This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3050555, IEEE Internet of Things Journal

Lightweight Fuzzy Extractor Based on LPN for Device and Biometric Authentication in IoT

Shahriar Ebrahimi, Siavash Bayat-Sarmadi, *Member, IEEE*

***Abstract*—User and device biometrics are proven to be a reliable source for authentication, especially for the internet of things (IoT) applications. One of the methods to employ biometric data in authentication are fuzzy extractors (FE) that can extract cryptographically-secure and reproducible keys from noisy biometric sources with some entropy loss. It has been shown that one can reliably build an FE based on the learning parity with noise (LPN) problem with higher error-tolerance than previous FE schemes. However, the only available LPN-based FE implementation suffers from extreme resource demands that are not practical for IoT devices. This paper proposes a lightweight hardware/software (HW/SW) co-design for implementing LPN-based FE. We provide different optimizations on architecture to decrease the resource requirements of the scheme. The proposed architecture is resistant against simple side-channel analysis and improves area and *area-time* product (AT) by more than 89% and 83%, respectively, compared to previous work. Our experimental results indicate that the proposed architecture can be implemented on off-the-shelf resource-constrained SoC-FPGA boards from different vendors such as Xilinx, Digilent, and Trenz. Moreover, we provide the first implementation results of LPN-based FE on an ASIC platform using HW/SW co-design.**

***Index Terms*—Fuzzy extractor (FE), biometric authentication, learning parity with noise (LPN), internet of things (IoT), hardware/software co-design.**

I. Introduction

To ensure the confidentiality and privacy of the users and their associated data in the internet of things (IoT), different security protocols are required that rely on at least a few cryptographic secret/private keys [1]. Secure storage and management of such keys can be very expensive or impractical for many IoT devices [2]. In addition, considering the distributed and remote structure of IoT end-nodes, pre-stored secret keys are extractable by employing physical attacks to device memories [3], [4].

In order to overcome issues related to secure key-management, many studies have suggested employing user or device biometrics for authentication [2], [5]. The user biometrics such as fingerprints and iris are unique to the users and can be exploited for both identification and authentication. On devices, physical unclonable functions (PUFs) act as an almost perfect biometric source unique to the device itself.

All authors are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. Emails: shebrahimi@ce.sharif.edu, sbayat@sharif.edu. This research was partially supported by Iran National Science Foundation (INSF) under the grant number 98012106 and Sharif University of Technology under the grant number G960803.

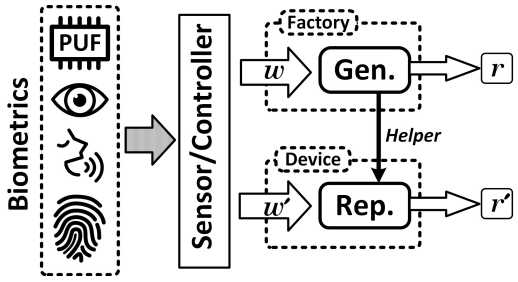

Fig. 1. Fuzzy Extraction Overview.

In 2004, Dodis et al. proposed the fuzzy extractor (FE) concept as the function that can extract cryptographically random and reproducible secure keys from noisy biometric inputs with some loss of entropy [2]. Thus, FEs can be employed in different scenarios, where secret key storage is not available. Such scenarios include 1) constrained resources and limited power/energy consumption, 2) high-security level forbidding secret key storage, or 3) preserving users' privacy. Fig. 1 shows a generalization of FE schemes including two phases: 1) *generation* and 2) *reproduction*. The *generation* phase is usually executed once during manufacturing process and produces a pair of *challenge* and *response* from a biometric input such as w. On the other hand, the *reproduction* phase can regenerate the same *response* given the *helper* and biometric data from the same source as $w' \approx w$ (with some noise threshold).

The security of traditional FE schemes such as [2], [6]–[8] is based on information-theoretic setting, which results in large amount of entropy loss [5]. Therefore, they require large *helper* data and assume different characteristics on the biometric source such as *independent and identically distributed* (*i.i.d*) data [9]. In 2013, Fuller et al. [9] proposed the first computational fuzzy extractor (CFE) based on learning with errors (LWE) problem [10]. The CFE benefits from provable security against classic and quantum attacks due to the underlying LWE problem [9], [11]. CFEs do not have the limitations of traditional FEs and can be employed on a wider range of biometric sources by significantly decreasing the *helper* data size.

Herder et al. [12] showed that one can reliably build an FE based on learning parity with noise (LPN) problem. The LPN-based FE, hereafter referred to as LPN_FE, has higher error-tolerance compared to previous FE schemes according to [12]. In [13], the only practical implementation of the LPN_FE was proposed using hardware/software (HW/SW) co-design. However, the proposed architecture in [13] suffers

from extreme resource consumption, which is not practical for IoT end-nodes.

This paper proposes a new lightweight HW/SW co-design architecture for implementing Herder's LPN_FE targeting different resource-constrained devices in IoT. Our implementations consume low resources in hardware and are suitable for off-the-shelf tiny SoC-FPGA boards, such as PicoZed [14], Cora Z7 [15], ZynqBerry [16], and Z-turn Lite [17]. In comparison with implementations from the previous work [13], the proposed architecture improves area and *area-time* product (AT) by 89% and 83%, respectively, on hardware.

The proposed architecture has constant-time operations and is resistant against timing attacks [18]. Furthermore, our simple power analysis (SPA) [19] results show that the proposed method is also secure against SPA. The proposed architecture is designed independently from platform and can also be implemented on the application-specific integrated circuit (ASIC). This is more suitable for IoT applications because of the relatively lower area and power consumption compared to FPGAs. Our ASIC implementation analysis of the proposed architecture shows low power (less than 2.2 mW) and energy (less than 14.5 μJ) consumption. These results indicate that the proposed architecture can operate by lightweight energy harvesting techniques according to the latest national institution for standards and technology (NIST) [20] criteria [21].

The main contributions of this paper are as follows:

- We propose a lightweight architecture for the implementation of LPN_FE using HW/SW co-design. Due to the low resource requirements of the proposed architecture, it can be efficiently implemented on the most lightweight off-the-shelf SoC-FPGA boards such as Digilent's Cora Z7 [15].
- Our SoC-FPGA implementation improves area and AT more than 89% and 83%, respectively, compared to the previous work.
- To the best of our knowledge, this work is the first to implement LPN_FE on the ASIC platform using HW/SW co-design targeting low-end IoT devices. The ASIC platform implementation results indicate that the proposed architecture has low power and energy consumption based on the NIST's lightweight crypto criteria [20], [21].
- The implementation of the proposed architecture is resistant against simple side-channel analysis attacks such as timing [18] and SPA [19].

The rest of the paper is organized as follows. Section II provides the necessary background to follow the rest of the paper. The scheme optimizations, proposed architecture, and security analysis are detailed in Section III. Section IV provides detailed implementation results. Finally, the paper is concluded in Section V.

II. Background and Related Work

We begin this section by formally defining LPN problem, its applications and hardness. Later, we provide details of Herder's LPN-based FE [12]. Finally, the only implementation of LPN_FE from [13] along with other related work are discussed.

A. Learning Parity with Noise (LPN) Problem

LPN is a subset of more well-known LWE problem, where all error and main vectors are sampled over $\mathbb{Z}_2^n = \{0,1\}^n$. Similar to LWE, the LPN problem has two versions: 1) *search* and 2) *decision*. The *search* version requires to find the secret vector, given polynomially many LPN samples from the same secret vector. On the other hand, the *decision* version distinguishes between two sets of LPN and uniformly random samples. In the following, we formally define the *search* version of the LPN problem according to [22].

Definition 1. Given uniformly random $s \in \{0,1\}^n$ and $A \in \{0,1\}^{m \times n}$, one can compute $b_i = \langle A.s \rangle \oplus e_i \in \{0,1\}^m$, where $e_i \leftarrow \chi$ are sampled from a Bernoulli distribution. The problem is finding s, while having polynomially many (b_i, A) pairs.

It is known that no algorithm can solve LPN problem efficiently in polynomial time for large enough e_i error vectors [23].

B. Fuzzy Extractor Based on LPN

In 2016, Herder et al. proposed a new CFE based on the LPN problem (a subset of LWE) and showed that it could be efficiently secure for a specific set of parameters [12]. According to [12], the LPN-based FE has the lowest entropy-loss among all FE schemes. This feature makes it more suitable for non-perfect natural biometric sources, i. e., human Iris, compared to previous FE schemes. They further proposed using additional *confidence information* from the biometric source to achieve higher noise-tolerance and polynomial-time key recovery. It is shown that the *confidence information* does not require to be private and can be publicly revealed without compromising the security of the scheme [13]. In the remaining of the section, we provide an overview of the LPN_FE scheme [12].

As same as other FE schemes, Herder's LPN_FE consists of two main phases: 1) *Generation* and 2) *Reproduction*. The *generation* phase is executed only once per each biometric data. Therefore, it is considered to be executed during the manufacturing/initialization process. Hence, no performance evaluations or limited resources are usually deemed during the *generation* phase.

On the other hand, the *reproduction* process extracts related keys to the biometric data repeatedly on device-ends [12], [13]. Therefore, it is necessary to focus on the efficient implementation of the *reproduction* phase on resource-constrained devices in the IoT network. The details of Herder's LPN_FE are as follows:

1) ***Generation phase****:* Fig. 2 shows the overall dataflow of the *generation* phase in LPN_FE. As mentioned earlier, this phase is only executed once per biometric data during the manufacturing process. The process starts with capturing biometric data (m bits) and selecting the most stable bits based on the data's confidence levels. Signal I indicates the indices of n selected bits (from m-bit biometric). Moreover, a completely random matrix ($A_{m \times n}$) is created by a true random generator (TRNG) module. Then, n corresponding rows of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3050555, IEEE Internet of Things Journal

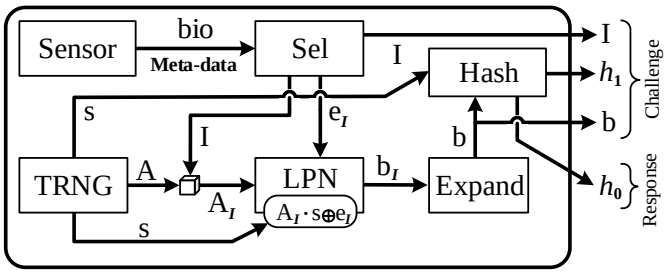

Fig. 2. LPN_FE scheme [12]: Generation Phase.

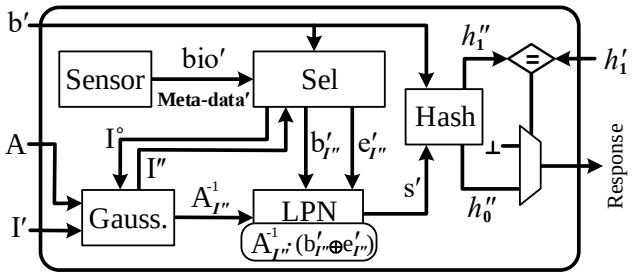

Fig. 3. LPN_FE scheme [12]: Reproduction Phase.

matrix A are chosen based on the index vector I to create a A_I square matrix with $n \times n$ dimensions. Note that matrix A is not private and can be exposed to the public without compromising the security [12].

In addition, an n-bit secret vector s is also generated by the TRNG module. The main goal of the scheme is to hide secret s by employing the known matrix A and the biometric vector e. The next operation is to calculate the *helper* data $b_I = \langle A_I \cdot s \rangle \oplus e_I$, which is cryptographically random and leaks no information about the secret s and the biometric vector e_I [12]. Note that the b_I has only n bits and is required to be expanded to m bits by inserting 0s based on the index vector I. Finally, to complete the secure sketch and generate a pair of *challenge* and *response*, *helper* data b is hashed using s as the secret key. Together, I, $h_1 = H(b|1, s)$ and b construct the *challenge*, while $h_0 = H(b|0, s)$ itself acts as the corresponding *response*.

2) ***Reproduction phase****:* This phase aims to reproduce the same *response*, given the *challenge* and the biometric data, as is shown in Fig. 3. The process starts with selecting the most stable bits of biometric data. Note that the index vector I° is most likely to be different in some parts from the given I'. Based on the given I', the matrix A and the calculated I°, one must perform a Gaussian elimination to find I'' with the highest statistical success rate compared to other possibilities. Besides, the matrix $A_{I''}$ must be invertible; otherwise, the Gaussian elimination should be repeated. Finally, the matrix $A^{-1}_{I''}$ is given to the LPN unit to calculate $s' = \langle A^{-1}_{I''} \cdot (b'_{I''} \oplus e'_{I''}) \rangle$. Now the scheme can reproduce correct response $h_0'' = h_0$ if-and-only-if the calculated s' matches the random s employed in the *generation* phase. This condition is verified based on the equality of $h_1'' = H(b'|1, s')$ and the given $h_1' = H(b|1, s)$. Note that if an adversary tries to expose the system with a wrong *challenge*, the condition of $h_1'' = h_1'$ will fail due to the hardness of the underlying hash function H [12].

C. Related Work

Since the introduction of FEs in 2004, various studies have aimed to implement different FE schemes efficiently. As mentioned earlier, the most common application of FE schemes is to correct the noisy responses of PUFs. In [24] and [25], PUFs are employed in FPGAs for providing different classes of intellectual property (IP) authentication in the hardware. Authors in [26] propose a new and efficient FE for biased PUFs based on the ternary debiasing method. Another study targetting efficient PUF-based authentication can be found in [27] by sacrificing the number of correctable keys to enhance the security of the PUF significantly. In [28], a compact and low-power FE is proposed for PUF-based authentication.

The main related work of this paper is [13], where a HW/SW co-design was proposed for implementing LPN_FE. The authors in [13] perform the Gaussian elimination process in software, while the rest of the calculations are done in hardware. However, as stated in [13], off-loading such calculations to software can comprise the system's security, since software-based attacks can easily break the system integrity and exploit biometric source [3], [4], [29]. For example, a malformed $A^{-1}_{I''}$ can help the attacker to extract secret biometric source in only 128 tries [13]. To ensure the correctness of given $A^{-1}_{I''}$ and A matrices from the software-side, the authors in [13] propose a verification process as follows:

- **Verification of A:** In this phase, the hardware gets matrix A row by row and feeds it into a SHA-256 module for verification against the pre-stored digestion of matrix A in the hardware. Moreover, according to vector I'', it stores the complete matrix $A_{I''}$ in the hardware; This requires $16,384 = 128 \times 128$ bits of registers.
- **Verification of $A^{-1}_{I''}$:** The hardware calculates multiplication of given $A^{-1}_{I''}$ by $A_{I''}$ to verify that the result is equal to the identity matrix $I_{128\times128}$. Besides, matrix $A^{-1}_{I''}$ is also stored in hardware for performing LPN computations, which requires an additional $16,384 = 128 \times 128$ bits of registers.

The authors in [13] claim that the inverse of the matrix $A_{I''}$ is unique and therefore, if matrix A is verified and $A^{-1}_{I''} \times A_{I''}$ results in identity matrix $I_{128\times128}$, then matrix $A^{-1}_{I''}$ is also verified.

III. Proposed Architecture

We employ the same parameter selection as [12], [13] by setting $m = 450$ and $n = 128$ to achieve 128 bits of security provided by the underlying LPN problem that is sufficient for IoT criteria announced by NIST [20]. In this section, we propose an optimized verification process for evaluating inputs from the software-side in the hardware. Later, we provide details of the proposed lightweight HW/SW co-design architecture for the LPN_FE scheme. Finally, we discuss the proposed architecture's resistance against simple side-channel analysis, such as timing and SPA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3050555, IEEE Internet of Things Journal

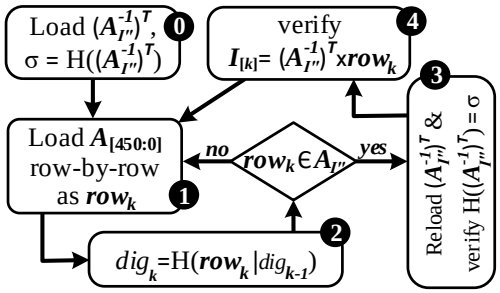

Fig. 4. The Proposed Verification Process of Inputs in Hardware.

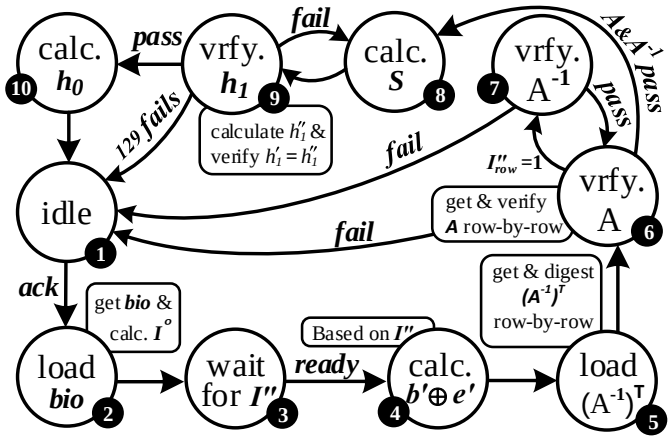

Fig. 5. Finite State Machine (FSM) of the Proposed Hardware for *Reproduction* Phase.

A. Optimizing Verification Process

As mentioned earlier, only *reproduction* phase is executed on the device-end and, therefore, requires to be implemented efficiently. Similar to the previous work [13], we have performed the Gaussian elimination in the software-side. Apparently, this side appears to be vulnerable to different attacks [3], [4], [13], [29]–[32]. Therefore, every input from the software side must be verified in hardware before employing in final calculations. However, ensuring the correctness of inputs (given $A_{I''}^{-1}$ and A matrices as discussed in Section II-C) in hardware has significant memory and performance overhead itself. According to [13], a complete multiplication of $I = A_{I''}^{-1} \times A_{I''}$ is required to ensure the correctness of given $A_{I''}^{-1}$ with more than 32k bits of registers to store $A_{I''}^{-1}$ and $A_{I''}$ matrices.

In this paper, instead of storing $A_{I''}$ and $A_{I''}^{-1}$ matrices, we verify them on the fly and propose a verification process that forces the software-side to provide the correct inputs whenever required by the hardware. Fig. 4 presents the overall verification flow of the proposed architecture. First, the hardware loads $A_{I''}^{-1}$ matrix in a transposed manner (i.e., $(A_{I''}^{-1})^T$) row by row and feeds each row to the SHA-256 module. After exactly 128 clock cycles, the hardware calculates the complete digestion of the matrix $(A_{I''}^{-1})^T$ as σ. Note that the hardware does not store any rows of the matrix and verifies the given matrices' correctness by comparing their digestions against the calculated σ.

Next, hardware starts to load matrix A row by row and feeds each row to the SHA-256 module to calculate complete digestion of A and compare it against pre-stored digestion from the manufacturing phase [13]. According to the indexing vector I'' (obtained from the software), the hardware can detect rows of the matrix A, which are included in $A_{I''}$. Upon loading every row of $A_{I''}$ such as row_k, the hardware immediately starts to multiply it by the entire $A_{I''}^{-1}$ matrix and verifies whether the result is equal to the corresponding row of the *Identity* matrix. As mentioned earlier, the hardware does not store the $A_{I''}^{-1}$ matrix but has calculated its complete digestion. During the multiplication phase, the software side sends the same $(A_{I''}^{-1})^T$ matrix to the hardware to perform multiplication of the row_k to $(A_{I''}^{-1})^T$ matrix in exactly 128 clock cycles. At the same time, the hardware compares the digestion of the given $(A_{I''}^{-1})^T$ matrix against previously calculated σ.

Finally, after 128 times multiplication of all $A_{I''}$ rows by the matrix $A_{I''}^{-1}$, the verification of the matrix $A_{I''}^{-1}$ is completed by comparing the results against $Identity_{128\times128}$ matrix [13]. Moreover, after loading all 450 rows of the matrix A, the hardware has calculated the complete digestion of the matrix A (using SHA-256 module) and can compare it against pre-stored digestion from *manufacturing* phase. If the verification of the matrix A is successful and the multiplication of $A_{I''}$ by $A_{I''}^{-1}$ is equal to the *identity* matrix $I_{128\times128}$, then matrix $A_{I''}^{-1}$ is considered to be verified [13].

B. Architecture Overview

Fig. 5 and Fig. 6 show the finite state machine (FSM) and the proposed architecture of the hardware, respectively. The process starts with loading biometric data and calculating I° at the same time (state 2 in Fig. 5 and *bio_selector* module in Fig. 6), which is done by selecting the biometric values with the highest confidence levels [12]. After sending I° to the processor (software side), the hardware waits for the final I'' to be calculated using Gaussian elimination. The software notifies the hardware by triggering the *ready* signal. Hardware immediately starts calculating $b'_{I''} \oplus e'_{I''}$ in exactly 450 clock cycles according to I'' (state 4 in Fig. 5 and $b'\_e'_{[127:0]}$ register in Fig. 6). It is worth mentioning that no pre-calculation is possible during the *waiting* phase since the hardware has no information regarding I''.

Hardware stores $b'_{I''} \oplus e'_{I''}$ vector in a 128-bit register and starts the verification process of the inputs by loading matrix $(A_{I''}^{-1})^T$ row by row. The complete digestion of the matrix $(A_{I''}^{-1})^T$ with SHA-256 will be available in exactly 128 clock cycles (state 5 in Fig. 5 and *Hash* A^{-1} register in Fig. 6). After this state, the hardware can ensure the integrity of the given $(A_{I''}^{-1})^T$ matrices during the process of *re-production* by verifying them against this digestion. Furthermore, the hardware starts to load matrix A row by row without storing it (state 6 in Fig. 5). Each row of the given matrix A is fed to the SHA-256 module to be included in the final digestion, which will be available after receiving all 450 rows of matrix A. The digestion will be evaluated against a pre-stored one in the hardware (during the *manufacturing* phase [13], the *Pre-Stored* register in Fig. 6). In addition to verification process of matrix A, the matrix $(A_{I''}^{-1})^T$ also requires to be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3050555, IEEE Internet of Things Journal

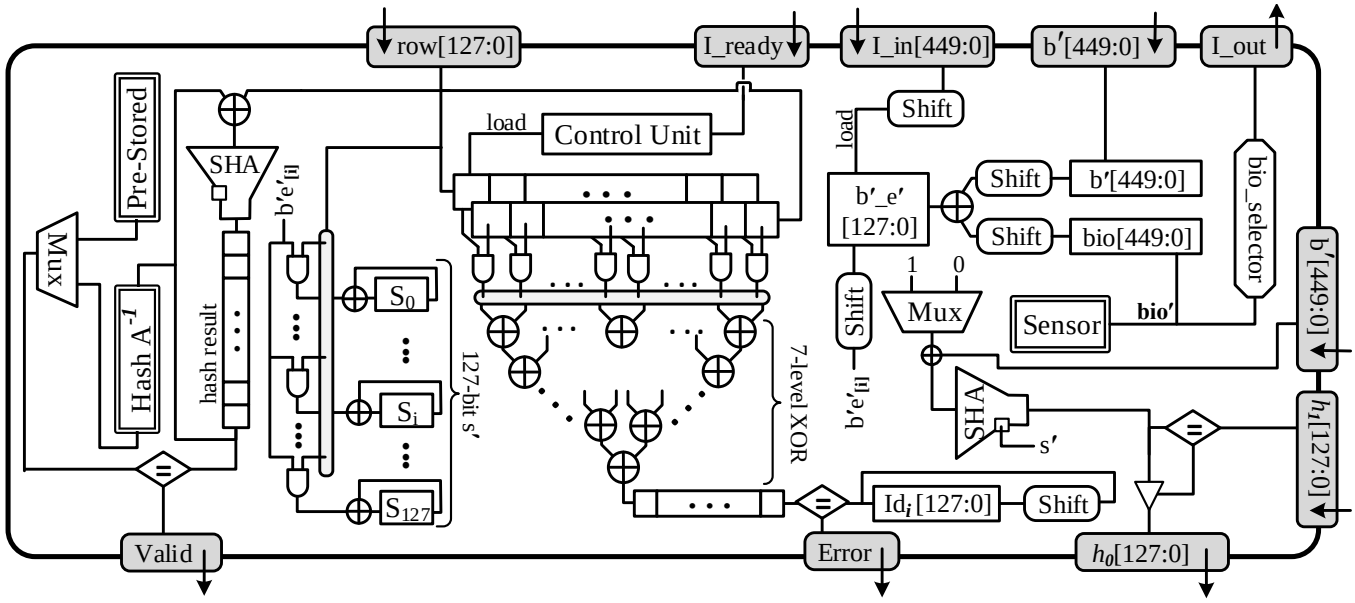

Fig. 6. Proposed HW/SW Co-designed Architecture.

verified simultaneously (refer to Section III-A for details). This verification is performed using multiplying $(A_{I''}^{-1})^T$ by $A_{I''}$, which must result in the *identity matrix* $I_{128\times128}$ (state 7 in Fig. 5).

In order to minimize the implementation costs of accesses to matrix elements in the multiplication, one of the matrices has to be available in the *transposed* manner. This way, the hardware can select corresponding columns instead of rows with lower complexity. The hardware starts re-loading the entire matrix $(A_{I''}^{-1})^T$ from the software side and performs a multiplication of the given row of the matrix $A_{I''}$ by the matrix $(A_{I''}^{-1})^T$. To convolve two 128-bit vectors in one clock cycle, as can be seen in Fig. 6, first, a 128 bitwise-AND is performed. Then, the modulo-2 addition of the resulting 128-bit vector is performed using a 7-level XOR tree. The final result of entire $(A_{I''}^{-1})^T$ matrix multiplication by one row of $A_{I''}$ is available after 128 clock cycles. The hardware verifies whether the resulting vector matches the corresponding row of the *identity* matrix (state 4 of the Fig. 4 and a circular shift-register $Id_i[127:0]$ in Fig. 6). Note that the hardware ensures the integrity of the re-loaded $(A_{I''}^{-1})^T$ by verifying its digestion against the pre-calculated one from the state 5 of the Fig. 5 (the very left SHA module and the *valid* output signal in Fig. 6).

After successful verification of $A_{I''}^{-1}$ and A matrices, the hardware starts to calculate $s'_{[127:0]} = \langle A_{I''}^{-1}.(b'_{I''} \oplus e'_{I''})\rangle$ (state 8 in Fig. 5). It is worth mentioning that the hardware can only verify the integrity of $A_{I''}^{-1}$ matrix in *transposed* manner (i.e. $(A_{I''}^{-1})^T$). Therefore, the same 7-level XOR tree cannot be employed. This is due to the fact that the $A_{I''}^{-1}$ matrix is now accessible column-by-column. Algorithm 1 presents the process of calculating $s'_{[127:0]} = \langle A_{I''}^{-1}.(b'_{I''} \oplus e'_{I''})\rangle$ when $(A_{I''}^{-1})^T$ is available instead of $A_{I''}^{-1}$. To this end, the hardware selects each bit of $b'_{I''} \oplus e'_{I''}$ vector per cycle (named $b'e'_{[i]}$ in Fig. 6). During each clock cycle, hardware performs AND operation of $b'e'_{[i]}$ bit with every bit in the given row of matrix $(A_{I''}^{-1})^T$. The results are XORed with previous values of the s' register. It is worth mentioning that the second loop of the Algorithm 1 (line 5) is executed in one clock cycle in the proposed architecture by employing 128 sets of 2-input AND and XOR gates. Therefore, the entire calculation of $s'_{[127:0]} = \langle A_{I''}^{-1}.(b'_{I''} \oplus e'_{I''})\rangle$ is completed in exactly 128 clock cylces.

Algorithm 1: Calulating s' by $(A_{I''}^{-1})^T$

Data: $(A_{I''}^{-1})^T$ and $b'_{I''} \oplus e'_{I''}$
Result: $s'_{[127:0]} = A_{I''}^{-1}.(b'_{I''} \oplus e'_{I''})$

1 **begin**
2 $s'_{[127:0]} \leftarrow 0$
3 **for** $i : 0 \rightarrow 127$ **begin**
4 $col \leftarrow (A_{I''}^{-1})^T_{[i]}$
5 **for** $k : 0 \rightarrow 127$ **begin**
6 $s'_{[k]} \leftarrow s'_{[k]} \oplus (col_{[k]}\ \&\ (b'_{I''} \oplus e'_{I''})_{[k]})$
7 **end**
8 **end**
9 **return** $s'_{[127:0]}$
10 **end**

Furthermore, the hardware verifies the calculated s' by evaluating $h''_1 = H(b'_{I''}|1, s')$ against given h'_1 (state 9 in Fig. 5 and the very right SHA module in Fig. 6). In case of failure in the verification process, the calculation of $s' = \langle A_{I''}^{-1}.(b'_{i''} \oplus e'_{i''})\rangle$ must be repeated with one bit-flip in $b'_{i''} \oplus e'_{i''}$ vector until it succeeds [12], [13]. There are exactly 128 possible cases of bit-flips in $b'_{i''} \oplus e'_{i''}$ vector. If none of these variations meet the criteria of $h''_1 == h'_1$, the hardware sends an error to the system; otherwise it proceeds to the next state (state 10 in Fig. 5) and returns $h''_0 = H(b'_{I''}|0, s')$ as the *response* to the given *challenge*.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3050555, IEEE Internet of Things Journal

TABLE I
NUMBER OF CLOCK CYCLES PER OPERATION

State†	2	4	5	6	7	8	9	10
Operation	load *bio* & calculate I°	calculate $b'_{I''} \oplus e'_{I''}$	load & digest matrix $(A_{I''}^{-1})^T$	load & verify matrix A	multiply & verify $Id_i = A_i \times A_{I''}^{-1}$	calculate s'	calculate h_1'' & verify $h_1'' = h_1'$	calculate h_0''
Clock Cycles	450	450	128 + SHA•	450 + SHA•	128 × (128+SHA)	128+SHA•	SHA•	SHA•

† States 1 and 3 are excluded due to the fact that the hardware does not perform any operations and waits for the software side.
• SHA stands for number of clock cycles required by a complete round of SHA-256 algorithm.

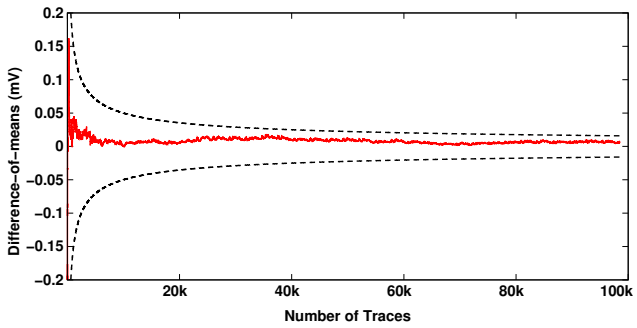

Fig. 7. Simple Power Analysis (SPA) of the Implementation.

C. Side-Channel Leakage Analysis

In this section, we evaluate the side-channel leakage of the proposed architecture and show it is resistant against SPA and timing attacks. This is mainly due to the fact that the hardware does not have any branches or path selections based on the secret data bits (biometric data e'). Table I presents clock cycle counts of each state (illustrated in Fig. 5) of the architecture. According to the table, as all states have constant number of clock cycles, the calculations are independent from the secret value. Note that it is necessary to verify A and $A_{I''}^{-1}$ matrices before calculating s' to prevent certain timing attacks on vector s' as discussed in [13]. Such characteristics make the proposed architecture secure against timing attacks.

Furthermore, for quantitative evaluation of the proposed hardware architecture's SPA resistance, a power analysis evaluation board, namely Sakura-X, is used. For high-accuracy power measurements, we have employed a *deep-memory high-performance* USB oscilloscope (PicoScope 6403D) that can sample power traces at 5 *GS/s* rate. We have performed the analysis in accordance with [33]. To execute SPA, we performed *difference-of-means* test for up to 100,000 random traces on $b'_{I''} \oplus e'_{I''}$ vector, while the first two bits are fixed to $\{10\}_2$. Fig. 7 presents the results of SPA evaluation of the proposed hardware architecture during 100,000 random traces. Note that the evaluations indicate that performing a successful SPA attack is not possible since the *difference-of-means* values do not cross the 99.99% confidence intervals of Pearson coefficients (the dashed lines).

IV. IMPLEMENTATION RESULTS

We have implemented the proposed architecture on both FPGA and ASIC platforms. In the following, we first provide details of our SoC-FPGA implementation results and evaluate them against previous work in terms of area, speed, and AT. Moreover, we analyze the results of our ASIC implementation using the 45nm Nangate standard cell library [36]. To the best of our knowledge, this is the first implementation of the computational fuzzy extractor on the ASIC platform.

A. FPGA Implementation

For reconfigurable hardware implementation, we considered different Xilinx family FPGAs, such as Artix, Kintex, and UltraScale. Moreover, Xilinx's Vivado development software is employed to report synthesis analysis of our implementations, including 1) number of lookup tables (LUTs), 2) number of flip-flops (FFs), 3) number of occupied slices (equal to LUT/FF pairs), and 4) overall frequency. Table II provides detailed synthesis results of the proposed architecture on FPGA.

The first implementation of Herder's LPN_FE scheme was proposed in [13], where the authors employed a Xilinx SoC-FPGA ZedBoard [34]. The board contains an Artix-7 XC72020 FPGA and an ARM Cortex-A9 processor. As shown in Table II, our implementation on the same device improves the area by more than 89.4% compared to [13]. However, due to the more complex verification process of the proposed method, the *reproduction* phase of this work is about 36 µ*s* slower compared to the previous work. Overall, the proposed method obtains more than 83% higher AT compared to the previous work [13].

The proposed method has low hardware complexity and, therefore, is suitable for resource-constrained SoC-FPGA boards that are designed for IoT. To this end, we provide implementation results of the proposed architecture on a wide range of Xilinx FPGAs such as Artix, Kintex, and UltraScale+ in tiny SoC-FPGA boards such as PicoZed [14], ZynqBerry [16] and UltraZed [35]. We also included Digilent's Cora Z7 [15] and MYiR's Z-Turn Lite SoC-FPGA boards that employ the same FPGA architecture (Artix-7) as ZedBoard. In contrast to ZedBord, Cora Z7 and Z-Turn Lite cost only around 99$ and have a limited number of 14k LUTs. The details of the available LUT and Flip-Flops in each device are presented in Table II. It is worth mentioning that previous work [13] cannot fit into such resource-constrained SoC-FPGA boards due to the high amount of LUT and Flip-Flop consumptions.

B. ASIC Implementation

The proposed architecture is designed independently from the platform. Therefore, we have managed to implement the architecture on an ASIC platform, which is more suitable for

TABLE II
COMPARISON OF FPGA IMPLEMENTATION RESULTS

scheme	SoC-FPGA Board				LUT	FF	Slice	CCs	Freq. (MHz)	Time (µs)	A×T$^{\diamond}$	Improvement$^{\mp}$	
	name	FPGA	MAX LUT/FF$^{\triangle}$	IoT cmp.$^{\bullet}$								Area	A×T
Jin et al.†[13] LPN core	Zedboard [34]	Artix-7	52k/100k	✗	28.5k	38k	>7k	-	-	68.20	477k		
Proposed method	Zedboard [34]	Artix-7	52k/100k	✗	2438	2730	745	26k	254	104.1	77.5k	89.4%	83.8%
	Cora Z7 [15]		14k/10k	✔									
	Z-turn Lite [17]		14k/28k	✔									
	ZynqBerry [16]		14k/28k	✔									
	PicoZed [14] 7030 Model	Kintex-7	14k/15k	✔	2438	2730	745	26k	254	104.1	77.5k		
	UltraZed [35]	UltraScale+	12k/20k	✔	2286	2647	361	26k	270	98.60	35.6k		

$^{\triangle}$ The maximum number of LUT/FF that device provides.
$^{\bullet}$ Indicates the compatibility of the SoC-FPGA board with IoT applications based on the dimensions and power consumption of the board.
$^{\diamond}$ A×T = number of Slices × time in microseconds. † Reported results are only for LPN core.
$^{\mp}$ To have a fair comparison, only improvements on the same FPGA architecture are reported. Note that improvements on the other FPGA architectures are higher. Improvement = $\frac{x-y}{x} \times 100$, where x and y are related to previous and the proposed method, respectively.

TABLE III
ASIC IMPLEMENTATION RESULT OF THE PROPOSED METHOD ON THE 45NM NANGATE STANDARD CELL LIBRARY

Freq. (MHz)	Area		Power$^{\bullet}$ (mW)		Time (ms)	Energy (µJ)
	µm^2	GE†	static	dynamic		
1	23.5k	29.5k	0.527	0.016	26.63	14.460
10	23.5k	29.5k	0.527	0.162	2.66	1.832
100	23.5k	29.5k	0.527	1.628	0.266	0.573

$^{\bullet}$ The power reported in the paper is calculated as "total power = static power + dynamic power". Dynamic power has almost linear relation with the frequency.
† GE stands for *gate equivalent* of the circuit in terms of nand-based implementation.

IoT end-nodes. To the best of our knowledge, we are the first to implement a CFE on the ASIC platform using HW/SW co-design. Table III shows details of the implemented architecture on ASIC using 45nm NanGate standard cell library [36].

The synthesized circuit consumes an area of $23.5\,\mu m^2$, and can be easily integrated with different SoC boards. For applications with low frequency, the dominant portion of the power consumption of the device is static. This indicates that the device's activity is low, which makes power side-channel analysis attacks even harder. Overall, the power consumption of the circuit varies in range of 0.53 mW (low frequency applications) to 2.2 mW (high frequency applications). The execution time of the *reproduction* phase for the low-frequency applications is 26.63 ms, which is still fast for an ultra-low-power lightweight node in IoT. Moreover, such execution time for the high-frequency applications is 100 times faster and takes 0.266 ms. Due to the low area consumption of the circuit, higher frequencies may result in higher power consumption, which can create a hot spot in the circuit.

According to recent NIST reports on lightweight cryptography [20], [21], the ASIC implementations of the proposed architecture can work with low-power energy harvesting units such as Vibration Piezo/electro-magnetic (EM) [37] and solar-based [21]. As a result of low-power circuit and high-speed operations, the circuit's overall energy consumption is also relatively low (from 0.5 to 14.5 µJ). Therefore, battery-based solutions can exploit the proposed LPN_FE architecture.

V. CONCLUSION

In this paper, we propose a lightweight architecture for LPN_based CFE using HW/SW co-design. The architecture is suitable for implementation on off-the-shelf resource-constrained SoC-FPGA boards, such as Cora Z7 [15]. The proposed architecture's implementation results improve area and AT compared to previous work by more than 89% and 83%, respectively. Moreover, the proposed architecture's hardware implementation executes in constant time and the number of operations is independent of secret input values (biometric data). This feature makes the proposed architecture to be resistant against different simple side-channel attacks such as timing [18] and SPA [19].

The ASIC implementation results, using the 45nm Nangate standard cell library [36], indicate that the proposed architecture requires only 0.266 µs to perform a complete round of *reproduction* phase when clocked at 100 MHz. According to the latest NIST reports regarding lightweight cryptography [20], our ASIC implementations can operate using ultra-low-power energy harvesting mechanics such as EM [37] and Solar-based [21]. To the best of our knowledge, this paper is the first to implement a CFE on the ASIC platform using HW/SW co-design.

Biometric authentication based on CFE schemes provides a better opportunity to preserve users' privacy. One of the remaining challenges in this field is the physical security of user biometrics during computations. The proposed architecture in this paper is prone to timing attack and SPA. However, an advanced differential power analysis (DPA) [38] or fault injection attack [39] may easily break the system. To this end,

one of the main future work of this study is to provide a secure implementation against such attacks.

REFERENCES

[1] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, and X. Fu, "Security vulnerabilities of internet of things: A case study of the smart plug system," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1899–1909, Dec 2017.

[2] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 523–540.

[3] M. B. Healy, H. C. Hunter, C. A. Kilmer, K.-h. Kim, and W. E. Maule, "Detecting a cryogenic attack on a memory device with embedded error correction," Apr. 10 2018, uS Patent 9,940,457.

[4] F. Cherpantier, "Tamper reactive memory device to secure data from tamper attacks," May 17 2011, uS Patent 7,945,792.

[5] G. Itkis, V. Chandar, B. W. Fuller, J. P. Campbell, and R. K. Cunningham, "Iris biometric security challenges and possible solutions: For your eyes only? using the iris as a key," *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 42–53, 2015.

[6] M. Hiller, D. Merli, F. Stumpf, and G. Sigl, "Complementary ibs: Application specific error correction for pufs," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*. IEEE, 2012, pp. 1–6.

[7] M. Hiller, M. Weiner, L. Rodrigues Lima, M. Birkner, and G. Sigl, "Breaking through fixed puf block limitations with differential sequence coding and convolutional codes," in *Proceedings of the 3rd international workshop on Trustworthy embedded devices*. ACM, 2013, pp. 43–54.

[8] M.-D. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[9] B. Fuller, X. Meng, and L. Reyzin, "Computational fuzzy extractors," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2013, pp. 174–193.

[10] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[11] S. Bayat-Sarmadi, S. Ebrahimi, and H. M. Boorani, "Quantum-resistant cryptoprocessing," Jun. 25 2020, uS Patent App. 16/807,394.

[12] C. Herder, L. Ren, M. van Dijk, M.-D. Yu, and S. Devadas, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 65–82, 2016.

[13] C. Jin, C. Herder, L. Ren, P. Nguyen, B. Fuller, S. Devadas, and M. van Dijk, "Fpga implementation of a cryptographically-secure puf based on learning parity with noise," *Cryptography*, vol. 1, no. 3, p. 23, 2017.

[14] "Picozed," [Online]. Available: http://zedboard.org/product/picozed.

[15] "Cora z7," [Online]. Available: https://store.digilentinc.com/cora-z7-zynq-7000-single-core-and-dual-core-options-for-arm-fpga-soc-development/.

[16] "Soc module with xilinx zynq z-7012s," [Online]. Available: https://shop.trenz-electronic.de/en/Products/Trenz-Electronic/TE07XX-Zynq-SoC/TE0726-Zynq-SoC.

[17] "Z-turn lite," [Online]. Available: http://www.myirtech.com/list.asp?id=565.

[18] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 104–113.

[19] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, 2011.

[20] "National institute of standards and technology." [Online]. Available: https://www.nist.gov/.

[21] C. Patrick and P. Schaumont, "The role of energy in the lightweight cryptographic profile," in *NIST Lightweight Cryptography Workshop*, 2016.

[22] N. J. Hopper and M. Blum, "Secure human identification protocols," in *International conference on the theory and application of cryptology and information security*. Springer, 2001, pp. 52–66.

[23] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[24] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2007, pp. 63–80.

[25] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly puf protecting ip on every fpga," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*. IEEE, 2008, pp. 67–70.

[26] M. Suzuki, R. Ueno, N. Homma, and T. Aoki, "Efficient fuzzy extractors based on ternary debiasing method for biased physically unclonable functions," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 616–629, 2018.

[27] Y. Wen and Y. Lao, "Efficient fuzzy extractor implementations for puf based authentication," in *2017 12th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2017, pp. 119–125.

[28] A. Aysu, E. Gulcan, D. Moriyama, and P. Schaumont, "Compact and low-power asip design for lightweight puf-based authentication protocols," *IET Information Security*, vol. 10, no. 5, pp. 232–241, 2016.

[29] S. Ebrahimi and S. Bayat-Sarmadi, "Lightweight and fault resilient implementations of binary ring-lwe for iot devices," *IEEE Internet of Things Journal*, 2020.

[30] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher *et al.*, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1–19.

[31] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure:{SGX} cache attacks are practical," in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[32] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer. js: A remote software-induced fault attack in javascript," in *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 2016, pp. 300–321.

[33] A. Aysu, M. Orshansky, and M. Tiwari, "Binary ring-lwe hardware with power side-channel countermeasures," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1253–1258.

[34] "Xilinx zedboard zynq soc-fpga," [Online]. Available: https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html.

[35] "Ultrazed-eg," [Online]. Available: http://zedboard.org/product/ultrazed-EG.

[36] "NanGate standard cell library." [Online]. Available: http://www.si2.org/.

[37] P. D. Mitcheson, E. M. Yeatman, G. K. Rao, A. S. Holmes, and T. C. Green, "Energy harvesting from human and machine motion for wireless electronic devices," *Proceedings of the IEEE*, vol. 96, no. 9, pp. 1457–1486, 2008.

[38] S. Ebrahimi and S. Bayat-Sarmadi, "Lightweight and dpa-resistant post-quantum cryptoprocessor based on binary ring-lwe," in *2020 20th International Symposium on Computer Architecture and Digital Systems (CADS)*. IEEE, 2020, pp. 1–6.

[39] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75–82, 1997.

Shahriar Ebrahimi received his B.Sc. degree in information technology from Sharif University of Technology (SUT), Tehran, Iran, in 2014, and the M.Sc. degree in computer engineering from the same university in 2016. Currently, He is a Ph.D Candidate in computer engineering at the Computer Engineering Department of SUT. His research interests include emerging technologies in IoT, network security and architecture, cryptographic engineering, post-quantum and lattice-based cryptography.

Siavash Bayat-Sarmadi received the B.Sc. degree from the University of Tehran, Iran, in 2000, the M.Sc. degree from Sharif University of Technology, Tehran, Iran, in 2002, and the Ph.D degree from the University of Waterloo in 2007, all in computer engineering (hardware). He was with Advanced Micro Devices, Inc. for about six years. Since September 2013, he has been a faculty member in the Department of Computer Engineering, Sharif University of Technology, where he is currently a tenured associate professor. His research interests include hardware security and trust, cryptographic computations, secure and dependable computing and architectures. He is a member of the IEEE.