Memristor-Based Lightweight Encryption

Muhammad Ali Siddiqi*§, Jan Andrés Galvan Hernández*, Anteneh Gebregiorgis*, Rajendra Bishnoi*, Christos Strydis*§, Said Hamdioui*‡ and Mottaqiallah Taouil*‡
*Quantum and Computer Engineering Department, Delft University of Technology, The Netherlands
§Department of Neuroscience, Erasmus Medical Center, Rotterdam, The Netherlands
‡Cognitive IC B.V., Delft, The Netherlands
{m.a.siddiqi, a.b.gebregiorgis, r.k.bishnoi, c.strydis, s.hamdioui, m.taouil}@tudelft.nl, jan-andres@live.nl

Abstract—**Next-generation personalized healthcare devices are undergoing extreme miniaturization in order to improve user acceptability. However, such developments make it difficult to incorporate cryptographic primitives using available target technologies since these algorithms are notorious for their energy consumption. Besides, strengthening these schemes against side-channel attacks further adds to the device overheads. Therefore, viable alternatives among emerging technologies are being sought. In this work, we investigate the possibility of using memristors for implementing lightweight encryption. We propose a 40-nm RRAM-based GIFT-cipher implementation using a 1T1R configuration with promising results; it exhibits roughly half the energy consumption of a CMOS-only implementation. More importantly, its non-volatile and reconfigurable substitution boxes offer an energy-efficient protection mechanism against side-channel attacks. The complete cipher takes 0.0034 mm^2 of area, and encrypting a 128-bit block consumes a mere 242 pJ.**

Index Terms—**Memristor, hardware security, lightweight encryption, side-channel attack, GIFT cipher, 1T1R**

I. Introduction

In recent years, small-form-factor edge devices have been considered as crucial components for next-generation personalized healthcare through medical body area networks (MBANs) [1]. Naturally, due to the sensitive nature of these devices, security and privacy become a critical concern, which require proper incorporation of cryptographic primitives and secure communication protocols. However, due to the ultra-resource-constrained nature of these devices, such as *mm-sized* neural implants, security primitives implemented using available CMOS technology become too costly for integration. The problem is further compounded when adding additional protection mechanisms to protect the security mechanisms themselves against side-channel attacks. Therefore, new emerging technologies need to be explored to address this issue.

New memristive technologies have recently been employed in implementing cryptographic primitives [2]–[12] because of their energy efficiency and ability to protect against side-channel attacks. However, to the best of our knowledge, there are no works that attempt to *simultaneously* take advantage of these characteristics for *block encryption*, which is the preferred type of encryption for resource-constrained devices.

In this work, we attempt to explore the use of memristors in constructing a lightweight block cipher that *also* offers side-channel protection at no significant additional cost. We first explore lightweight block ciphers in literature and select a suitable candidate (i.e., GIFT) to showcase the potential effectiveness of memristors in securing ultra-resource-constrained edge devices. We then implement the selected cipher using a 1T1R memristive crossbar. In essence, this work provides the following contributions:

- Restructuring of a round of GIFT encryption (i.e., substitution, permutation and round-key addition) in order to allow RRAM-crossbar implementation. This makes it possible to execute a round in a *single read* operation and to *reuse* the same hardware for multiple (40) rounds.
- Exploration and evaluation of the design choices for implementing XOR functionality that is used in round-key addition.
- A 40-nm implementation of the GIFT cipher using a 1T1R RRAM configuration.

The rest of the paper is organized as follows: Section II provides a brief background and an overview of related works. Section III discusses a suitable cipher that can be used to showcase the utility of memristors in lightweight encryption. Section IV explains our proposed scheme followed by results in Section V. We draw overall conclusions in Section VI.

II. Background

A. Memristor and RRAM background

The memristor is a two-terminal element (see Figure 1) that behaves like an ordinary resistor at a given instance [13]. It is *non-volatile* and can either have a Low-Resistance State (LRS) or a High-Resistance State (HRS). These states depend on the voltage applied to one of the terminals and the duration over which this is done. It can be used as a non-volatile memory element, where HRS and LRS usually denote logic states '0' and '1', respectively. The processes of changing the resistance from LRS to HRS and HRS to LRS are called *reset* and *set*, respectively. As an emerging technology, memristor offers simplicity, fast switching speeds, ultra-low power consumption and high integration density [14]. On top of that, the memristor is CMOS-compatible and shows the potential to overcome the von-Neumann bottleneck and sizing problem of transistors [15].

It is most common to configure memristors in a crossbar structure due to its simplicity. A memristor crossbar is traditionally used as a memory structure to, for example, replace the traditional SRAM. This is also referred to as the Resistive-RAM (RRAM). Additionally, the structure may be used as an

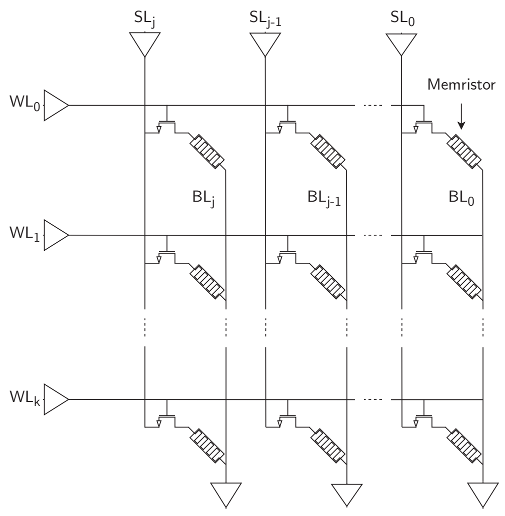


Fig. 1. Basic structure of a 1T1R crossbar array.

accelerator to drive neuromorphic applications by means of Vector-Matrix Multiplications (VMM) [16].

There are many different crossbar configurations, which are referred to as *n-element-m-resistor* (nXmR) arrays; one crossbar bitcell consist of *n* element(s) (such as, transistors or diodes) and *m* memristor(s). The most basic and area-efficient configuration is 1R, which only consists of memristors. However, its difficulty lies in selecting individual bitcells: without a specified selector, current sneak paths are induced. This problem is avoided by implementing crossbars with an additional element per bitcell, such as by using 1T1R (1-transistor-1-memristor) (see Figure 1), which is the most popular and widely used structure. It enables easy selection and programming of the bitcells for in-memory (in-situ) computations using Word Lines (WL) and Selection Lines (SL) [15]. Next to that, the transistor and memristor elements can be stacked, allowing greater density. It is also possible to construct crossbars using more (memristive) elements. An example of this is the 2T2R crossbar proposed in [17].

B. Memristor-based hardware security

Besides their energy efficiency, memristors intrinsically show stochastic behavior [18] like resistance variability [7], [19], probabilistic switching [5], [6] and Random Telegraph Noise (RTN) [20]. On top of that, they also exhibit Device-2-Device/Cycle-2-Cycle (D2D/C2C) variations [4], [6], which increase the difficulty of performing side-channel analysis as they increase the noise levels in traces and make it harder to perform profiling attacks. With the aforementioned stochastic properties, Physical Unclonable Functions (PUFs), True Random Number Generators (TRNGs), and chaotic circuits can be built depending on the employed resistive material [2], [3], [5], [9], [21]. These security schemes are used for authentication, key-generation and encryption.

In terms of *encryption*, a memristor-based block cipher can offer the following advantages: A memristor crossbar allows reconfigurability, which can be used to update the cipher substitution boxes (SBs) for masking at run-time to protect against side-channel attacks [22]. In this case, the cipher output should be adjusted accordingly to maintain the correct functionality. The non-volatile nature of such crossbars allow the SBs to retain their value, which allows us to power off the design to save energy. On the other hand, a CMOS-only implementation will be using e.g., an SRAM or logic implementations of the SBs. SRAM-based solutions allow reconfiguration but have less variation and cannot be powered off, whereas logic implementations (such as and-or trees) cannot be modified. Another option could be to have embedded flash-based SBs, however, these memories are more power consuming than memristor-based ones (such as, RRAM) during active mode [23].

C. Related work

Sun et al. [5] propose a memristor-based PUF that prevents attacks using triggered solubility when necessary. This is achieved by using water-assisted transfer printing. Cai et al. [6] use a randomly-initialized memristor crossbar to perform VMM for creating hypervectors as a means for encryption. It relies on crossbar non-idealities and C2C variations. However, for decryption, the authors propose a neural network. Similarly, in [4], a 1T1R crossbar is used to store plaintext and perform in-situ XOR operations with key bits for encryption. The key bits are generated using the subthreshold slope of each transistor. These vary intrinsically and hence function as a PUF. Two other works discuss the concept of key-less encryption, using memristors as a source of entropy [3], [10]. Khedkar et al. [24] propose a memristor-based AES design with the aim of protecting against differential power attacks. Each AES SB is implemented using a neural network having 8 inputs, 1 output, and a hidden layer of 48 neurons, which is not suitable for heavily resource-constrained edge devices. However, it should be noted that energy efficiency was not the aim of their work.

Despite these promising schemes, many of them are not necessarily lightweight, especially the above-mentioned block cipher. In addition, the majority of these solutions require frequent operational switching of memristors, which should be avoided due to the increase in energy consumption and decrease in lifetime of the memristive device [16]. In brief, there is still a void in literature when it comes to a lightweight memristor-based block cipher that also supports energy-efficient side-channel protection.

III. Towards memristor-based encryption

A. Looking for the right cipher

In order to showcase the possibility of using memristors in lightweight block encryption, we refer to existing literature to find a suitable encryption scheme that can serve as a proof of concept. Multiple extensive literature reviews on the state-of-the-art lightweight cryptography already exist [25]–[27]. We evaluate the available ciphers by looking at the **throughput**, **power/energy consumption**, **design simplicity**, and most importantly, their potential **compatibility with memristors**.

These include: PRESENT, RECTANGLE, SIMON, SPECK, GIFT, SLIM, μ^2, ANU-II, NLBIST, Piccolo and BORON.

In terms of throughput and energy, Speck, Simon and GIFT show the best results. Regarding simplicity, GIFT outperforms the other two [25]. Moreover, its simple structure (mainly consisting of substitution boxes) enables a straightforward crossbar implementation. More specifically, a memristor crossbar can be composed in a way such that a GIFT encryption round can be performed by only a single *read* action (as will be discussed in Section IV). For these reasons, the GIFT cipher is used as a reference and inspiration towards implementing a lightweight memristor-based encryption block suitable for next-generation edge devices. The next section will briefly explain the working principle of GIFT.

B. The GIFT cipher

Similar to the known standardized algorithms such as AES, SKINNY and PRESENT, GIFT is based on substitution-permutation network, in which the plaintext nibbles are replaced with other values, followed by rearrangement. The cipher is inspired by its predecessor PRESENT but has improved security and efficiency [28]. The GIFT family consists of two members: GIFT-64 and GIFT-128. The former takes in 64 bits and uses 28 rounds while the latter encrypts 128 bits using 40 rounds. Both versions use a 128-bit encryption key. Figure 2 shows the cipher architecture. A round of GIFT consists of three basic operations:

1) **SubCells**: The plaintext nibble is substituted with a 4-bit sequence through an SB. Figure 2 shows the substitution specification for such a 4-bit SB.
2) **PermBits**: The output of the SB is permuted. Each of the 4 bits is re-routed through hard-wiring.
3) **AddRoundKey**: For the 64(128)-bit version a 32(64)-bit Round Key (RK) is extracted from the *key state*, which is partitioned into 2(4) 16-bit words. Two bits of each of the permuted nibbles (i.e., two LSBs of the nibbles in the case of GIFT-64, and middle two bits in the case of GIFT-128) are XOR-ed with two bits from the RK. Figure 2 illustrates this for a single round of GIFT-64.

Since only two key bits per nibble are used when performing the XOR operations, each RK is updated and shifted after every round to ensure that the other part of the key state is used. This is done by performing a 32-bit right rotation. This is followed by a 2-bit and 12-bit right rotation performed on the two MSB bytes and the two bytes thereafter, respectively. Figure 2 depicts the complete key state update. In addition to the RK, there also exists a 7-bit Round Constant (RC), which is applied to bit positions $n-1$, 3, 7, 11, 15, 19 and 23, where $n = 64, 128$. After every round, the RC is updated by means of a rotational left shift followed by two XOR operations between the new LSB, MSB and '1'. The RK and RC schedules are the same for both versions of the cipher.

IV. Proposed Solution

A. Design overview and assumptions

The general idea behind our approach is to take advantage of the bit-slicing topology of GIFT and compress all the operations shown in Figure 2 into *one* lightweight module made of a memristor-based crossbar. As stated earlier, GIFT is a natural candidate for a crossbar-based implementation due to its simple structure. For example, the first operation, i.e., *SubCells*, is realized using 4-bit SBs, which are normally implemented using Look-Up Tables (LUTs). These LUTs can in turn be implemented using memristor crossbars. A memristive crossbar structure is preferred over cascaded Memristor-based Logic Gates (MLGs) due to the possibility of high-density and in-memory computations. Moreover, a crossbar structure mimicking LUTs eliminates frequent writing/reset operations on memristors, which can introduce reliability issues and shorten lifetime of these components [16]. Figure 3 shows the top view of the proposed design. To the best of our knowledge, no other work has proposed a memristor-based lightweight block cipher at the time of writing.

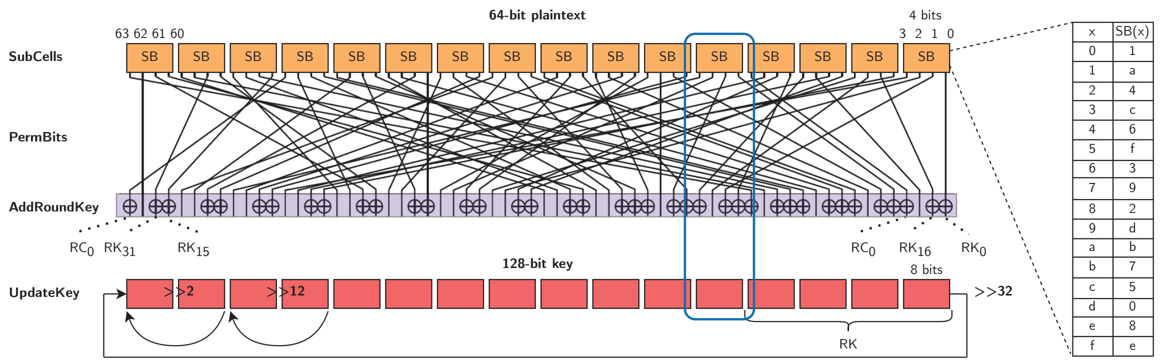
The key aspects of our approach are as follows:

- Mapping the three GIFT operations and key scheduling to a memristor crossbar makes it possible to execute a GIFT encryption round in a single *read* operation.
- Only at the start of an encryption session would a *write* operation be required to program the constant/key-bit values.
- Each crossbar implementation covers the 40 encryption rounds, i.e., the same hardware is reused for each round.
- By minimizing switching activity and by mapping all the operations to the crossbar, lesser energy consumption and footprint is achieved.

Before diving deeper into the design, a couple of assumptions must be established. Firstly, in this work, the 128-bit GIFT version is considered. However, our approach also applies to the 64-bit version. Secondly, the encryption key needs to be created and exchanged beforehand between the transmitter and the receiver to perform the encryption. This prior key exchange is considered outside the scope of the paper and has been already addressed by prior work. For instance, in the case of a medical-implant edge application, a body-coupled-communication-based protocol can be used for this purpose [29].

B. Substitution box

Regarding the memristor-based LUT implementation of the SB, there are multiple options for the crossbar structure: Passive crossbar arrays (1R) are an attractive solution for high-density and low-power integration. However, they have weaknesses such as current sneak paths and floating state issues during reading and writing operations. Furthermore, this leakage becomes more severe when the number of cells increases [30]. An established solution to this problem is the nTnR structure. The most commonly used structure is 1T1R, but 2T2R has also shown significant benefits [17].



x	SB(x)
0	1
1	a
2	4
3	c
4	6
5	f
6	3
7	9
8	2
9	d
a	b
b	7
c	5
d	0
e	8
f	e

Fig. 2. Architecture of GIFT-64 [28]. The encircled section (in blue) denotes a slice of a single round of encryption. The substitution specification of a 4-bit SB is shown on the right.

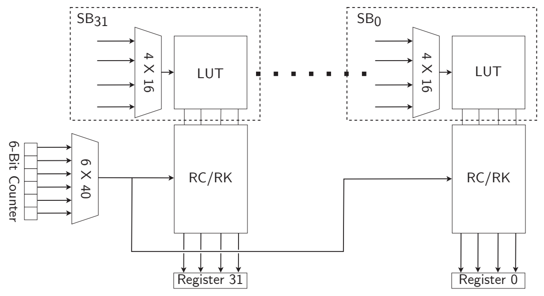


Fig. 3. Top view of the memristor-based GIFT cipher.

However, as the name suggests, 2T2R is twice the size of 1T1R. Mapping an SB LUT onto a 1T1R crossbar results in the structure shown in Figure 4. The SB unit can have 16 possible 4-bit input values, and the internal LUT has 16 rows of four memristor/transistor pairs, with each memristive cell representing 1 bit of the SB output. Given a 4-bit input, an address decoder selects one of the 16 rows (corresponding to Figure 2) by applying a voltage to the word line (WL) connected to the transistor gates of that row, which enables the respective memristive cells. The details of the crossbar operation will be provided in Section IV-E.

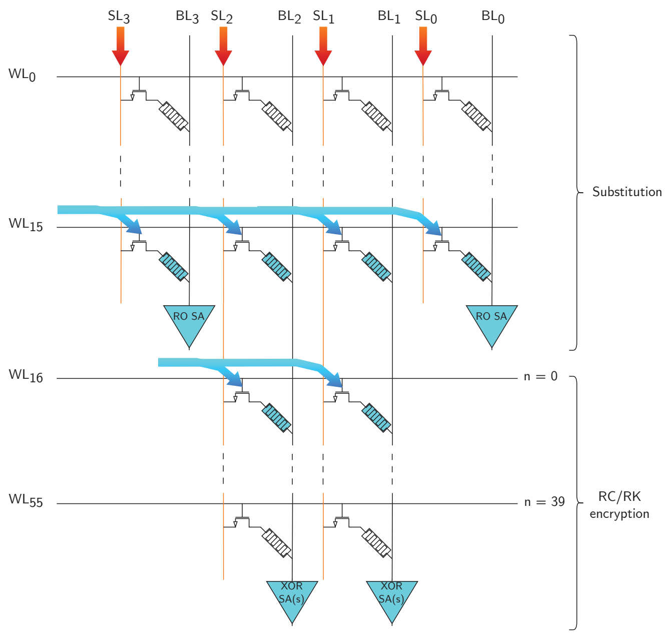


Fig. 4. The slice architecture of the 1T1R GIFT cipher and operation sequence corresponding to a round of encryption.

C. XOR operation

The second operation of the GIFT encryption is *AddRoundKey*, which is performed using XOR. Finding the appropriate MLG for such an operation is crucial for achieving low energy and area overheads. As a result, the following notable MLG architectures from literature were considered: Memristor-Aided Logic (MAGIC) [31], Single-Cycle In-Memristor XOR (SIXOR) [30], Scouting Logic [32], 1T1R In-Situ Boolean Logic [33], Stateful 1T1R NANDs [34], Material Implication Memristor Logic [35], Memristor-Ratioed Logic (MRL) [36], CMOS-Like Logic [37], Parallel Input-Processing Memristor Logic [38], and Dual Sense Amplifier Crossbar Logic (DSA) [39].

Many of these MLGs are limited due to (1) cascading problems, such as requiring signal restoration between stages, etc.; (2) destructiveness, i.e., the follow-up operation requires all memristors to be initialized again by means of a write operation; and (3) a long sequence of operations. MLGs implemented in a constrained environment should have very low power consumption and acceptable delay. MLGs that require many sequential resets may, therefore, be less desirable. As of now, the usage of standalone MLGs such as SIXOR, MAGIC, and MRL is too inefficient for encryption in next-generation edge devices [12]. On the other hand, crossbar solutions show good potential for parallel in-memory computing applications. Moreover, they can be easily made compatible with the SB-unit crossbar. We thus employ both Scouting Logic and DSA to implement XOR in this work:

1) Scouting-Logic-based XOR (SXOR): The Scouting Logic family [32] offers OR, AND, and XOR operations and is crossbar compatible. It is non-destructive as the resistive states are preserved across multiple operations. Hence, no switching

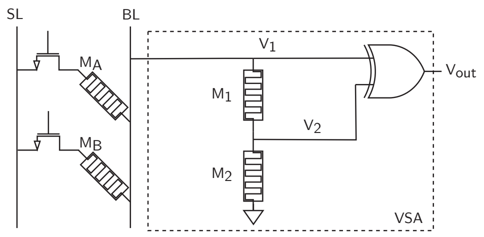


Fig. 5. SXOR: VSA-based Scouting Logic XOR.

between resistive states is necessary, which is power-efficient. Moreover, a single logic operation only requires one step, in which no initialization and restoration are needed as it primarily consists of a reading operation. The core of the Scouting Logic design varies depending on the employed sensing scheme, i.e., voltage (VSA) and current sense amplifiers (CSA). It has been shown in [12] that CSA has higher area and power consumption overheads than VSA. However, it is faster than VSA. Since performance is not a bottleneck for the targeted edge applications, VSA is employed in this work for the SXOR design (see Figure 5).

VSA generates a reference using additional memristors M_1 and M_2. It also uses a CMOS XOR gate as a threshold function to determine the proper output pulse based on the generated reference. When a read pulse is applied from SL to M_A and M_B (which contain the two inputs for the XOR operation), a current determined by the equivalent input resistance will flow from the Bit Line (BL) to the VSA. Depending on whether the current (or resulting voltage) is higher or lower than the gate threshold, the output will be either '0' or '1'.

2) DSA-based XOR (DXOR): In DSA [39], complex gates, such as bitwise XOR, are composed using two sense amplifiers (SAs). The DSA scheme is meant for crossbar operations, and due to its simplicity, it is able to perform the operations using a single cycle only. However, this comes at the cost of using two SAs and a single 2-bit NOR gate.

By combining the reference currents of the NOR and NAND SAs, an XOR operation is realized, following the equation $Y_{\text{XOR}} = X1_{\text{AND}}$ **NOR** $X2_{\text{NOR}}$, as shown in Figure 6, where $X1$ and $X2$ are the outputs of the respective SAs.

In [39], current mirrors are used in the crossbar to drive the SAs. For the design of the DXOR-GIFT, a voltage-based SA is used to eliminate the need for current mirrors, resulting in higher energy efficiency. The schematic for the voltage-based SA is adapted from the SA proposed in [39], with minor alterations to target only the XOR functionality (see Figure 6).

D. Permutation

Considering the permutation scheme in [28], which is also depicted as the web of connections in Figure 2, each n^{th} bit of every nibble is connected to the n^{th} bit of another nibble that drives the SB of the next round. For example, the 1^{st} bit (i.e., the second output) of SB_0 (the rightmost SB in the first round) is connected (permuted) to the second input of SB_4 during the next round, which is also encircled in Figure 2. Thus,

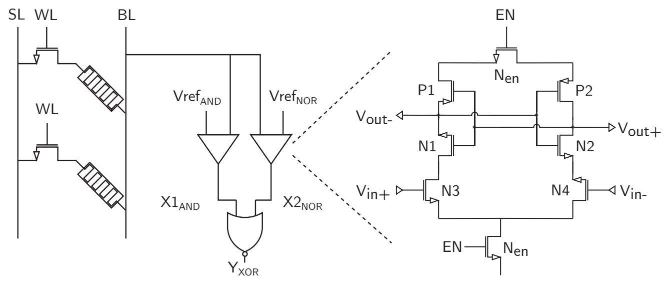


Fig. 6. DXOR: DSA-based XOR with the expanded voltage-based SA (right).

conceptually, SB_0 of round 1 drives SB_4 of round 2 with an XOR operation in between. However, there is no difference between the SBs of each round since they are all identical. This means that one can consider the n^{th} output of SB_n to simply be fed back to the n^{th} input of the same SB.

Figure 2 shows that it is always the same bits in each nibble that are involved in XOR operations. The only thing that changes is the value of the key bit after every *UpdateKey*. Knowing this and the fact that every bit in the nibble is routed to the same relative position, it is possible to anticipate the RK and RC bit values that are used in the XOR operations of every round if the encryption key is known. More specifically, instead of performing RK/RC updates every round, these values are pre-computed (offline) and arranged at their relative bit positions within the nibble after following the permutation table. During the initialization of the encryption module, these values are uploaded only once to the memristors in WL_{16} to WL_{55}, which saves significant overhead due to the removal of the active key-schedule operation in its entirety.

E. Encryption round

The means of incorporating the permutation and key-scheduling approach are illustrated in Figure 4. In addition to the SB crossbar proposed in Section IV-B for *substitution*, a 40×2 1T1R crossbar (or 40×3 for the few nibbles with additional RC) is connected in series for *RC/RK encryption*. As mentioned in the previous section, this 40-row crossbar (WL_{16} to WL_{55}) contains the RK (and RC) bit values arranged according to the permutation table and the key schedule for the complete encryption of a 128-bit plaintext. This means that *active* key scheduling is not required anymore. For GIFT-128, only the middle two bits of each nibble are encrypted with the appropriate RK bits [28], as opposed to the first two bits of each nibble for GIFT-64. The XOR SAs (i.e., DSA-based or scouting-logic-based) are connected to the bottom of the crossbar. Figure 4 also illustrates the flow of all the above operations for a single slice. An encryption round is as follows:

1) Based on the SB input and the mapping presented in Figure 2, the corresponding WL is driven by a voltage pulse, upon which the NMOS switches in that row are closed, and the respective memristors are selected. In addition, the RC/RK memristors corresponding to the first

round are selected. The activated rows are shown using blue arrows in Figure 4.

2) A read pulse is generated for each SL (red arrows in Figure 4), allowing current to flow through the memristors and the BL, into the XOR and read-out (RO) SAs.
3) With the desired rows selected, an XOR operation is performed between the top and bottom parts for bitcells 1 and 2, respectively. Depending on the resistive states of these bits, a '0' or '1' pulse is generated at the output. Bitcells 0 and 3 are read out without performing XOR.
4) All the above steps are performed within the same cycle, and the output is temporarily stored in an output register.
5) In the next round, the register output is fed back again to the input of the same SB.

F. Crossbar address decoders

The purpose of the address decoders in Figure 3 is to drive the desired WLs in the crossbar, resulting in the selection of the memristors embedded in the corresponding row. These decoders can add significantly to the design overhead since our GIFT-128 design requires 32 4-to-16 address decoders and a single 6-to-40 address decoder (see Section IV-G). As a result, different designs were considered for the optimized implementation of these decoders. For example, one approach is to decode the bits by means of a sequence of PMOS and NMOS transistors connected in series, where each one of them corresponds to the expected input bit. The disadvantage of such a circuit is that it requires precise sizing of the transistors. Moreover, the voltage drop caused across a transistor becomes an issue when considering the transistor overdrive. Consequently, a repeater is required to provide enough input drive for the WLs. A better solution is to use a complete logic scheme that allows for the use of the smallest possible transistor sizes. A second approach is based on NAND/NOR trees, such as the one proposed for decoding addresses in SRAMs [40], [41]. This approach is relatively lightweight, reliable, and can also be applied to memristor crossbar arrays, which is why it is used in this work.

G. RC/RK selector

The entire architecture requires a single RC/RK selector for selecting the WLs in all the 32 RC/RK units shown in Figure 3. The goal of this selector is to select a WL for every round, from round 1 to 40. One approach is to use a shift register for selecting each WL after every bit shift. However, a 40-bit shift register is costlier than using a 6-bit counter that drives a 6-to-40 address decoder. As a result, the latter option is used for our purpose.

V. Results

A. Experimental setup

Both the 1T1R-GIFT designs, i.e., the ones based on the DSA-based XOR (DXOR-GIFT) and scouting-logic-based XOR (SXOR-GIFT), were implemented as SPICE netlists using Cadence Virtuoso. Cadence Spectre was used for design simulation/verification and power measurements. The designs were realized using the TSMC 40 nm library. In the original work [28], the GIFT implementation runs at a frequency of 10 MHz. The same is done for the 1T1R implementations for consistency. Table I summarizes the design parameters for these implementations.

TABLE I
Design parameters

Technology node	40 nm
Technology	RRAM (HfO_2*)
Operating voltage	0.9
Operating frequency	10 MHz

* JART VCM v1b model from [19].

B. Implementation

The 1T1R-GIFT SPICE netlist entails a single GIFT-128 slice, as highlighted in Section IV-A. The 1T1R model employed in our implementation is adapted from [17], which uses the HfO_2-based memristor from [19] and considers non-idealities such as wire resistances and capacitances. All the CMOS-based logic used in the designs is implemented using the cells provided by the TSMC 40 nm standard library. In general, minimal size is used for the gates. However, the gates in the final stages of the decoders are four times larger to ensure sufficient drive strength for the WLs.

1) SXOR-GIFT: In [32], while operating Scouting Logic VSA, the resistive states of the memristors are retained. Hence, as with all other memristors in this design, they only need to be programmed once. For the voltage divider to work properly, the acting memristors need to be scaled accordingly to guarantee an output of (at least) 0.6 V_{DD} and 0.4 V_{DD} for logic '1' and '0', respectively. Since this work utilizes a different structure and technology, the scaling rules stated in [32] cannot be applied. For example, the original Scouting Logic VSA requires M_1 and M_2 to be $2\times LRS$ and $2.5\times LRS$, respectively. However, when doing so for the proposed cipher, the small resistive values result in V_1 and V_2 not crossing 0.20-0.30 V, which is way below the required threshold value of 0.45 V. Moreover, keeping the 1:1.25 ratio between the two memristors in Scouting Logic VSA results in too large of a margin between V_1 and V_2, causing V_2 to stay below the threshold and resulting in operational failure. Using the simplified model of the employed memristor crossbar, the proper resistor values were determined: it was found that SXOR-GIFT performs well when using 2 kΩ and 250 kΩ for M1 and M2, respectively. Since the XOR operations are only performed on the middle two bits of each nibble, the remaining LSBs and MSBs just need to be read out using a read-out (RO) SA. For this purpose, Scouting Logic VSA is downsized to just a single memristor, M_1, which is programmed to 550 kΩ, and the XOR gate in the SA is replaced by an OR gate.

2) DXOR-GIFT: For the AND and NOR SAs in the DXOR sensing schemes, constant voltage references of 0.45 V and 0.43 V are used, respectively. Regarding the read-out of the LSBs and MSBs, only a single read-out SA suffices, for which a reference of 0.43 V is used.

TABLE II
IMPLEMENTATION RESULTS

	SXOR-GIFT	DXOR-GIFT	CMOS-GIFT[§] [28]
Average Power (μW)	257.6*	60.38*	116.6
Energy (pJ)	1030.4	241.52	478.1
Area (mm^2)	0.0034	0.0034	–
Latency** (us)	4	4	4

* Extrapolated for 32 slices | ** Duration of 40 encryption rounds (10 MHz clock) | § Using STM 90 nm library | '–': Lacking information.

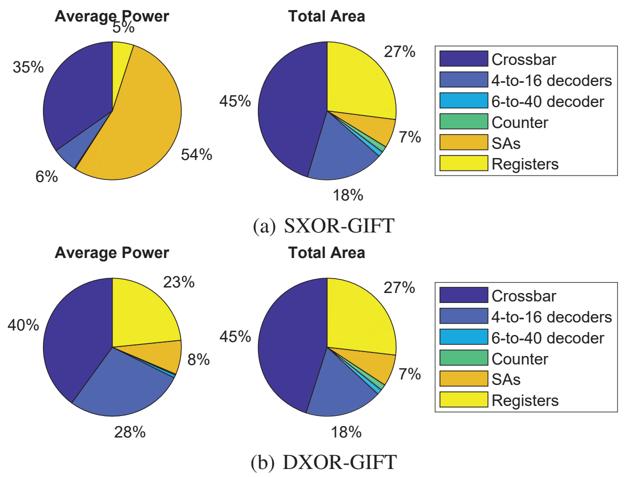


Fig. 7. Average-power and total-area breakdown

C. Energy- and area-efficiency analysis

The implementation results are summarized in Table II. It can be seen that both SXOR-GIFT and DXOR-GIFT consume almost the same amount of area. However, SXOR-GIFT has more than $4\times$ the power consumption of DXOR-GIFT. The breakdown of the average power and total area is illustrated in Figure 7. It can be seen that the scouting logic SAs are the bottleneck in the case of SXOR-GIFT, whereas power consumption is relatively evenly spread in the case of DXOR-GIFT. Overall, DXOR-GIFT outperforms SXOR-GIFT and hence can be considered a better approach for implementing lightweight block ciphers.

D. Discussion

Our analysis indicates that memristors show significant promise in terms of their use as building blocks for lightweight block ciphers. In terms of energy, processing, and area overheads, the results of the 1T1R implementation are roughly twice as good as those of the prior 90-nm CMOS-only implementation [28], as shown in Table II. More importantly, this new approach to designing lightweight block ciphers offers additional low-cost protection against side-channel and template attacks, as highlighted in Section II-B. Lastly, our analysis shows that DXOR is a preferable approach compared to the state-of-the-art MLGs available in the literature. However, there may be room for improvement in the 1T1R-GIFT implementation depending on future advances in technology:

Researchers have pointed out the difficulty of creating long MLG chains due to the loss in drive strength over several gate stages and the destructive nature of the operations. Despite numerous solutions, it is still sub-optimal due to limitations related to the number of operands, the required number of repeaters, and the number of operations. Solving these problems would allow SB construction using a simple memristor-based AND-OR tree, thereby significantly reducing the footprint.

Another alternative could be to shift towards a 2T2R approach, as in [17]. It would result in a twofold increase in area, but in-situ operations can be done using differential sensing, which doubles the sensing margin compared to [32] and hence increases reliability. Also, the energy consumption with 2T2R operands is significantly lower. Unfortunately, this structure does not support XOR operations yet. Besides, researchers are also exploring the possibility of *stacking* memristor crossbars, thereby significantly reducing the architecture footprint. These are also referred to as *3D vertical RRAMs*. Some successful proposals exist that exploit different memristor-transistor compositions to produce 1T1R pillars [42], [43].

Lastly, the transistor of the 1T1R structure is a limiting factor in scaling down the crossbar. However, advancements in resistive switching materials and selection methods can help in solving this limitation.

So, can ultra-resource-constrained edge devices benefit from a memristor-based security solution? At the moment, it does seem so. On top of that, further advancements in RRAM technology can potentially unlock the full capability of memristor architectures and, consequently, make them a fitting building block for ultra-lightweight security applications.

VI. CONCLUSIONS

In this paper, we proposed a lightweight implementation of the GIFT cipher using an RRAM-based architecture in a 1T1R configuration. The design was implemented using a 40 nm process technology. Not only do our scheme's area and power overheads compare favorably to those of a CMOS-only implementation, but it also allows for housing the substitution boxes (SBs) in an energy-friendly and non-volatile fashion. The reconfigurability of these SBs allows the cipher operation to be masked, providing protection against side-channel attacks at no significant additional cost. Such a lightweight design can significantly contribute to securing small-form-factor edge devices for next-generation personalized healthcare.

ACKNOWLEDGMENT

This work has been supported by the EU-funded projects SEPTON, SECURED, and CONVOLVE, with the grant agreement numbers 101094901, 101095717, and 101070374, respectively.

REFERENCES

[1] M. A. Siddiqi, G. Hahn, S. Hamdioui, W. A. Serdijn, and C. Strydis, "Improving the security of the ieee 802.15. 6 standard for medical bans," *IEEE Access*, vol. 10, pp. 62 953–62 975, 2022.

[2] S. Lv, J. Liu, and Z. Geng, "Application of memristors in hardware security: A current state-of-the-art technology," *Advanced Intelligent Systems*, vol. 3, no. 1, p. 2000127, 2021.

[3] L. Wang, T. Dong, and M.-F. Ge, "Finite-time synchronization of memristor chaotic systems and its application in image encryption," *Applied Mathematics and Computation*, vol. 347, pp. 293–305, 2019.

[4] L. Yang, L. Cheng, Y. Li, H. Li, J. Li, T.-C. Chang, and X. Miao, "Cryptographic key generation and in situ encryption in one-transistor-one-resistor memristors for hardware security," *Advanced Electronic Materials*, vol. 7, no. 5, p. 2001182, 2021.

[5] J. Sun, Z. Wang, S. Wang, M. Yang, H. Gao, H. Wang, X. Ma, and Y. Hao, "Physical unclonable functions based on transient form of memristors for emergency defenses," *IEEE Electron Device Letters*, vol. 43, no. 3, pp. 378–381, 2022.

[6] J. Cai, A. Amirsoleimani, and R. Genov, "Hyperlock: In-memory hyperdimensional encryption in memristor crossbar array," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 960–964.

[7] W. Dai, X. Xu, X. Song, and G. Li, "Audio encryption algorithm based on chen memristor chaotic system," *Symmetry*, vol. 14, no. 1, p. 17, 2021.

[8] N. Du, H. Schmidt, and I. Polian, "Low-power emerging memristive designs towards secure hardware systems for applications in internet of things," *Nano Materials Science*, vol. 3, no. 2, pp. 186–204, 2021.

[9] Y. Pang, B. Gao, B. Lin, H. Qian, and H. Wu, "Memristors for hardware security applications," *Advanced Electronic Materials*, vol. 5, no. 9, p. 1800872, 2019.

[10] B. Cambou, D. Hély, and S. Assiri, "Cryptography with analog scheme using memristors," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 16, no. 4, pp. 1–30, 2020.

[11] A. P. James, "An overview of memristive cryptography," *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2301–2312, 2019.

[12] J. A. Galvan Hernández, "Memristive security in free-floating neural implants," Master's thesis, TU Delft Faculty of Electrical Engineering, Mathematics and Computer Science, 2022. [Online]. Available: http://resolver.tudelft.nl/uuid:cdd29307-3660-4dbf-921e-dda05faaf16c

[13] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[14] R. Wang, J.-Q. Yang, J.-Y. Mao, Z.-P. Wang, S. Wu, M. Zhou, T. Chen, Y. Zhou, and S.-T. Han, "Recent advances of volatile memristors: Devices, mechanisms, and applications," *Advanced Intelligent Systems*, vol. 2, no. 9, p. 2000055, 2020.

[15] H. Li, S. Wang, X. Zhang, W. Wang, R. Yang, Z. Sun, W. Feng, P. Lin, Z. Wang, L. Sun *et al.*, "Memristive crossbar arrays for storage and computing applications," *Advanced Intelligent Systems*, vol. 3, no. 9, p. 2100017, 2021.

[16] S. Zhang, G. L. Zhang, B. Li, H. H. Li, and U. Schlichtmann, "Aging-aware lifetime enhancement for memristor-based neuromorphic computing," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1751–1756.

[17] A. Singh, R. Bishnoi, R. V. Joshi, and S. Hamdioui, "Referencing-in-array scheme for rram-based cim architecture," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1413–1418.

[18] Y. Bai, H. Wu, R. Wu, Y. Zhang, N. Deng, Z. Yu, and H. Qian, "Study of multi-level characteristics for 3d vertical resistive switching memory," *Scientific reports*, vol. 4, no. 1, p. 5780, 2014.

[19] EMRL. (2020) JART – Jülich Aachen Resistive Switching Tools. [Online]. Available: http://www.emrl.de/JART#Artikel_1

[20] C.-Y. Huang, W. C. Shen, Y.-H. Tseng, Y.-C. King, and C.-J. Lin, "A contact-resistive random-access-memory-based true random number generator," *IEEE Electron Device Letters*, vol. 33, no. 8, pp. 1108–1110, 2012.

[21] Y. Liu, L. Chen, X. Li, Y. Liu, S. Hu, Q. Yu, T. Chen, and Y. Liu, "A dynamic aes cryptosystem based on memristive neural network," *Scientific Reports*, vol. 12, no. 1, p. 12983, 2022.

[22] H. Groß, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," *Cryptology ePrint Archive*, 2016.

[23] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori, "Evaluation of hybrid memory technologies using sot-mram for on-chip cache hierarchy," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 3, pp. 367–380, 2015.

[24] G. Khedkar, C. Donahue, and D. Kudithipudi, "Towards leakage resiliency: memristor-based aes design for differential power attack mitigation," in *Machine Intelligence and Bio-inspired Computation: Theory and Applications VIII*, vol. 9119. SPIE, 2014, pp. 40–50.

[25] A. Joseph *et al.*, "An analysis on low cost and performance of hardware and software oriented lightweight block ciphers for iot applications," in *Proceedings of the International Conference on IoT Based Control Networks & Intelligent Systems-ICICNIS*, 2021.

[26] V. A. Thakor, M. A. Razzaque, and M. R. Khandaker, "Lightweight cryptography algorithms for resource-constrained iot devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, pp. 28 177–28 193, 2021.

[27] N. M. Naser and J. R. Naif, "A systematic review of ultra-lightweight encryption algorithms," *International Journal of Nonlinear Analysis and Applications*, vol. 13, no. 1, pp. 3825–3851, 2022.

[28] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift: A small present: Towards reaching the limit of lightweight encryption," in *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Springer, 2017, pp. 321–345.

[29] M. A. Siddiqi, R. H. Beurskens, P. Kruizinga, C. I. De Zeeuw, and C. Strydis, "Securing implantable medical devices using ultrasound waves," *IEEE Access*, vol. 9, pp. 80 170–80 182, 2021.

[30] N. TaheriNejad, "Sixor: Single-cycle in-memristor XOR," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 5, pp. 925–935, 2021.

[31] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Magic—memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.

[32] L. Xie, H. A. Du Nguyen, J. Yu, A. Kaichouhi, M. Taouil, M. Al-Failakawi, and S. Hamdioui, "Scouting logic: A novel memristor-based logic design for resistive computing," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2017, pp. 176–181.

[33] Z.-R. Wang, Y.-T. Su, Y. Li, Y.-X. Zhou, T.-J. Chu, K.-C. Chang, T.-C. Chang, T.-M. Tsai, S. M. Sze, and X.-S. Miao, "Functionally complete boolean logic in 1t1r resistive random access memory," *IEEE Electron Device Letters*, vol. 38, no. 2, pp. 179–182, 2016.

[34] W. Shen, P. Huang, M. Fan, R. Han, Z. Zhou, B. Gao, H. Wu, H. Qian, L. Liu, X. Liu *et al.*, "Stateful logic operations in one-transistor-one-resistor resistive random access memory array," *IEEE Electron Device Letters*, vol. 40, no. 9, pp. 1538–1541, 2019.

[35] 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 Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, 2013.

[36] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Mrl—memristor ratioed logic," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*. IEEE, 2012, pp. 1–6.

[37] I. Vourkas and G. C. Sirakoulis, "A novel design and modeling paradigm for memristor-based crossbar circuits," *IEEE Transactions on Nanotechnology*, vol. 11, no. 6, pp. 1151–1159, 2012.

[38] G. Papandroulidakis, I. Vourkas, N. Vasileiadis, and G. C. Sirakoulis, "Boolean logic operations and computing circuits based on memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 12, pp. 972–976, 2014.

[39] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic ram," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 470–483, 2017.

[40] A. K. Mishra, D. P. Acharya, and P. K. Patra, "Novel design technique of address decoder for sram," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*. IEEE, 2014, pp. 1032–1035.

[41] S. Singh and S. Akashe, "Low power consuming 1 kb ($32\times$ 32) memory array using compact 7t sram cell," *Wireless Personal Communications*, vol. 96, pp. 1099–1109, 2017.

[42] J. Wu, F. Mo, T. Saraya, T. Hiramoto, and M. Kobayashi, "A monolithic 3-d integration of rram array and oxide semiconductor fet for in-memory computing in 3-d neural network," *IEEE Transactions on Electron Devices*, vol. 67, no. 12, pp. 5322–5328, 2020.

[43] M. Ezzadeen, D. Bosch, B. Giraud, S. Barraud, J.-P. Noel, D. Lattard, J. Lacord, J.-M. Portal, and F. Andrieu, "Ultrahigh-density 3-d vertical rram with stacked junctionless nanowires for in-memory-computing applications," *IEEE Transactions on Electron Devices*, vol. 67, no. 11, pp. 4626–4630, 2020.