increasing the number of inputs even by "1" literal has a distinguishable change in the operation completion time (see Fig. 9). Therefore, it can be leveraged to mask the true structure of any MAGIC implementation for any of the operations.

B. Minterms With Expanded Literals

1) Dynamic Computing in Memory (DCIM): Each minterm in a function can be implemented by the maximum number of inputs. In this scenario, all minterms show the same power profile and SCA alone fails. However, an adversary can still try all the possible input patterns and generate input–output pairs. Next, it can be used to determine the function using a Karnaugh map to reveal the simplified Boolean expression.

For example, $a + bc$ can be implemented in a four input system as, $ab\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + ab\bar{c}d + a\bar{b}\bar{c}d + abcd + abc\bar{d} + a\bar{b}cd + a\bar{b}c\bar{d} + \bar{a}bcd + \bar{a}bc\bar{d}$. Furthermore, it will become complicated for the adversary to find the function when $a+bc$ and $ab + ac + ad + bd + cd$ have the same number of minterms in the expanded version and when $a + bc$ has more minterms than $ab\bar{c} + \bar{a}bc + \bar{a}\bar{b}\bar{c}$. This technique can protect the IP at the cost of increased area and power overhead. An example is shown in Fig. 14 where the two functions consume the same power. Power and area overhead depend on the implemented function. For the example in Fig. 14, power consumption increases by 36% and AND array area stays the same (since the crossbar array with enough BLs to implement the minterms already exists). Power overhead depends on the number of original BLs and number of BLs with expanded literals. Simulations show that our array consumes 50 μW due to leakage power and activation power of the transistors and peripherals. Each BL with a $C_{\text{BL}} = 100$ fF consumes 70 μW for charging (discharging) in precharge (predischarge) phase. For the $a + b.c$ example, the power consumption changes from 190 to 750 μW. The power consumption changes from 400 to 750 μW for the $a.b + a.c + a.d + b.d + c.d$ function. It should be noted that this is the power consumption of precharge or predischarge phase and not the power consumption of the whole array.

2) Memristor-Aided Logic (MAGIC): We consider two representative example functions $a + bc$ and $abd + \bar{a}\bar{b}d + \bar{a}b\bar{d}$ for MAGIC as well. The first function requires a 2-fan-in AND and a 2-fan-in OR operation, while the second function requires

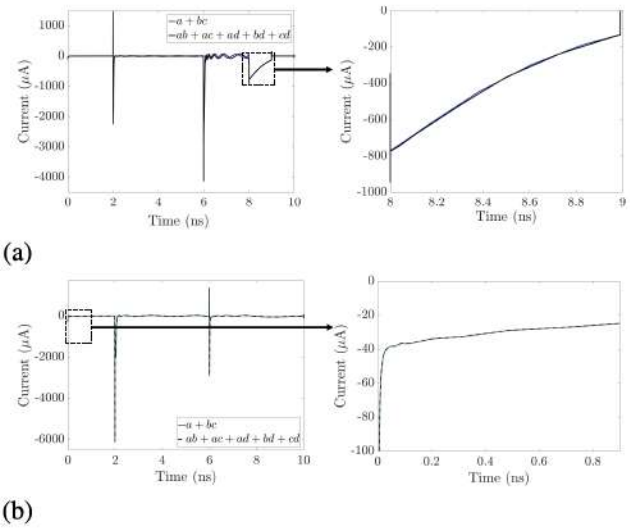


Fig. 14. Analysis of expanding the literals as countermeasure. Power profile of, (a) AND during precharge and (b) OR during computation for computing $a + bc$ and the expanded version of it. They overlap completely with each other.

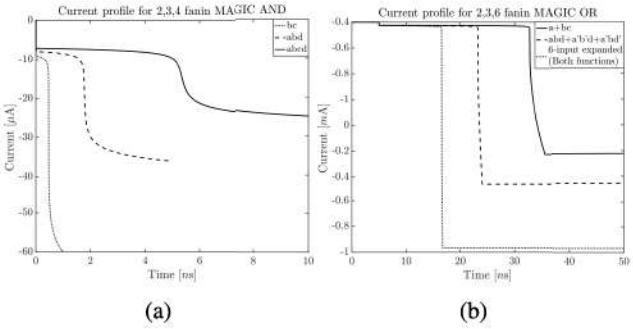


Fig. 15. Simulation results that show masking of original function structure using maximized SOP form for two functions, (a) 2, 3, and 4-fan-in MAGIC AND current profiles; (b) 2, 3, and 6-fan-in MAGIC OR current profiles.

three 3-fan-in ANDs and one 3-fan-in OR operation. Expanding these functions into their maximized SOP form will require six 6-fan-in ANDs and one 6-fan-in OR operation for both of the functions. Since these operations are identical, SCA delivers the same result and masks the true structure of the operation. Fig. 15(a) shows that 2-fan-in and 3-fan-in ANDs have distinctly different operation completion time as depicted by a sharp change in their current profiles. The 4-fan-in AND for both operation in their maximized SOP form is shown to be identical. Similarly, Fig. 15(b) shows the distinctly different 2-fan-in and 3-fan-in current profiles of each function's OR operation and depicts the identical 6-fan-in OR current profiles for their maximized form. This attack model will not incur any area overhead since the maximized SOP form will only leverage previously presented RRAM cells in the crossbar array for any additional literals. However, it will incur some power overhead due to the increase in the number of SOP minterms to be computed.

The above-mentioned countermeasures increase the RE effort. For example, $ab + cde + fgh$ without any countermeasures would require 84 inputs to determine the function

structure. Implementing the countermeasures increases the required number of combinations to 256 (RE effort increases by 3.04×). RE effort increases exponentially with the number of inputs.

VI. Discussions

A. Extracting the Exact Function for MAGIC

In the absence of complementary inputs, adversary can find number of implemented minterms in each cycle by setting all the inputs to "1" ("0") for NAND (NOR) design and measuring the current of large spikes in the power profile. After finding number of minterms, the adversary has to measure the power profile of first cycle of operation to find the number of inputs of each minterm. When the structure of implemented function is revealed, the adversary has to try the possible test patterns and analyze the output to find the implemented function. However, when there are complementary inputs adversary has to use the method explained in Section III-E to set all the minterms to "0" ("1") for NAND (NOR) design. Once the adversary has access to the number of minterms, he/she can find fan-in of each of the minterm. Again, when the structure is revealed adversary can try possible patterns and analyze the output to find the implemented function.

B. Number of Test Chips Needed for an Attack

We performed 1000-point MC simulations to develop SCARE's power model for DCIM/MAGIC. In the absence of models, adversary can fabricate few chips to launch the attack with minor loss in accuracy. The mean value of worst case margin (e.g., margin between AND7/AND8 for DCIM and OR7/OR8 for MAGIC) is degraded by 3.5%, 3.2%, 1.8% (for DCIM) and 5.2%, 4.77%, and 2.8% (for MAGIC) for 25, 50, and 100 chips, respectively, compared to 1000-point Monte-Carlo. Furthermore, standard deviation increases by 18.5%, 15.9%, and 4.9% for DCIM and 21.3%, 9%, and 7.9% for MAGIC. Adversary can minimize measurement noise by taking multiple samples of the device under attack and averaging them.

C. Realistic Attack and Parallelism

IMC architectures include MAGIC, DCIM, and matrix-vector multipliers (MVMs) [29]. DCIM and MAGIC can implement arbitrary functions while MVM can only implement dot product. Under parallel operations/functions, the adversary can find the number of functions by observing the number of outputs in DCIM/MAGIC. In DCIM, number of outputs = number of NOR gates. Power in second cycle yields NOR gate fan-ins (e.g., two OR_2, one OR_3). Power in cycle-1 yields the number of AND gates/fan-ins. After determining the total number of AND/OR gates, adversary can run a limited number of input patterns to relate the input bits to the corresponding observed output bit. For example, if a function has two AND_2 and another function has a AND_1 and a AND_3, SCARE can identify that the structure is a AND_1, two AND_2, and a AND_3. Next, the adversary has to try the possible patterns of inputs and relate each of the minterms to an output. In this example, adversary can apply patterns specific to AND_1, AND_2, and AND_3 and if any of the outputs are switched from "0" to "1" adversary can relate the test input (and the corresponding minterm) to that specific output. Once each minterm is related to its respective output, the functions are revealed. Compared to brute-force, SCARE reduces the number of test patterns to RE functionality, e.g., 81.25% less patterns to identify 4-b adder (see Table III).

In MAGIC, designers need multiple arrays to implement functions in parallel where power peaks might overlap. The number of functions can be found from the magnitude of power (e.g., for writing 1- versus 2-output RRAMs). While multiple array operations can cause overlapping power spikes due to more than one AND/OR, the adversary can determine the individual gates by dividing the total power as a summation of individual powers as modeled before. Timing differences between the completions of gate operation can also be exploited.

D. Applicability of Existing SCA Obfuscation Techniques

SCA obfuscation techniques [30] proposes to inject random code execution to scramble the power profile and prevent SCA on cryptographic implementations. Such protection techniques, if extended to IMC architectures, will impose significant throughput overhead since random functions between the actual ones will incur extra delay. This is in addition to area and power overheads. Duplicating logic with complementary operations [31] can eliminate the asymmetry between power drawn to process 0 and 1. This technique will not protect IMC against SCA since the function and its complement may have different number of minterms. Therefore, they may consume different amount of power.

E. Scalability of the Attack Models

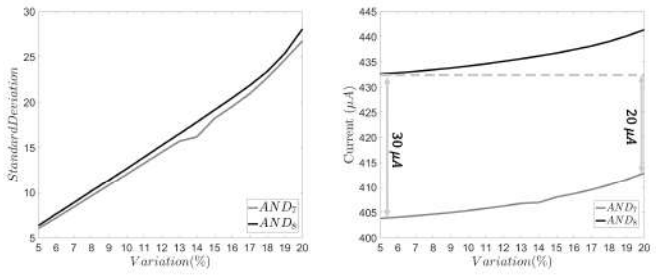
Our simulations show that the PDF of the distributions overlap significantly when the number of inputs is greater than 10. Therefore, other statistical analysis, such as mean value and STD can help to find number of inputs.

F. SCARE End-Applications and IP Targets

Any function that can be implemented by DCIM and MAGIC can be targeted by SCARE. Authors in [32] have implemented SHA-3 encryption based on multiple levels of DCIM using AND and OR arrays which can be attacked by SCARE.

G. Maximum Allowed Variations

We ran MC simulations by sweeping the amount of variations from 5% to 20% to find the maximum allowed variations for which distributions can be distinguished. We validated AND_7 and AND_8 since they have the highest overlap between PDF of precharge current. Fig. 16(a) shows that the STD of the current increases with the amount of variations. Since adversary does not know the amount of variations, analyzing STD does not help to find the implemented function (e.g., STD of AND_7 with 6% variations is almost equal to the STD of



(a) (b)

Fig. 16. (a) Standard deviation of current profiles of AND_7 and AND_8; (b) mean value of current profiles of AND_7 and AND_8.

AND_8 with 5% variations). However, as shown in Fig. 16(b), the mean value of current profiles are highly distinct and even mean value of AND_7 with 20% variations is not close to the mean value of AND_8 with 5% variations. So, SCARE can reduce the cost of RE without even with high values of variations.

VII. Conclusion

This article proposes SCARE, a noninvasive RE on IMC using SCA for the first time. SCARE is applied to two well-known emerging technology-based IMC architectures (DCIM and MAGIC). We also present possible countermeasures to mitigate SCARE attack.

References

[1] A. Agrawal, A. Jaiswal, C. Lee, and K. Roy, "X-SRAM: Enabling in-memory Boolean computations in CMOS static random access memories," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4219–4232, Dec. 2018.

[2] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic RAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 470–483, Dec. 2017.

[3] K. Korgaonkar, R. Ronen, A. Chattopadhyay, and S. Kvatinsky, "The Bitlet model: Defining a litmus test for the bitwise processing-in-memory paradigm," 2019, *arXiv:1910.10234*. [Online]. Available: http://arxiv.org/abs/1910.10234

[4] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, Apr. 2017.

[5] S. Motaman and S. Ghosh, "Dynamic computing in memory (DCIM) in resistive crossbar arrays," in *Proc. IEEE 36th Int. Conf. Comput. Design*, Oct. 2018, pp. 179–186.

[6] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2014, pp. 8326–8330.

[7] W. Kang, H. Wang, Z. Wang, Y. Zhang, and W. Zhao, "In-memory processing paradigm for bitwise logic operations in STT-MRAM," *IEEE Trans. Magn.*, vol. 53, no. 11, pp. 1–4, Nov. 2017.

[8] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic design within memristive memories using memristor-aided loGIC (MAGIC)," *IEEE Trans. Nanotechnol.*, vol. 15, no. 4, pp. 635–650, Jul. 2016.

[9] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proc. ACM/EDAC/IEEE Design Automat. Conf.*, Jun. 2016, pp. 1–6.

[10] X. Yin, M. Niemier, and X. S. Hu, "Design and benchmarking of ferroelectric FET based TCAM," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Mar. 2017, pp. 1444–1449.

[11] M. Imani, Y. Kim, and T. Rosing, "MPIM: Multi-purpose in-memory processing using configurable resistive memory," in *Proc. 22nd Asia South Pacific Design Automat. Conf.*, Jan. 2017, pp. 757–763.

[12] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, Jun. 2016, pp. 27–39.

[13] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, Jun. 2015, pp. 336–348.

[14] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, Jun. 2015, pp. 105–117.

[15] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2017, pp. 273–287.

[16] A. Haj-Ali, R. Ben-Hur, N. Wald, R. Ronen, and S. Kvatinsky, "Imaging-in-memory algorithms for image processing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4258–4271, Dec. 2018.

[17] S. Kvatinsky *et al.*, "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[18] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 10, pp. 2054–2066, Oct. 2014.

[19] T. Sugawara, D. Suzuki, M. Saeki, M. Shiozaki, and T. Fujino, "On measurable side-channel leaks inside ASIC design primitives," *J. Cryptograph. Eng.*, vol. 4, no. 1, pp. 59–73, Apr. 2014.

[20] M. N. I. Khan, S. Bhasin, A. Yuan, A. Chattopadhyay, and S. Ghosh, "Side-channel attack on STTRAM based cache for cryptographic application," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 33–40.

[21] H.-S. P. Wong *et al.*, "Metal-oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[22] D. J. Bernstein. (2005). *Cache-Timing Attacks on AES*. [Online]. Available: http://cr.yp.to/papers.html

[23] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems*. Berlin, Germany: Springer-Verlag, 2001.

[24] M. N. I. Khan and S. Ghosh, "Information leakage attacks on emerging non-volatile memory and countermeasures," in *Proc. Int. Symp. Low Power Electron. Design*, Jul. 2018, pp. 1–6.

[25] R. Giterman, M. Vicentowski, I. Levi, Y. Weizman, O. Keren, and A. Fish, "Leakage power attack-resilient symmetrical 8T SRAM cell," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 10, pp. 2180–2184, Oct. 2018.

[26] *Predictive Technology Model*. Accessed: Aug. 2020. [Online]. Available: http://ptm.asu.edu/

[27] *Arizona State University RRAM Model*. Accessed: Aug. 2020. [Online]. Available: http://nimo.asu.edu/memory/

[28] J.-J. Huang, Y.-M. Tseng, W.-C. Luo, C.-W. Hsu, and T.-H. Hou, "One selector-one resistor (1S1R) crossbar array for high-density flexible memory applications," in *IEDM Tech. Dig.*, Dec. 2011, pp. 31.7.1–31.7.4.

[29] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53rd Annu. Design Automat. Conf.*, Jun. 2016, pp. 1–6.

[30] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "A smart random code injection to mask power analysis based side channel attacks," in *Proc. 5th IEEE/ACM Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, New York, NY, USA, Sep. 2007, pp. 51–56.

[31] R. M. Avanzi, C. Heuberger, and H. Prodinger, "Minimality of the Hamming weight of the τ-NAF for Koblitz curves and improved combination with point halving," in *Selected Areas in Cryptography*. Berlin, Germany: Springer, 2006, pp. 332–344.

[32] K. Nagarajan, S. S. Ensan, M. N. I. Khan, S. Ghosh, and A. Chattopadhyay, "SHINE: A novel SHA-3 implementation using ReRAM-based in-memory computing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Jul. 2019, pp. 1–6.