Received: 1 December 2021 | Revised: 24 June 2022 | Accepted: 6 July 2022
DOI: 10.4218/etrij.2021-0455

ORIGINAL ARTICLE

Implementation and characterization of flash-based hardware security primitives for cryptographic key generation

Mi-Kyung Oh[1] | **Sangjae Lee**[1] | **Yousung Kang**[2] | **Dooho Choi**[3]

[1]Radio & Satellite Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea
[2]Information Security Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea
[3]Department of AI Cyber Security/College of Science & Technology, Korea University Sejong, Sejong, Republic of Korea

Correspondence
Dooho Choi, Department of AI Cyber Security/College of Science & Technology, Korea University Sejong, Sejong, Republic of Korea.
Email: doohochoi@korea.ac.kr

Funding information
This research was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2018-0-00230, [TrusThingz Project]). This research was supported by the Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF) and the Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science and ICT, Republic of Korea (No. 2020M3C1C1A01084523)

Abstract
Hardware security primitives, also known as physical unclonable functions (PUFs), perform innovative roles to extract the randomness unique to specific hardware. This paper proposes a novel hardware security primitive using a commercial off-the-shelf flash memory chip that is an intrinsic part of most commercial Internet of Things (IoT) devices. First, we define a hardware security source model to describe a hardware-based fixed random bit generator for use in security applications, such as cryptographic key generation. Then, we propose a hardware security primitive with flash memory by exploiting the variability of tunneling electrons in the floating gate. In accordance with the requirements for robustness against the environment, timing variations, and random errors, we developed an adaptive extraction algorithm for the flash PUF. Experimental results show that the proposed flash PUF successfully generates a fixed random response, where the uniqueness is 49.1%, steadiness is 3.8%, uniformity is 50.2%, and min-entropy per bit is 0.87. Thus, our approach can be applied to security applications with reliability and satisfy high-entropy requirements, such as cryptographic key generation for IoT devices.

KEYWORDS
cryptographic key, entropy, flash memory, hardware security primitive, physical unclonable function (PUF)

This is an Open Access article distributed under the term of Korea Open Government License (KOGL) Type 4: Source Indication + Commercial Use Prohibition + Change Prohibition (http://www.kogl.or.kr/info/licenseTypeEn.do).
1225-6463/$ © 2022 ETRI

1 | INTRODUCTION

Large-scale Internet of Things (IoT) devices are increasingly being connected to the Internet to provide various services beyond the state-of-the-art, such as smart healthcare, industrial control, consumer electronics, and drone-based safety surveillance and agriculture systems [1–3]. Accordingly, the number of points where an attacker can penetrate the network through the most vulnerable IoT device is increasing. Therefore, even for resource-constrained IoT devices, it is necessary to ensure strong security that matches the required security level of the entire connected network. A conventional security mechanism provides a manufacturer-generated cryptographic key that is programmed into the device, which is unique to every device. However, as the number of IoT devices increases exponentially (expected to reach 82 billion in 2025), programming a different key into every device will become an unreasonably costly, time-consuming, or even impossible process. Therefore, cost-effective and easily applied security solutions for large-scale IoT devices are required [4, 5].

A physical unclonable function (PUF), which is unpredictable, unclonable, and unique to a specific device or chip, is considered a fundamental hardware security primitive, especially for resource-constrained IoT devices [1, 6, 7]. A PUF generates unique values that cannot be physically replicated due to variations in microstructures that are randomly generated, even via the same manufacturing process. The PUF is generated and used when needed and disappears without being saved to memory or a file [8, 9]. Through error correction and entropy amplification of the PUF output, it can be widely used for security applications such as device-specific root key, identity, and authentication credentials [10–12].

Various PUF systems have been presented, including ring oscillator (RO) PUF [13, 14], arbiter PUF [15, 16], and memory-based PUF [17–19]. Most types of PUFs required additional circuitry dedicated to the device, thus increasing hardware complexity, which in turn makes it difficult to apply to resource-constrained IoT devices. The prominent candidates as an intrinsic PUF for IoT device security are those using static random-access memory (SRAM) [17], dynamic random-access memory (DRAM) [18, 20, 21], and flash memory [19, 22, 23].

Although SRAM- and DRAM-based PUFs are viable security components, they have several drawbacks. The SRAM unit is usually used as a memory element for the processor, and it is always powered. Therefore, the SRAM PUF response must be extracted during a very early boot stage before the SRAM is used by the processor; this means that SRAM PUFs cannot be used during run-time. DRAM PUFs based on the decay process are known to take more than a few minutes to extract meaningful PUF responses [20], which limits their use in practical security applications.

Another source of memory-based hardware security primitives is commercial off-the-shelf (COTS) flash memory chips. Flash memories are widely used as nonvolatile storage, where NOR flash can be used as code storage for any read-only applications and NAND flash can be used as media storage for a large amount of data. There has been an increasing interest in the generation of flash-based PUFs using cell-to-cell manufacturing variation. Sakib and others proposed a program disturb-based flash PUF, with a PUF response determined as a bit flip pattern due to program stress operation on the erased state page [19]. However, this technology suffers from aging with increasing number of PUF trials. Furthermore, Jia and others presented methods to extract a 128-bit flash PUF response based on partial erasing and partial programming, where the number of partial erasing and programming operations required to flip the selected cells is recorded as the raw output numbers, which are post-processed to obtain a reliable 128-bit key [22]. However, there are no unpredictability results for the generated 128-bit key, which may limit their widespread use in security applications. Kim and others developed a flash PUF by properly modifying the reading voltage [23], which overcomes the aging problem [19]. However, some circuitry (hardware) modifications are required to switch the reading voltage of the flash memory for both normal operation and PUF operation, which makes it difficult to apply it to COTS devices.

In this study, we also considered a COTS flash memory chip as a source of hardware security primitive, where the extraction is based on variability of tunneling electrons in the floating gate of the flash cell. We first define a general hardware security source model and then propose a new flash PUF extraction method from the COTS flash memory chip without hardware modification. Unlike existing flash PUFs, we extract the response in a one-time program/erase (P/E) cycle, leading to robustness over aging. To reproduce PUF responses under various operating conditions, we propose an adaptive method to enhance reliability. The resulting flash-based PUF shows good uniqueness, steadiness, uniformity, and randomness (unpredictability). The main contribution of this study is to provide an efficient and reliable flash PUF that can be practically used in security applications, such as high-entropy cryptographic key generation for IoT devices.

The remainder of this paper is organized as follows. Section 2 defines a general hardware security source mode with a hardware security primitive. Section 3 introduces uncertainty of flash memory resulting from residual electrons in the floating gate of a flash cell and then proposes a flash PUF extraction method that is robust to various operating conditions, aging, and random errors. The

corresponding experimental results and discussion are presented in Section 4. Section 5 demonstrates an application for high-entropy cryptographic key generation based on the proposed flash PUF. Section 6 finally concludes the paper.

2 | HARDWARE SECURITY SOURCE MODEL

We first define a hardware security source model to describe a hardware-based random bit generator for use in security applications, such as high-entropy cryptographic key generation. Figure 1 illustrates the source model including two main parts. We note that this model is similar to an entropy source model defined by NIST [24], in that it must generate a sequence of random bits, but the *output* must be reproduced for the same input (*challenge*).

In Figure 1A, the hardware security primitive corresponds to various PUF sources, such as RO PUF, Arbiter PUF, and Memory PUFs, as previously described. The post-processing component is a function to improve the quality of primitive *raw response* [25]. Debiasing is a post-processing algorithm used to eliminate bias in *raw response*. It is well known that the biased PUF response is problematic when used with an error correction method. Due to helper data entropy loss [11], it is common that remaining entropy in the corrected PUF response cannot be assumed. To avoid this case, post-processing can be applied to the *raw response*, and this optionally post-processed *raw response* is finally called the *response* of the hardware security primitive in our model.

In Figure 1B, errors are corrected so that the same *response* is always generated for the same *challenge*, and an entropy can be amplified by a conditioning function (such as a Hash function) [11, 12]. If there is no error in the *response* or the entropy itself is large enough, these components do not need to be used. Then, the final high-entropy *output* of this model can be used for security applications such as device-specific cryptographic key

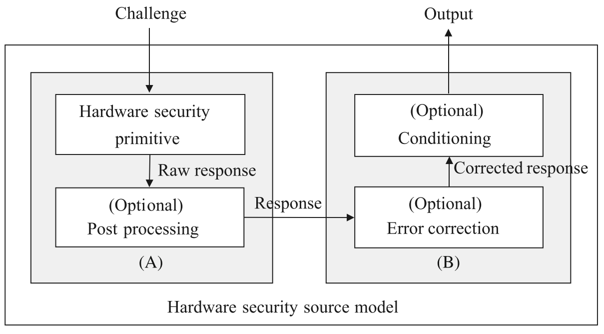


FIGURE 1 Schematic of the hardware security source model

generation. All optional blocks in Figure 1 inevitably incur entropy loss. First, there is bit loss in the debiasing algorithm depending on the number of "1" in the *raw response*. Second, the helper data stored for error correction leak information about the *response*, and its entropy loss depends on the error-correcting codes used. Finally, the error-corrected *response* can be compressed to amplify its entropy (e.g., via a Hash function) such that the *output* has high-entropy or full-entropy per bit. Therefore, considering all of these factors, it is necessary to increase the number of extracted *raw response* bits from the hardware security primitive to derive the final *output* with the desired entropy.

In Sections 3 and 4, we focus on the first part shown in Figure 1A, where the flash memory is used as our hardware security primitive and the performance (e.g., uniqueness, steadiness, uniformity, and randomness) is assessed on the *response* bit stream. Section 5 describes the calculation of the number of *raw response* bits according to the required *output* bits and the level of entropy for key generation of IoT devices, which considers the second part with error correction and entropy amplification.

3 | PROPOSED FLASH-BASED SECURITY PRIMITIVE

This section describes the primary operation of flash memory to clarify the source of the uncertainty, to demonstrate that flash memory is a good source of a hardware security primitive. Then, we propose a flash PUF extraction algorithm, especially focusing on robustness over operation conditions such as environmental variations, device-dependent timing variations, and aging.

3.1 | Flash memory operation and uncertainty

A flash memory cell is a floating gate MOS transistor, that is, a transistor with a gate completely surrounded by dielectrics, the floating gate (FG), and electrically governed by a capacitively coupled control gate [26]. Being electrically isolated, the FG stores electrons for the cell device. The quality of the dielectrics guarantees the non-volatility, whereas its thickness allows the possibility to program or erase the cell via electrical pulses. Usually, the gate dielectric, that is, the one between the transistor channel and the FG, is called a "tunnel oxide" because Fowler–Nordheim (FN) electron tunneling occurs through it [26].

Figure 2 shows the current-voltage plane of the erased state cell and programmed state cell, where both states give the same transconductance curve, but it is shifted by a magnitude that is proportional to the stored electron charge Q in the FG. When reading the current at a fixed gate bias V_{read}, a very high current ($I_d > I_{d,ref}$) means the erased state of logic "1" and zero current means the programmed state of logic "0" [26].

The erasing process is performed by the FN electron tunneling mechanism, which is a quantum-mechanical tunnel induced by an electric field. Applying a strong electric field across a thin oxide for a sufficient period of time (e.g., typically several hundreds of milliseconds), it is possible to force electron tunneling through it without destroying its dielectric properties. However, stopping the erasing process within the required erasing time prevents electron tunneling. The red lines in Figure 2 represent shifted transconductance curves due to the erasing process being stopped after different time periods, where the magnitude of the shift is proportional to the residual electron charge in the FG. Reading the cell with V_{read} thus gives a cell-dependent fixed logic value of "0" or "1." When the erasing process is stopped and its current-voltage plane is positioned at the boundary that determines "0" and "1," the cell is read randomly without yielding a fixed logic value. This causes errors in the PUF response, which are discussed later in this section as they introduce reliability issues.

On the other hand, each flash cell may store a different number of charges in FG as a result of variations during the manufacturing process, which results in a different threshold voltage V_{th} per cell. The distribution of V_{th} is wide, so cells with low (or high) V_{th} show faster (or slower) erase times [26]. Given enough time for the electrons stored in the FG to tunnel through the tunnel oxide, all dedicated cells in a flash memory device will eventually output the erased logic state of 1. However, if the erasing process is intentionally stopped at a specific elapsed time, a unique bit flip ("0" → "1") occurs for each cell location. Exactly which bits are erased (changed from "0" to "1") earlier than the others depends on the quantity of tunneling electrons present when the erasing process is stopped, which results in uncertainties in the flash memory performance.

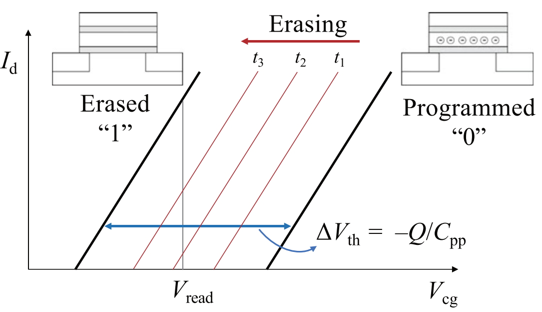


FIGURE 2 Erasing and read operation of a flash cell

3.2 | Basic concept of proposed flash PUF extraction

Based on the cell-to-cell variability of tunneling electrons in FG during the erasing process, we first explain a basic concept to extract a flash PUF response. Figure 3 shows the erasing process over an elapsed time considering an 8-bit flash memory with an initial state of all "0" as an example. When a memory is read at the elapsed time t_1, there is no cell that has been bit-flipped from "0" to "1," and the first bit flip occurs at t_2. Over time, the number of bit-flipped cells increases, and eventually all 8-bit cells are erased to the final state of "1."

As described earlier, the bit-flipped cells at a specific elapsed time is unique for each device, and for each cell in the same device, which enables the construction of a flash PUF based on this randomness. Consequently, we can extract a flash PUF response to set a value obtained by reading a dedicated memory at a specific elapsed time. As an example in Figure 3, the readout value at t_3, that is, "01001000," is used as the *raw response* of the flash PUF. This raw response will be ideally reproduced when the same elapsed time t_3 is applied to the flash PUF.

3.3 | Reliability considerations

In practice, each time a PUF instance is queried with a challenge, it will return a slightly different response due to errors. To use a PUF in security applications, the number of erroneous bits in the *response* should be minimized so that the error correction overheads decrease

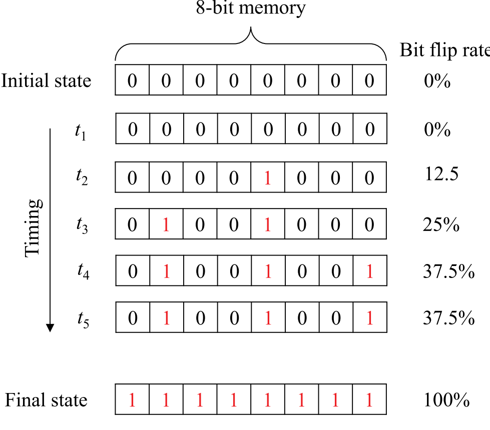


FIGURE 3 Basic concept of the flash PUF extraction

significantly. Here, we consider several factors that cause errors, toward the aim of establishing a reliable flash PUF that will react similarly under the expected operating conditions.

One obvious factor contributing to errors is environmental variations, especially the dependence of the MOSFET V_{th} on the operating temperature T. The variations in temperature lead to variations in the number of tunneling electrons in FG, which influences V_{th}. In this case, the current-voltage plane will be shifted by variations in T. Other common variations in environmental conditions, such as voltage and pressure, are not important in this work because COTS devices are not expected to be influenced by them under realistic conditions.

Next, the COTS device-dependent timing variation is also an important factor, mainly arising from variations in transferring commands to the flash memory chip in the asynchronous interface, which is controlled by a microcontroller. To extract our response from the flash PUF, the erasing process must be stopped after a specific elapsed time, but unpredictable timing variations occur at the interface between the MCU and flash memory chip, leading to errors.

Moreover, aging is another influential operating condition that can drastically influence flash memory. Flash chips are usually specified for 10^5 P/E cycles and cycling causes a fairly uniform degradation of the cell performance, mainly due to tunnel oxide degradation. This effect leads to an increased erase time, which in turn reduces the reliability of flash PUF.

Apart from the aforementioned three factors, variations in V_{th} always result in the corresponding random errors in the proposed flash PUF extraction. When the erasing process is stopped, the V_{th} is not shifted sufficiently from the programmed state to the erased state, and some cells will be an uncertain state. In these cases, the cells can be read as either logic state "0" or "1." This is a random behavior that leads to an error in a PUF response. Note that these errors can be used as a source of random numbers or nonce. However, in this work, we focus on the static response behavior for PUF applications, where it is desirable to eliminate random-number-like errors. The easy method for achieving this is to perform multiple readouts and filter the errors by a majority rule. For example, if logic state "0" appears more than 3 times out of 5 readouts, the resulting logic state is considered to be "0," and vice versa.

Figure 4 shows the number of bit flips (i.e., erased bits) in a page (2048 bits) as a function of the erase elapsed time. Even if the elapsed time is fixed during the erasing process, some cells experience bit flips whereas others do not. For example, the number of bit flips at 34 ms of erase time varies from 685 bits to 823 bits at room temperature $T = 25\,^\circ\text{C}$, which corresponds to a 7% variation occurs. This is mainly due to environmental variations, as previously explained. Figure 4 also shows that variations in bit flipping increase as the elapsed time increases, which leads to a corresponding increase in the bit error rate. Here, the appropriate erase time for each page of a flash chip can be set to ensure small variations in the number of bit flips. Figure 4 also shows the number of bit flips at different temperatures. The graph is shifted to left at a high temperature of $T = 70\,^\circ\text{C}$, whereas the graph is shifted to right at low temperature of $T = -20\,^\circ\text{C}$.

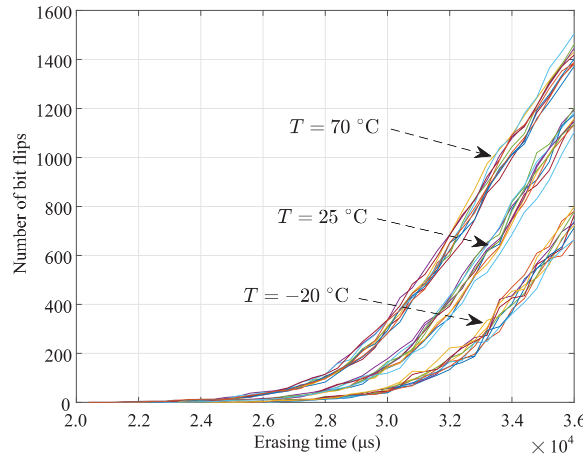


FIGURE 4 Number of bit flips during erasing process for different temperatures

---

Algorithm 1 Extraction algorithm for the flash PUF *raw response*

Require:
  Specific sector to erase: S_i
  Number of target pages belonging to S_i: P
  Delay time to suspend: T_s
  Target threshold of bit flips: n_{th}
Ensure: Number of bit flips among P pages, n_{bf}, needs to reach n_{th}.
  Program the target P pages in S_i to all 0s;
  Set the initial $n_{bf}(j)$ *as* 0 *at* $j = 0$;
  Erase S_i;
  while $n_{bf}(j) \geq n_{th}$ **do**
    $j = j + 1$;
    Delay T_s *and SUSPEND*;
    Read P pages m times and apply majority rule;
    Record n bits *in rawResponse*;
    Count bit flips in rawResponse and update $n_{bf}(j)$;
    RESUME;
  end while
  Record rawResponse and j;

3.4 | Reliable flash PUF extraction algorithm

Considering the reliability factors discussed above, we propose an adaptive time to stop the erasing process to satisfy the target number of bit flips in a specified memory region. By increasing the small amount of elapsed time during the erasing process, we can read the memory at each elapsed time and check whether the number of bit flips satisfies the target value. In other words, the challenge for the flash PUF now becomes the target number of bit flips in a specified memory region, rather than a fixed elapsed time, and its raw response will be the readout value when the erasing process is stopped at the time that satisfies the target number of bit flips.

The proposed flash PUF implementation requires system-privileged flash operations (such as Reset, Erase, Program, and Suspend/Resume), which are expected to be available in most commercial flash memory chips. Relying on these operations, we finally propose the flash-based PUF extraction mechanism in Algorithm 1. Initially, the specific sector S_i and pages P belonging to the selected sector S_i are selected to extract *n raw response* bits. The P are programmed to the logic state "0." In this case, the readouts of P are all zeros and thus the number of bit flips in the initial state is $n_{\text{bf}} = 0$, where the bit flip means the transition from "0" to "1" and is calculated by counting the number of "1" in n bits. Then, an "Erase" operation is started in S_i and an iterative "Suspend"-"Resume" operation is performed at a time interval T_s. Once erasing is stopped or suspended at the jth time interval, n cells (P pages) m times are read and the majority rule is applied to these multiple readouts to filter out erroneous bits. Then, the filtered readout values of n cells are recorded in *rawResponse*. The number of bit flips among n bits in *rawResponse* is counted to determine $n_{\text{bf}}(j)$. If $n_{\text{bf}}(j)$ is larger than the predefined target number of bit flips n_{th}, then its recorded *rawResponse* becomes the Flash PUF *raw response* and the extraction process stops after the "Resume" operation. Otherwise, the "Resume" and "Suspend" operations are repeated with T_s until the stop condition, $n_{\text{bf}}(j) \geq n_{\text{th}}$, is satisfied.

Furthermore, the target number of bit flips n_{th} in the extraction algorithm must be carefully selected. There is a tradeoff between the error rate occurring in the *raw response* and the efficiency of post-processing to remove the bias in the *raw response*. If a target is selected where excessive bit flipping occurs, the number of bit flips can greatly vary when performing the extraction of the *raw response*, which leads to errors. Therefore, it is desirable to determine n_{th} as a function of the erasing time to avoid an abrupt increase in bit flips. On the other hand, if n_{th} is selected where bit flips rarely occur, the *raw response* becomes too sparse (i.e., almost all "0"), so that the number of bits in the debiased *response* will be small.

We analyzed the time required to extract PUF response bits using Algorithm 1. Denoting T_p as the page program time and T_r as the page read time, the time overhead is calculated as

$$T = P \cdot T_p + N \cdot (T_s + T_{\text{sus}} + m \cdot P \cdot T_r), \qquad (1)$$

where N is determined as the j that satisfies $n_{\text{bf}}(j) \geq n$th and T_{sus} is a suspend latency where the following reading operation is possible after the Suspend step. Note that from (1), for each iteration, a delay time T_s, suspend delay T_{sus}, and m times P page read time ($m \cdot P \cdot T_r$) are required. The main advantage of the proposed flash PUF is that it can be easily applied to most IoT sensor devices equipped with flash memory without any hardware modifications. In addition, it requires only one P/E cycle to extract a PUF response, which minimizes the effect of chip aging. Moreover, the proposed extraction algorithm is an adaptive method that can cope with various operating conditions to provide a reliable PUF response. The capability of extracting a PUF response from even a single page of flash memory is sufficient to enable a large number of challenge response pairs, which in turn makes it attractive for various security applications.

3.5 | Post-processing (debiasing)

As mentioned in Section 2, a simple post-processing operation can be used to increase the quality of the hardware security primitive *raw response* [25]. Because bias inevitably occurs due to the characteristics of the proposed flash PUF, we suggest using von Neumann's method as a post-processing function to reduce the statistical bias of the *raw response* to give an unbiased *response*. Von Neumann's method produces independent unbiased random bits from biased bits. If the two consecutive input bits are "00" or "11," then they are discarded. If the input bits are "01" or "10," then the first input bit is taken [27, 28]. Debiasing typically produces side information called *debiasing alignment dataD* that contains information where bits are maintained or discarded. This D needs to be stored publicly and used during reproduction to avoid errors due to misalignment in the reproduced response data. Note that it was previously demonstrated that no entropy leakage due to D occurs when applying von Neumann's method [28].

It is known that for a source that produces biased random bits $(s_1, s_2, \ldots)$, with $\Pr(s_i = 0) = p$ and $p \neq 1/2$, von Neumann's method extracts approximately $n \cdot p \cdot (1-p)$

unbiased bits from n biased bits. Although this debiasing method is a very simple post-processing function, the number of output bits is proportionally reduced by $p \cdot (1-p)$. Considering the proposed flash PUF extraction algorithm, p becomes $1-n_{\text{th}}/n$ because n_{th} is the number of "1" among the n *raw response* bits, so the dropping rate d mentioned in Section 2 will be

$$d = \left(1 - \frac{n_{\text{th}}}{n}\right) \cdot \frac{n_{\text{th}}}{n}. \quad (2)$$

Then, the number of *response* bits after debiasing would be approximately $n \cdot d$. Some efficient post-processing methods are also being developed [25, 29, 30], but these are beyond the scope of this work.

4 | EXPERIMENTAL RESULTS AND DISCUSSION

We implemented and tested the proposed Flash PUF. We first introduce four commonly used performance metrics for PUF responses (uniqueness, steadiness, uniformity, and randomness), where each of these metrics quantifies an essential quality factor of a PUF [14]. Then, we describe our test environments and performance results. Experimental results on the temperature dependency and comparisons are also provided in this section.

4.1 | Performance metrics

Uniqueness (Extra-Chip Variation: EC) is the variability across multiple devices and was assessed by evaluating the inter-Hamming distance of the responses among the PUF devices. Note that the optimal value for uniqueness is 50%. Deviations from this optimal value demonstrate a correlation between PUF instances.

Steadiness (Intra-Chip Variation: IC) is the repeatability over multiple measurements in a particular PUF device and this is assessed by evaluating the intra-Hamming distance of the responses in a PUF device. This metric is used to determine the required error correction method and related parameters. An ideal PUF always returns the same response for a given challenge, so the value for steadiness is 0%.

Uniformity (U) is also an important metric to describe the distribution of 0s and 1s in a PUF response, where the ideal uniformity is achieved when "0" and "1" occur with equal probability. The proposed Flash PUF is processed by von Neumann's debiasing method, so the uniformity is always near the optimal value of 50%.

Randomness (R) represents the unpredictability. In contrast with three first metrics, there is no specific formula to evaluate the randomness of PUFs. We suggest using the test provided by the NIST randomness test suite (NIST SP 800-90B) [24]. This test suite provides min-entropy, which helps establish the minimum response length required to meet a device's security requirements. For example, if a PUF generates 256 bit responses, but those responses contain only 0.25 min-entropy per bit, the maximum security level that can be achieved from security algorithms or protocols using these responses is 64 bits of security. The optimal min-entropy per bit is 1.

4.2 | Test environments and results

We designed our own test platform to implement the proposed flash PUF. The platform is equipped with a widely used microcontroller unit (MCU) at speeds up to 216 MHz, which is the STM32F779 with Arm Cortex-M7. We used the MT25QL512 NOR-Type 64 MB flash memory from Micron Technology. The flash chip has 1024 sectors of size 64 KB and each sector includes 256 pages with a size of 256 bytes. The MCU communicates with the flash memory using a quad serial peripheral interface (QSPI). To realize Algorithm 1 of the proposed flash PUF, we used commands that the MT25QL512 flash device already supports: PAGE PROGRAM, SECTOR ERASE, ERASE SUSPEND/RESUME, and READ [31]. If the flash device that provides these commands is connected to the MCU through an SPI (or QSPI), anyone can implement Algorithm 1. Moreover, many commercial flash devices, especially NOR-Type SPI flash chips, support those commands [31, 32], so the proposed flash-based PUF can be applied in many cases.

Figure 5 shows our test platform and test environment, where the evaluation is performed over 8 boards.

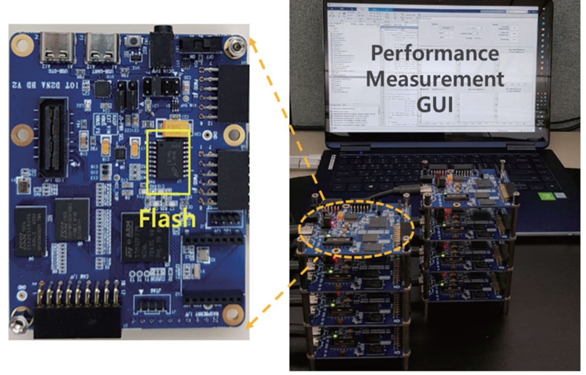


FIGURE 5 Test platform and test environment

Our tests involved two steps. First, we calculated the uniqueness, steadiness, and uniformity for a 128-bit PUF response. Second, we perform a randomness test by NIST SP 800-90B that requires at least a 1-Mbit PUF response. For both tests, the von Neumann debiasing method in Section 3.5 was used and implemented with the on-board MCU.*Test 1*: For uniqueness, steadiness, and uniformity, we considered a given one page (256 bytes) at a specific sector for each platform to extract 2048 bits of *raw response*. The parameters for Algorithm 1 were number of raw response bits $n = 2048$; number of target pages $P = 1$, delay time to suspend $T_s = 200$ μs. The target threshold of bit flips is $n_{th} = 220$, which is the value to avoid an abrupt increase in bit flips according to the erasing time, as shown in Figure 4. The 2048 bits were repeatedly extracted $m = 5$ times, filtered using the majority rule, and then processed by von Neumann's debiasing method. Because $n = 2048$ and $n_{th} = 220$, p is approximately 0.9 and d was calculated as 0.09 as described in Section 3.5. Therefore, we obtained approximately 184 unbiased bits from 2048 bits, that is, $n \cdot d = 2048 \times 0.09 = 184$. Among the unbiased bits, we took the first 128 bits for the metric calculations over 8 different boards. Using (1), we calculated the time overhead to extract 2048 raw response bits in Test 1. For the NOR flash chip used in the experiments, a typical page program time (256 bytes) was $T_p = 120$ μs and a typical suspend latency is $T_{sus} = 15$ μs [31]. To reduce the time overhead in the experiment, the iteration in Algorithm 1 was started from 28 ms, because meaningful bit flips only occurred after this time, and the number of bit flips reached to the target n_{th} around 31 ms at room temperature (Figure 4). Noting that $T_s = 200$ μs, $P = 1$, and $m = 5$ in test 1, it takes (120 μs + 28ms) for a 1 page program and erase time before the first suspend, and (200 μs + 15μs + $5 \cdot T_r$) for each iteration. The number of iterations N was different for each of the 8 different boards, and it was found experimentally within 15 iterations in most cases. Although the exact T_r is not provided by [31], the overall timing overhead is approximately 32 ms to extract 2048 raw response bits, assuming a T_r of several microseconds. Table 1 summarizes the performance of the proposed flash PUF at room temperature. The IC and EC calculations were performed by recording the PUF responses with a laptop 20 times. Table 1 shows that EC for the 8 boards is 49.1%, which is almost optimal. The IC is 2%–5% for each board, which is close to the optimal value of 0%. The final uniformity value was approximately 50%.

Test 2: To evaluate the randomness, we considered 32 sectors to extract sufficient *raw response* bits because the NIST test suite require at least 1 Mbit [24]. All parameters were the same as for *Test 1*, but the collected *raw response* bits was $n = 2^{24}$. Then, the unbiased *response* bits that were obtain from the von Neumann method were approximately 2 Mbit. The NIST test suite provided 10 entropy estimates and min-entropy was determined as

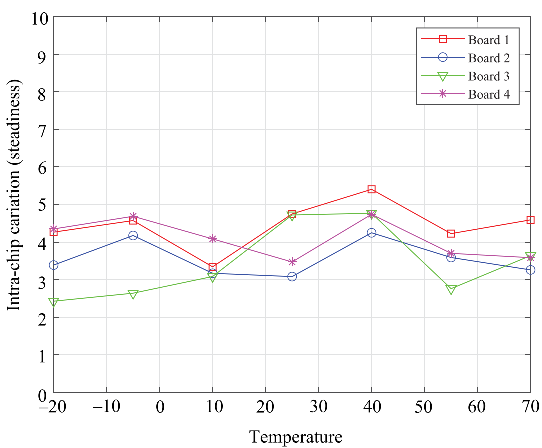


FIGURE 6 Steadiness as a function of temperature

TABLE 1 Performance evaluation of the flash PUF

Board	Uniqueness (*EC*)	Steadiness (*IC*)	Uniformity (*U*)	Min. Entropy h_{min}
Board 1	49.10%	4.75%	53.13%	0.8751
Board 2	49.10%	3.08%	50.31%	0.8592
Board 3	49.10%	4.72%	52.81%	0.8731
Board 4	49.10%	3.47%	46.06%	0.8767
Board 5	49.10%	3.59%	47.19%	0.8876
Board 6	49.10%	4.19%	50.16%	0.8801
Board 7	49.10%	2.39%	49.69%	0.8882
Board 8	49.10%	4.53%	52.03%	0.8919

the minimum entropy value among these estimates [24]. Table 1 shows that the min-entropy for each device was approximately 0.85–0.89 per bit.

4.3 | Temperature dependence

To evaluate the steadiness of the proposed flash PUF under different temperatures, 4 boards of the test platform in Figure 5 were placed in a temperature chamber. The temperature was varied from −20 °C to 70 °C with 15 °C increments, including room temperature 25 °C as the reference PUF response. Figure 6 shows the steadiness values for each of the 4 boards at the various test temperatures, where 20 measurements were taken at each temperature. The steadiness only fluctuated slightly over the entire temperature, remaining within 6%.

4.4 | Comparisons and discussion

Table 2 compares the performance of the proposed PUF and existing PUFs in terms of the four performance metrics (i.e., *EC*, *IC*, *U*, and *R*), and run-time extractability. The optimal values for *EC*, *IC*, *U*, and *R* are 50%, 0%, 50%, and 1, respectively. Although a direct comparison is not straightforward because of the different platforms used in each study, this overview indicates that the proposed PUF scheme is appropriate for resource-constrained IoT devices.

The conventional SRAM PUF shows excellent performance close to the optimal values, as shown in Table 2. SRAM PUF can be modified for use during run-time by allocating a dedicated SRAM section for PUF use and implementing SRAM to be powered on and off. However, such modifications are not easy to find in general IoT devices, so access to SRAM PUF during normal operation is limited. In contrast, DRAM PUF can be accessed at any time, but typically takes a few minutes to generate a response. In addition, DRAM PUFs show relatively low uniqueness performance (*EC*).

We also compared the existing flash PUFs presented in previous studies [19, 22] that do not require any hardware modifications. These PUFs have good performance in terms of uniqueness, steadiness, and uniformity. However, the program disturb-based flash PUF of [19] requires many program cycles (e.g., around 10,000). [22] mainly proposed post-processing algorithms (different from the debiasing method) to select the most reliable bits among raw response bits, leading to high reliability. Moreover, there were no randomness (unpredictability) results presented in these studies. Although existing flash PUFs can be good candidates for specific security applications, we believe that our high-performance flash PUF with only one P/E cycle could be a practical security technique to extract the cryptographic key for IoT devices.

5 | APPLICATION FOR CRYPTOGRAPHIC KEY GENERATION

We now introduce an application for cryptographic key generation based on the proposed flash PUF, where a high-entropy key can be generated with the second part of hardware security source model described in Section 2. In general, cryptographic keys are generated from a true random number generator, stored in nonvolatile memory, and loaded when necessary. Such key management is not desirable in an IoT environment where the number of devices is large. It is expected that more than 82 billion IoT devices will be connected to the Internet by 2025, and at least 256-bit keys are required for this cases. In addition, because the generated 256-bit key must be unpredictable for the attacker, it is desirable to have high-entropy or full-entropy conditions. In this section, we show how an effective 256-bit key can be generated using our flash-based hardware security primitive.

Figure 1B consists of a reproduction block that can correct errors and a Hash function that can amplify entropy with compression [11, 12]. Keeping in mind that the PUF based cryptographic key must be reproducible

TABLE 2 Comparison of the proposed PUF and existing PUFs

PUFs	Uniqueness (*EC*)	Steadiness (*IC*) (Nominal)	Uniformity (*U*)	Randomness (*R*) (min-entropy)	Run-time extraction
Proposed flash PUF	49.1%	2%–5%	46%–53%	0.85–0.89	Possible
SRAM PUF [17]	49.7%	2%–5%	Not reported	0.57–0.7	Partially possible
DRAM PUF [18]	35.09%	2.2%	Not reported	Not reported	Difficult
Sakib's flash PUF [19]	49%–51%	3%–8%	42%–51%	Not reported	Possible
Jia's flash PUF [22]	46.8%–49.9%	$<10^{-6}$ (post-processed)	46.9%–53.6%	Not reported	possible

on demand, an error correction code (ECC) can be used to reproduce the initial PUF response with helper data [33, 34]. However, there is a drawback in storing helper data, which can introduce a security problem [11]. Entropy is reduced by the code rate r of the ECC due to information leakage of the helper data. In contrast, when using a Hash function, which is one of the vetted conditioning functions defined by NIST SP 800-90B, the entropy for a k-bit Hash function output (h_{out}) can be estimated [24]. Furthermore, full-entropy output bits, that is, $h_{\text{out}} = k$, can be produced when the input entropy h_{in} is at least $c \cdot k$ with a compression rate of the Hash function $c \geq 2$. Otherwise, the output entropy is lower.

Specifically, let us reconsider the hardware security source model from an entropy perspective, as shown in Figure 7. Assuming that k-bit *output* is generated with full-entropy, we consider the debiasing method with dropping rate d, the ECC with code rate r, and the Hash function with compression rate c. If the min-entropy per bit at the *response* bit stream is $h_{\min}$, the output entropy at the ECC or the input entropy at the Hash function would be $h_{\text{in}} = r \cdot n \cdot d \cdot h_{\min}$. For full-entropy requirements, h_{in} should be at least $c \cdot k$, so the minimum number of bits for the extracted *raw response*, denoted as n, is given by

$$n = \frac{c \cdot k}{r \cdot d \cdot h_{\min}}, \tag{3}$$

where the minimum possible c is 2.

Then, we applied (3) to the proposed flash PUF to evaluate how many *raw response* bits are required for the full-entropy 256 output bits. Based on the experiments described in Section 4, $h_{\min} = 0.87$, as in Table 1, and $d = 0.09$. The used ECC is the convolutional code with $r = 1/3$ and memory length 6, which corrects errors up to $IC = 15\%$ [34]. The required number of *raw response* bits is then calculated as $n = 19$ Kbits. Therefore, if 19 Kbits *raw response* are extracted from the flash PUF, the final 256-bit *output* of the Hash function (e.g., SHA-256) is guaranteed to have full-entropy, which can be used as a successful cryptographic key for IoT devices.

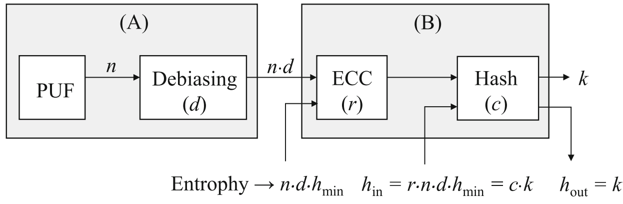


FIGURE 7 Hardware security source model considering entropy

5.1 | Security analysis

Facilitated by the proposed flash PUF and its corresponding ECC and Hash function, a full-entropy 256-key is obtained at the *output* in the hardware security source model. Therefore, an attacker can decode the encrypted data only with a brute force attack. As the resources required for brute force attacks grow exponentially with increasing key size (i.e., 2^k), the 256-bit key generated by the proposed PUF is considered computationally secure against brute force attacks.

6 | CONCLUSION

In this paper, we presented a flash-based hardware security primitive for cryptographic key generation for IoT devices. Relying on variability of tunneling electrons in the FG during the erasing process, we proposed a flash PUF construction using a one-time P/E cycle. Moreover, the adaptive PUF extraction algorithm was shown to be robust for various operating conditions, such as temperature and timing variations. The easy multiple readout capability provided by our flash PUF results in enhanced reliability over a variable random behavior. Experimental results using COTS memory chips confirmed the excellent performance of the PUF in terms of uniqueness, steadiness, uniformity, and randomness. Because the proposed flash PUF enables the generation of credentials or keys for resource-constrained IoT devices at any time without hardware modification, security threats to IoT environments can be reduced by an easily adopted security framework. Future works will include performance evaluation of the proposed flash PUF with various flash devices.

CONFLICTS OF INTEREST

The authors declare that there are no conflicts of interest.

ORCID

Mi-Kyung Oh https://orcid.org/0000-0002-5712-8625
Dooho Choi https://orcid.org/0000-0001-5625-4067

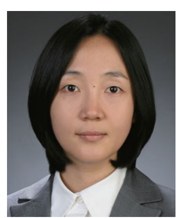
REFERENCES

1. S. Li, T. Zhang, B. Yu, and K. He, *A provably secure and practicle PUF-based end-to-end mutual authentication and key exchange protocol for IoT*, IEEE Sensors J. **21** (2021), no. 4, 5487–5501. https://doi.org/10.1109/JSEN.2020.3028872
2. V. P. Yanambaka, S. P. Mohanty, E. Kougianos, and D. Puthal, *PMsec: Physical unclonable function-based robust and lightweight authentication in the Internet of Medical Things*, IEEE Trans. Consum. Electron. **65** (2019), no. 3, 388–397. https://doi.org/10.1109/TCE.2019.2926192

3. A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari, *Enabling drones in the internet of things with decentralized blockchain-based security*, IEEE Int. Things J. **8** (2021), no. 8, 6406–6415. https://doi.org/10.1109/JIOT.2020.3015382
4. C. Huth, D. Becker, J. G. Merchan, P. Duplys, and T. Güneysu, *Securing systems with indispensable entropy: LWE-based lossless computational fuzzy extractor for the Internet of Things*, IEEE Access **5** (2017), no. 2, 11909–11926. https://doi.org/10.1109/ACCESS.2017.2713835
5. Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, *A survey on security and privacy issues in Internet of Things*, IEEE Int. Things J. **4** (2017), no. 5, 1250–1258. https://doi.org/10.1109/JIOT.2017.2694844
6. M. N. Aman, K. C. Chua, and B. Sikdar, *Physical unclonable functions for IoT security*, (Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, Xi'an, China), 2016, pp. 10–13. https://doi.org/10.1145/2899007.2899013
7. J. R. Wallrabenstein, *Practical and secure IoT device authentication using physical unclonable functions*, (IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria), 2016, pp. 99–106. https://doi.org/10.1109/FiCloud.2016.22
8. R. S. Pappu, *Physical one-way functions*, Ph.D. dissertation, MIT, Cambridge, MA, USA, (2001).
9. S. Lim, B. Song, and S. Jung, *Highly independent MTJ-based PUF system using diode-connected transistor and two-step post-processing for improved response stability*, IEEE Trans. Inf. Forensics Security **15** (2020), 2798–2807. https://doi.org/10.1109/TIFS.2020.2976623
10. U. Rührmair, S. Devadas, and F. Koushanfar, *Security based on physical unclonability and disorder*, *Introduction to Hardware Security and Trust*, Springer, New York, NY, USA, 2012, pp. 65–102. https://doi.org/10.1007/978-1-4419-8080-9_4
11. C. Herder, M. Yu, F. Koushanfar, and S. Devadas, *Physical unclonable functions and applications: A tutorial*, Proc. IEEE **102** (2014), no. 8, 1126–1141. https://doi.org/10.1109/JPROC.2014.2320516
12. M. D. Yu and S. Devadas, *Secure and robust error correction for physical unclonable functions*, IEEE Design Test Comput. **27** (2010), no. 1, 48–65. https://doi.org/10.1109/MDT.2010.25
13. A. Maiti, J. Casarona, L. McHale, and P. Schaumont, *A large scale characterization of RO-PUF*, (IEEE International Symposium on Hardware-Oriented Security and Trust, Anaheim, CA, USA), 2010, pp. 94–99. https://doi.org/10.1109/HST.2010.5513108
14. A. Cherkaoui, L. Bossuet, and C. Marchand, *Design, evaluation, and optimization of physical unclonable functions based on transient effect ring oscillators*, IEEE Trans. Inf. Forensics Security **11** (2016), no. 6, 2191–1305. https://doi.org/10.1109/TIFS.2016.2524666
15. Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, *Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGA*, (International conference on reconfigurable computing and FPGAs, Quintana Roo, Mexico), 2010, pp. 298–303. https://doi.org/10.1109/ReConFigure2010.24
16. B. Li, S. Chen, and F. Dan, *Design and implementation of an improved MA-APUF with higer uniqueness and security*, ETRI J. **42** (2019), no. 2, 205–216. https://doi.org/10.4218/etrij.2019-0081
17. G.-J. Schrijen and V. Leest, *Comparative analysis of SRAM memories used as PUF primitives*, (Design, Automation & Test in Europe Conference & Exhibition, Dresden, Germany), 2012, pp. 1–14. https://doi.org/10.1109/DATE.2012.6176696
18. Q. Tang, C. Zhou, W. Choi, G. Kang, J. Park, K. K. Parhi, and C. H. Kim, *A DRAM based physical unclonable function capable of generating >10^{32} challenge response pairs per 1Kbit array for secure chip authentication*, (IEEE Custom Integrated Circuits Conference, Austin, TX, USA), 2017, pp. 1–4. https://doi.org/10.1109/CICC.2017.7993610
19. S. Sakib, M. T. Rahman, A. Milenković, and B. Ray, *Flash memory based physical unclonable function*, (Proc. SoutheastCon, Huntsville, AL, USA), 2019, pp. 1–6. https://doi.org/10.1109/SoutheastCon42311.2019.9020567
20. A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, *Decay-based DRAM PUFs in commodity devices*, IEEE Trans. Dependable and Secure Compt. **16** (2019), no. 3, 462–475. https://doi.org/10.1109/TDSC.2018.2822298
21. I. Kumari, M. Oh, Y. Kang, and D. Choi, *Rapid run-time DRAM PUF based on bit-flip position for secure IoT devices*, (Proc. SENSORS, New Delhi, India), 2018. https://doi.org/10.1109/ICSENS.2018.8589608
22. S. Jia, L. Xia, Z. Wang, J. Lin, G. Zhang, and Y. Ji, *Extracting robust keys from NAND flash physical unclonable functions*, *Information Security, ISC 2015*, J. Lopez and C. Mitchell, (eds.), Lecture Notes in Computer Science, Vol. **9290**, Springer, Cham, 2015. https://doi.org/10.1007/978-3-319-23318-5_24
23. M. Kim, D. Moon, S. Yoo, S. Lee, and Y. Choi, *Investigation of physically unclonable functions using flash memory for integrated circuit authentication*, IEEE Trans. Nanotechnology **14** (2015), no. 2, 384–389. https://doi.org/10.1109/TNANO.2015.2397956
24. NIST Special publication (SP) 800-90B, *Recommandation for the entropy sources used for random bit generation*, 2018. https://doi.org/10.6028/NIST.SP.800-90B
25. B. Sunar, W. J. Martin, and D. R. Stinson, *A provably secure true random number generator with built-in tolerance to active attacks*, IEEE Trans. Computers **56** (2007), no. 1, 109–119. https://doi.org/10.1109/TC.2007.250627
26. R. Bez, E. Camerlenthgi, A. Modelli, and A. Visconti, *Introduction to Flash memory*, Proc. IEEE **91** (2003), no. 4, 489–502. https://doi.org/10.1109/JPROC.2003.811702
27. B. Skoric, *A trivial debiasing scheme for helper data systems*, 2016. Cryptology ePrint ARchive, Report 2016/241.
28. R. Maes, V. Leest, E. Sluis, and F. Willems, *Secure key generation from biased PUFs*, (International Workshop on Cryptographic Hardware and Embedded Systems, Saint-Malo, France), 2015, pp. 517–534.
29. M. Suzuki, R. Ueno, N. Homma, and T. Aoki, *Quaternary debiasing for physical unclonable functions*, (IEEE 48th International Symposium on Multiple-Valued Logic, Linz, Austria), 2018, pp. 7–12. https://doi.org/10.1109/ISMVL.2018.00010
30. S. H. Kwok, Y. L. Ee, G. Chew, K. Zheng, K. Khoo, and C. H. Tan, *A comparison of post-processing techniques for biased*

random number generators, (International Workshop on Information Security Theory and Practices), 2011, pp. 175–190.
31. Micron Serial NOR Flash Memory (MT25QL512), 2019. Available: https://www.micron.com/products/nor-flash/serial-nor-flash/part-catalog/mt25ql512abb1ew9-0sit
32. GigaDevice Flash Memory Device (GD25B256E), 2020. Available: https://www.gigadevice.com/flash-memory/gd25b256e
33. S. Puchinger, S. Müelich, M. Bossert, M. Hiller, and G. Sigl, *On error correction for physical unclonable functions*, (10th International ITG Conference on Systems, Communications and Coding, Hamburg, Germany), 2015, pp. 1–6.
34. S. Müelich and M. Bossert, *Applying convolutional codes to key extraction using ring oscillator PUFs*, (Workshop on Optimal Codes and Related Topics, Sofia, Bulgaria), 2017, pp. 98–103.

AUTHOR BIOGRAPHIES



Mi-Kyung Oh received her B.S. degree from the Department of Electrical Engineering, Chung-ang University, Seoul, Korea in 2000 and M.S. degree and Ph.D in Electrical Engineering from the Korea Advanced Institute of Science and Technology (KAIST), Deajeon, Korea in 2002 and 2006, respectively. From September 2002 to February 2004, she was a visiting researcher in the Department of Electrical and Computer Engineering at the University of Minnesota, USA. She is currently a principal researcher at the Electronics and Telecommunications Research Institute (ETRI) in Korea. Her research interests are wireless communication systems, SoC design, and IoT security technologies.

Sangjae Lee received his B.S. and M.S. degrees in Electrical Engineering from Chonbuk National University, and a Ph.D in Information and Communication Engineering from Chungbuk National University, Korea in 1999, 2001, and 2013, respectively. He has been a Principal Researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, since 2000. He has been developing technologies for home gateway, home server, IEEE1394, VoIP, network traffic controller, and wireless PAN MAC, and UWB SoC. His current research interests include wireless MAC and SoC design for wireless PAN, and IoT security technologies.

Yousung Kang received B.Eng. and M.Eng. degrees in electronics engineering from Chonnam National University, Gwangju, South Korea, in 1997 and 1999, respectively, and a Ph.D. degree in electrical and electronic engineering from KAIST, Daejeon, South Korea, in 2015. He has been a Principal Researcher with the Electronics and Telecommunications Research Institute, Daejeon, since 1999. Since 2004, he has been with the IT International Standard Expert of Telecommunications Technology Association, Seoul, South Korea. He was a Visiting Researcher with Queens University Belfast, Belfast, U.K., from 2011 to 2012. His current research interests include cryptographic protocol, side channel analysis, and key-hiding technology.

Dooho Choi received a B.S. degree in mathematics from Sungkyunkwan University, Seoul, South Korea, in 1994, and the M.S. and Ph.D. degrees in mathematics from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1996 and 2002, respectively. He is currently an Associate Professor at Korea University Sejong in Korea from 2021. He was a professor at the University of Science and Technology from 2015 to 2020 and a visiting research fellow at Queens University Belfast from 2016 to 2017. He worked for the Electronics and Telecommunications Research Institute (ETRI) as a principal researcher from 2002 to 2020. His main research interests include side channel analysis and its countermeasure design, quantum crypto analysis, and security technologies of IoT.

How to cite this article: M.-K. Oh, S. Lee, Y. Kang, and D. Choi, *Implementation and characterization of flash-based hardware security primitives for cryptographic key generation*, ETRI Journal **45** (2023), 346–357. https://doi.org/10.4218/etrij.2021-0455