Side Channel and Fault Analyses on Memristor-Based Logic In-Memory

Pietro Inglese, Ioana Vatajelu, Giorgio Di Natale

Side Channel and Fault Analyses on Memristor-Based Logic In-Memory

Pietro Inglese, Elena-Ioana Vatajelu, Giorgio Di Natale
Univ. Grenoble Alpes, CNRS, Grenoble INP*, TIMA, 38000 Grenoble, France
{pietro.inglese,ioana.vatajelu,giorgio.di-natale}@univ-grenoble-alpes.fr

***Abstract*— In-memory computing is a promising approach to address the challenges faced by traditional computing architectures, such as the memory wall and the energy consumption of data transfer. By storing and processing data entirely in main memory, in-memory computing can offer significant improvements in performance and scalability. This solution might also bring more security, thanks to the limited data movement which mitigates the risk of information leakage via the communication buses. In this paper, we present the results of side-channel and fault analyses on one of the most researched solutions for logic-in-memory based on memristive memory arrays. Our results show that both analyses can easily reveal secret information.**

Keywords — logic-in-memory, in-memory computing, memristors, side channel analysis, fault analysis

I. INTRODUCTION

With CMOS technology being close to its physical limits, making it harder to scale down the size and improve the performances, we are witnessing the end of Moore law. Furthermore, today's common computer architectures are facing increasing issues, such as the memory wall/Von Neumann bottleneck, characterized by a high energy consumption due to the data moving between the memory and the processing unit.

Emerging non-volatile resistive memories such as Resistive RAM (RRAM), Spin-Transfer Torque Magnetic Random Access Memory (STT-MRAM), and Phase Change Memory (PCM) have In-Memory Computing capabilities, and they promise to solve these issues by increasing the computation speed, the parallelism and power efficiency. There is a very wide variety of In-Memory Computing (IMC) solutions that exploit existing technologies. They enable logic (also called Logic-In Memory, LIM) and/or arithmetic operations directly inside the memory boundaries. The operations are performed without the need of transferring data to/from the CPU, thus saving time and energy therefore mitigating the memory wall. IMC is made possible by exploiting the physical characteristics of the memory device and by inserting control and processing elements in the peripheral logic (e.g., in the write drivers and sense amplifiers), which enable the computation.

Besides the need for power efficiency and computation speed, the need for security has also becoming increasingly important. This has led to the development of hardware components and IPs for cryptography, but it has also created new types of threats and hardware attacks, such as: side-channel attacks, which exploit information leaked through a device's physical characteristics, such as power consumption or electromagnetic emissions; fault injection attacks, which aim to introduce faults into a device's hardware in order to disrupt its normal operation or extract sensitive information.

In classical architectures, data and cryptographic keys are stored in the main memory, and transferred to the processor to execute the cryptographic functions. Therefore, confidential information transits unencrypted via the communication buses, being susceptible to information leakage. In contrast, with IMC secure operations can be performed without resorting to data transfer, therefore mitigating the risk of data leakage and avoiding exposure to attacks. Among the many IMC solutions, this paper focuses on LIM based on MAGIC (Memristor-Aided Logic, [1]) which is able to perform any logic operation within the memory array.

Within this paper we demonstrate that:

- MAGIC-based operations have a power consumption profile which is data-dependent, thus enabling side-channel attacks
- memristive memory arrays are very sensitive to variations of electrical operation conditions, and thus prone to fault attacks

To the best of our knowledge, this is the first time where side-channel and fault analysis are performed on the MAGIC-based LIM implementation in the context of secure applications, even though side-channel analysis has been already used to reverse-engineer the functional structure of IPs implemented with MAGIC in a memristive array [2], whereas [3] performs Side Channel and Differential Power Analyses on another type of LIM implementation, i.e., the Complementary Resistive Switching (CRS).

The paper is organized as follows: Section II introduces the Logic-In-Memory paradigm and its main basic operations together with our simulation environment and the proposed case study; Section III presents the results of the Differential Power Analysis on the circuit under study, while Section IV reports the effects of electrical perturbations on its behavior, which can be exploited to perform fault attacks. Section V concludes the paper.

II. BACKGROUND

Performing logic operations within a memory array is only possible when the memory cells are dotted of specific physical characteristics and the peripheral logic is redesigned to allow for computation. There exist different techniques for enabling logic in memory operations, some that compute with array-stored inputs and yield array-stored outputs (such as MAGIC) and others that compute with array-stored inputs but the output is obtained as an electrical signal at the periphery (such as Scouting Logic [4]) or the input is presented as voltage signals (such as CRS [3]). In this work we are concerned with the former technique, because it eliminates completely the data movement outside of the memory array. However, this type of computation can only be performed on memristors, since only they possess the required physical characteristics.

A memristor is a type of electronic component that functions as a variable resistor and can be used as a non-volatile memory device. The memristor stores data in the

* *Institut National Polytechnique Grenoble Alpes*

form of resistance levels (its minimum resistance is the Low Resistive State-LRS and its maximum resistance is the High Resistive State-HRS, which can be used to represent the logical state of "1" and "0", respectively) and its resistance can vary in function of the electrical signals applied to it. Due to the variety of materials used in their fabrication, memristors can have different electric behaviors. They can be controlled in voltage or current (with different polarities for the LRS to HRS transition and the HRS to LRS transition, respectively) and their resistance can either exhibit a continuous transition between resistive states or remain unchanged until a certain voltage/current threshold is reached and only then transition between resistive states. The latter class of memristors is very useful for applications such as data storage and LIM because it allows for non-destructive read and logic operations when the control voltage (or current) is below the threshold.

There are several LIM solutions described in literature, such as MAGIC [1], FELIX [5], and IMPLY [6], which implement some basic logic functions: NOT and NOR for MAGIC; NAND and OR for FELIX and implication for IMPLY. The IMPLY solution has the disadvantage of being input-destructive (i.e., the inputs are not preserved after the operation is executed), making the reuse of data difficult. The FELIX is not a robust solution, as demonstrated in [7]. In contrast, MAGIC solution is robust and non-input destructive, therefore suitable for the implementation of complex logic. Moreover, MAGIC (i.e. NOT and NOR operations) is not sensitive to dynamic and static memristive variability (cycle-to-cycle and device-to-device), while it is sensitive to variations in control signals, as shown in [7].

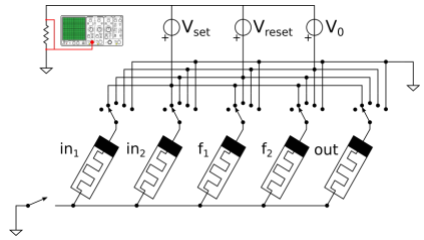
The MAGIC NOR operation requires three memristors: two in parallel as input values (in_1 and in_2) and the third, in series, for the output (*out*). The logic operations are carried out by first setting the output memristor to '1' (LRS), then providing a voltage V_0 to the resistive structure. The output memristor will switch to '0' (HRS) if its voltage drop is large enough, which depends on the input values. This switch happens when at least one of the input memristors is at '1', thus performing a NOR operation. The MAGIC NOT operates in the same way, but only with one input memristor.

The analysis we performed in this work is based on simulation. We used Cadence Spectre and the VTEAM (Voltage ThrEshold Adaptive Memristor) model to assess our hypothesis [8]. Parameters, duration and control voltage values for the MAGIC NOT and NOR have been selected according to [7]: R_{off} (HRS, logic 0): 300KΩ; R_{on} (LRS, logic 1): 1KΩ; a cycle time of 0.25ns; a control voltage V_0 of 1.4V.

The goal of this study is the analysis of the sensitiveness to side-channel and fault injections on MAGIC-based LIM implementations in the context of secure applications. The most significant logic operation for cryptography is the XOR operation (commonly used in all encryption algorithms, including DES, AES, and PRESENT), which secures the plaintext by combining it with the secret key. Therefore, we have chosen to focus our analysis on the XOR gate, as we believe that if this operation is not secure, the overall algorithm is not secure either.

In order to implement XOR with MAGIC, it is necessary to concatenate multiple NOR and NOT operations by obtaining a sequence of five steps (as fully described in [9]): 1) MAGIC NOT (in_1, f_1), 2) MAGIC NOT(in_2, out), 3) MAGIC NOR(f_1, out, f_2), 4) MAGIC NOR(in_1, in_2, f_1) and 5) MAGIC NOR(f_1, f_2, out). These steps involve the use of five memristors: in_1 and in_2 as input memristors, f_1 and f_2 as functional memristors used to store temporary results, and *out* as the output memristor where the final result of the operation is written. Figure 1a shows the schematic we implemented to perform our simulations. V_{set} and V_{reset} are used to initialize memristors to the desired logic value, and V_0 is used to perform the NOR and NOT logic operations.

(a)



(b)

Inputs	Step 1 f1=NOT(in1)	Step 2 out=NOT(in2)	Step 3 f2=NOR(out,f1)	Step 4 f1=NOR(in1,in2)	Step 5 out=NOR(f1,f2)
00	1	1	0	1	0
01	1	0	0	0	1
10	0	1	0	0	1
11	0	0	1	0	0

Figure 1. (a) Netlist for the MAGIC-based XOR. (b) Truth-table of MAGIC-based XOR

III. Side Channel Analysis

A. Background

Side-Channel Attacks exploit the fact that secure devices leak physical information during data processing. This physical leakage (e.g., power dissipation [10], electromagnetic emanation, timing information) can be measured externally and used for compromising confidential data, such as the secret key of a cryptographic system. Side-channel attacks such as Simple and Differential Power Analysis (SPA and DPA) have become popular since, without proper countermeasures, they require the knowledge of the algorithm, a model correlating the physical measurements and the processed data, but not the physical implementation of the target device.

On classical CMOS-based circuits, DPA exploits the fact that transitions (from 0 to 1 or from 1 to 0) of the logic gates require energy (that can be measured via an oscilloscope). On the other side, without transitions, the gate's transistors only have static power consumption. Therefore, by measuring the current consumed by the circuit, it is possible to create a correlation with internal circuit's transitions. The most common information leakage models are the hamming distance and hamming weight. The Hamming distance model assumes that the power consumption of a device is correlated with the number of bits that change between two input states. The Hamming weight model assumes that the power consumption of a device is correlated with the number of bits set to 1 in the input data.

On the contrary, in resistive-based circuits, we observe large variations in currents consumed by the circuit, with or without state transitions. In this paper, we investigate how side-channel analysis can be performed based on this principle.

B. *Current consumption of MAGIC-based XOR*

In order to create a proper information leakage model for the memristive-based LIM operations, we simulated the MAGIC-based XOR operation while measuring the corresponding current profile, for all input combinations, as shown in Fig. 2. It should be noticed that the XOR is obtained by concatenating the 5 operations (as shown in Fig. 1b) and each operation is performed in two steps, i.e., the SET of the output memristor, followed by the actual NOR (or NOT) operation involving the inputs. Some particularities of the current behavior should be noticed:

- A current spike is observed every time the output memristor changes its state from the preset value (i.e., switches from '1' to '0'). The positions of the current spikes reflect the truth tables in Fig. 1b. In addition, the spike amplitude is at least 3 orders of magnitude larger than what would be observed in a classical CMOS gate.
- The energy consumed during the SET operation depends on the initial state of the memristor. Indeed, if the initial state is LSR, the SET operation does not change the state of the memristor but it has high energy consumption. If the initial state is HRS, the SET operation changes the state of the memristor but its energy consumption its very low before the switch happens. In operations 1, 2 and 3, the initial states of memristors f_1, f_2, and *out* are assumed unknown (and in any case not related to the input values – for simplicity, in the simulation we assumed their initial state to HRS), while in operations 4 and 5, the initial states of memristors f_1 and *out* depend on the input values. Indeed, before the SET of f_1 in operation 4, the state of f_1 is the result of operation 1, i.e., not(in_1), while before the SET of *out* in operation 5, the state of *out* is the result of operation 2, i.e., not(in_2). For the case 0⊕0, the energy consumption during the last two operations is therefore very high, while for the case 1⊕1 is very low in comparison. For the cases 0⊕1 and 1⊕0, the energy has an intermediate value.

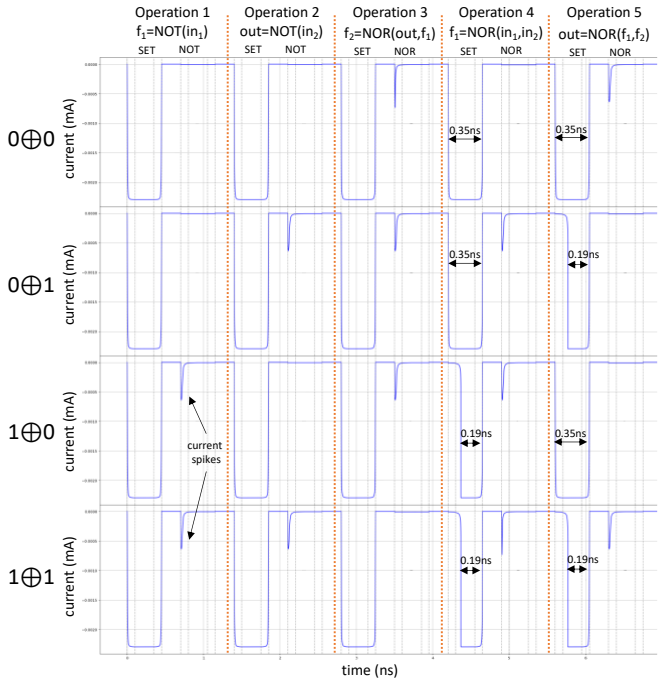


Figure 2. MAGIC-based XOR current curves

C. *DPA on MAGIC-based XOR*

In the execution of the DPA, the supply current measurements of a large number of encryptions are divided over two sets by means of a selection function based on the information leakage model (which is data-dependent) and a guess on the secret key. The difference between the averages of the two sets will approach zero for a wrong key guess, but has noticeable peaks if the correct secret key has been predicted.

In order to prove that the MAGIC-XOR operation can be attacked via DPA, we have created a circuit able to perform eight 2-bit XOR operations at the same time. We have fixed one of the two inputs (to emulate the presence of a secret key) and we have applied exhaustively all possible input combinations (i.e., 256). Based on observation of Fig. 2, we have created our selection function in such a way that 0⊕0 operations belong to the first set (the one contributing to the energy consumption), 1⊕1 belong to the second set, while 0⊕1 and 1⊕0 are ignored. We used the tool in [11] to perform the DPA. The result of the attack is shown in Fig. 3, where each line represents the DPA result for each key guess. The line of maximum amplitude corresponds to the correct key guess, thus showing the success of the attack.

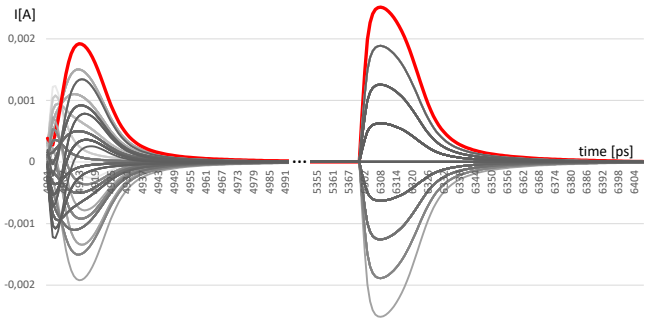


Figure 3. DPA result on eight 2-bit XOR operations. The red line corresponds to the correct key guess

This finding is noteworthy as we are dealing with a system that utilizes resistors with two vastly different resistance values, and the currents involved in the computation are on the order of mA when the resistance is low or in the order of µA when the resistance is high. Therefore, there is a significant correlation with the processed data, which can be exploited by side-channel analysis. Moreover, this outcome can be extended to any type of In-Memory Computing where the values of resistances (or currents) differ substantially between the two logical states.

IV. Fault Analysis

A. *Background*

Fault attacks [12] are a class of attacks that exploit weaknesses in a system by introducing controlled faults or errors in its operation. The goal of a fault attack is to cause the system to behave in an unintended way, reveal sensitive information, or break its security measures. Fault attacks can be performed by manipulating the physical environment in which the system operates, such as temperature, voltage, electromagnetic radiation, or clock signals.

Differential Fault Attacks (DFAs) and Safe Error Attacks (SEAs) are especially potent against cryptographic systems.

A DFA exploits the differences in the behavior of a system when operating normally versus under faulty conditions to obtain secret information. On the other hand, a SEA induces transient faults to cause a single-bit information leak, depending on whether the targeted algorithm produces an error or not.

Since the most significant logic operation for cryptography is the XOR, we show the effects of voltage manipulation on the behavior of the MAGIC-based XOR, to demonstrate the feasibility of DFA and SEA on MAGIC-based cryptography. As shown in Fig.1b, the MAGIC-based XOR operation is a concatenation of several NOR (and NOT) operations. If one of these is not performed correctly, the result of the XOR is not correct either.

B. Fault Analysis of MAGIC-based XOR

In this study, we assume an attacker is able to manipulate the voltage of the system to change the behavior of NOR and NOT operations, thus affecting the result of the XOR. Under these conditions, three scenarios are possible:

1. The attacker manipulates only the SET voltage (V_{SET}).
2. The attacker manipulates only the control voltage V_0.
3. The attacker manipulates the main power supply, thus affecting both the SET voltage (V_{SET}) and the control voltage V_0 in one or multiple cycles.

To understand the effect of this attack on the full XOR gate, we have first analyzed its effect on the basic NOR/NOT operations. The following effects have been observed:

1. If V_{SET} is below the threshold voltage of the SET operation, the memristor cannot be initialized at '1', which is the first step in performing any NOR/NOT operation. This will not affect the correctness of the operation if the memristor is already at '1'. However, if the memristor is at '0', the correctness of the operation depends on the input values, as shown in Table I, column "Low V_{SET}". The 'X' value is shown when the result of the operation depends on the initial state of the memristor, which might be unknown.
2. If V_0 is low, the voltage drop on the output memristor might not be enough to change its state from '1' to '0', when at least one of the inputs is at '1' (as per MAGIC-operation principle described in section II). This situation is illustrated in Table I, in the column "Low V_0"
3. If both VSET and V0 are low, the two effects described before are combined, and the output memristor is not able to change its initial state, as shown in Table I, column "Low V_{SET} and V_0".

MAGIC XOR is built as the concatenation of 5 NOT/NOR steps. We have considered several attack scenarios (AS), based on the instant and duration of the perturbation: (AS1) only the V_{SET} in one of the 5 steps; (AS2) only the V_0 in one of the 5 steps; (AS3) both V_{SET} and V_0 in one or multiple consecutive steps. These attacks can be realized with expensive means (AS1 and AS2, which require very precise time resolution, such as EM injection) or with more affordable means for the attack AS3 (voltage or power glitching). Table II shows the expected outputs of the XOR operation when affected by the aforementioned attacks.

	Inputs	Initial output state	Expected	Low V_{SET}	Low V_0	Low V_{SET} and V_0
NOT	0	0	1	0 / 1 X	1	0 / 1 X
		1				
	1	0	0	0 / 0 0	1	0 / 1 X
		1				
NOR	00	0	1	0 / 1 X	1	0 / 1 X
		1				
	01	0	0	0 / 0 0	1	0 / 1 X
		1				
	10	0	0	0 / 0 0	1	0 / 1 X
		1				
	11	0	0	0 / 0 0	1	0 / 1 X
		1				

Table I. MAGIC NOT and NOR behavior for control voltages affected by external perturbations

In Table II, cells are filled with different colors, based on the exploitability of the attack:

- Green cell: the attack has no effect on the result of the operation;
- Yellow cells: the effect of the attack depends on the initial state of the memristors f_1, f_2 and *out* before the execution of the XOR operation. This state might be unknown to the attacker, and therefore the exploitation of this attack is not guaranteed;
- Red cells: the effect of the attack can be predicted and exploited, no matter the initial states of the memristors.

	AS1	AS2	AS3 (Low V_{SET} and V_0)				
	Low V_{SET}	Low V_0	1 step	2 steps	3 steps	4 steps	5 steps
Step 1 (S1)	0X10	0111	0X1X				
Step 2 (S2)	01X0	0111	01XX	0XXX (S1+S2)			
Step 3 (S3)	011X	0000	0XXX	0XXX (S2+S3)	0XXX (S1+S2+S3)		
Step 4 (S4)	0110	0000	0010	00XX (S3+S4)	00XX (S2+S3+S4)	XXXX (S1+S2+S3+S4)	
Step 5 (S5)	0010	1111	1010	1010 (S4+S5)	1010 (S3+S4+S5)	XXXX (S2+S3+S4+S5)	XXXX (all steps)

Table II. Results of the attack on the XOR gate for the 3 proposed attack scenarios. The four bits in each cell represents the result of the XOR for the four input combinations 00,01,10,11. The expected value is "0110".

Based on the results in Table II, we can conclude that an attack targeting a perturbation of V_0 in one cycle, will always succeed in altering the output of the XOR computation. If the attack is performed with lower resolution (i.e., targeting both V_{SET} and V_0 over one or multiple cycles), the probability of a successful attack is reduced.

V. Conclusion

In-Memory Computing holds great potential for tackling the obstacles encountered by conventional computing architectures, including the memory wall and the energy expenditure of data transfer. Moreover, this approach could enhance security measures, as limited data movement decreases the likelihood of information leakage through communication buses. In this paper, we have presented the results of side-channel and fault analyses on the MAGIC solution, which is representative of in-array memristive-based logic computation.

We have shown that the MAGIC-based XOR operation are sensitive to both side-channel analysis and fault attacks. More in particular, we have demonstrated that the significant correlation with processed data resulting from the utilization of resistors with vastly different resistance, highlights the potential for side-channel analysis. We have also shown that

the perturbation of voltage sources is an efficient means of inflicting fault attacks. To conclude, even under the assumption supported by In-Memory Computing that there is no data movement, it is still crucial to implement countermeasures to safeguard sensitive data and ensure the integrity of the computation. Regarding countermeasures, the classical ones used for CMOS-based architectures, such as hiding or masking, could represent a first approach to reduce the information leakage: the key idea is to mask the connection between the leakage of the device and the operations being processed. However, [3] indicates that hiding could be not as effective as for CMOS, and it suggests to carefully assess such countermeasures for memristive applications, showing the need for new methods to protect memristive LIM implementation from Side Channel Attacks.

REFERENCES

[1] S. Kvatinsky *et al.*, "MAGIC—Memristor-Aided Logic," *IEEE Trans. Circuits Syst. II*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[2] S. Sayyah Ensan *et al.*, "SCARE: Side Channel Attack on In-Memory Computing for Reverse Engineering," *IEEE Transactions on VLSI Systems*, vol. 29, no. 12, pp. 2040–2051, Dec. 2021.

[3] L.-W. Chen *et al.*, "On Side-Channel Analysis of Memristive Cryptographic Circuits," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 463–476, 2023.

[4] L. Xie *et al.*, "Scouting Logic: A Novel Memristor-Based Logic Design for Resistive Computing," in *2017 IEEE ISVLSI*, Jul. 2017

[5] S. Gupta *et al.*, "FELIX: fast and energy-efficient logic in memory," in *Proceedings of the ICCAD*, Nov. 2018

[6] S. Kvatinsky *et al.*, "Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies," *IEEE Transactions on VLSI Systems*, vol. 22, no. 10, pp. 2054–2066, Oct. 2014.

[7] P. Inglese *et al.*, "On the Limitations of Concatenating Boolean Operations in Memristive-Based Logic-In-Memory Solutions," in *2021 16th DTIS*, Jun. 2021

[8] S. Kvatinsky *et al.*, "VTEAM: A General Model for Voltage-Controlled Memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.

[9] P. Inglese *et al.*, "Memristive Logic-in-Memory Implementations: A Comparison," in *SMACD / PRIME 2021*, Jul. 2021, pp. 1–4.

[10] P. Kocher *et al.*, "Differential Power Analysis," in *Advances in Cryptology — CRYPTO' 99*, Berlin, Heidelberg, 1999, pp. 388–397.

[11] G. Di Natale *et al.*, "An Integrated Validation Environment for Differential Power Analysis," in *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, Jan. 2008, pp. 527–532.

[12] D. Karaklajić *et al.*, "Hardware Designer's Guide to Fault Attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 12, pp. 2295–2306, Dec. 2013.

BIOGRAPHIES

Pietro Inglese is currently a PhD Candidate at TIMA Laboratory. He received the M.S. degree in Electronic Engineering from Politecnico di Torino in 2019. His research interests are In-Memory Computing and Hardware Security.

Elena-Ioana Vatajelu is researcher with CNRS on the design, test and reliability of Integrated Circuits. She obtained her PhD from UPC Spain in 2011. Her expertise is on the reliability and the robustness assessment, design-for-reliability, test strategies and security primitives for CMOS and beyond CMOS RAMs in traditional and non-Von Neumann computing paradigms.

Giorgio Di Natale received the PhD in Computer Engineering in 2003. He works as Director of Research with CNRS. His research interests include hardware security and trust, secure circuits design and test, reliability evaluation and fault tolerance.