IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 18, 2019, accepted June 1, 2019, date of publication June 14, 2019, date of current version July 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923174

PreLatPUF: Exploiting DRAM Latency Variations for Generating Robust Device Signatures

B. M. S. BAHAR TALUKDER[1], (Student Member, IEEE), BISWAJIT RAY[1], (Member, IEEE), DOMENIC FORTE[2], (Senior Member, IEEE), AND MD TAUHIDUR RAHMAN[1], (Member, IEEE)
[1]ECE Department, The University of Alabama in Huntsville, Huntsville, AL 35899, USA
[2]ECE Department, University of Florida, Gainesville, FL 35899, USA

Corresponding author: B. M. S. Bahar Talukder (bms.btalukder@uah.edu)

This work was supported in part by the National Science Foundation under Grant CNS-1850241 and in part by the University of Alabama in Huntsville.

ABSTRACT Physically unclonable functions (PUFs) are potential security blocks to generate unique and more secure keys in low-cost cryptographic applications. Dynamic random-access memory (DRAM) has been proposed as one of the promising candidates for generating robust keys. Unfortunately, the existing techniques of generating device signatures from DRAM is very slow, destructive (destroy the current data), and disruptive to system operation. In this paper, we propose *precharge* latency-based PUF (PreLatPUF) that exploits DRAM *precharge* latency variations to generate signatures. The proposed PreLatPUF is fast, robust, least disruptive, and non-destructive. The silicon results from commercially available *DDR*3 chips from different manufacturers show that the proposed key generation technique is at least $\sim 1,192X$ faster than the existing approaches, while reliably reproducing the key in extreme operating conditions.

INDEX TERMS DRAM-PUF, DRAM latency-based PUF, robust key generation.

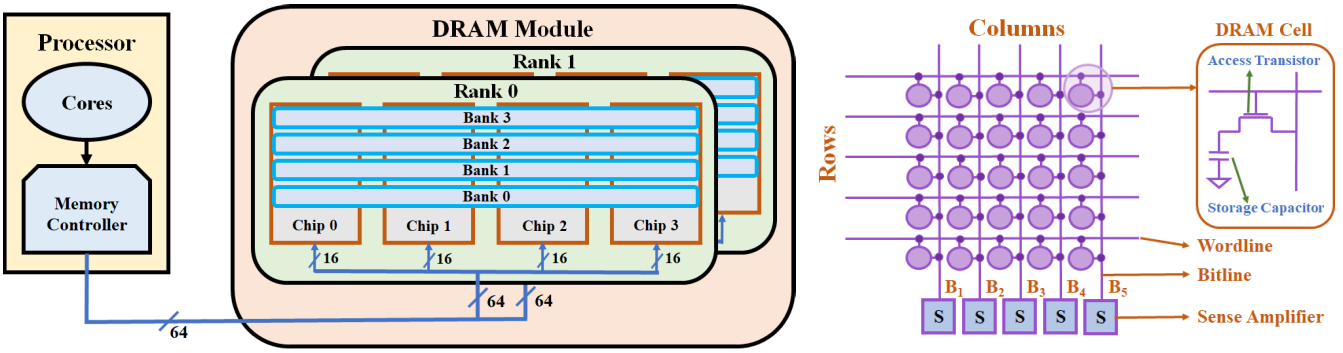
I. INTRODUCTION

Physical unclonable functions (PUFs) play important roles in security by offering a high level of protection in cryptographic applications with the capability of strong volatile key or unique ID generation. A PUF is a circuit that generates unique fingerprints by exploiting the inherent and unavoidable manufacturing process variations during fabrication [1], [2]. Identification, authentication, secure communication, IC obfuscation to prevent IC piracy in semiconductor supply chain, detection of counterfeit ICs, etc. are a few common applications of PUFs because of their unique and unpredictable characteristics [1], [3]–[9]. In recent years, PUFs have also been used in IoT applications because they enable low-cost solutions with a high level of security [10]–[12].

In addition to low-cost, the memory-based PUF provides an opportunity to implement PUF-based schemes to the existing system [2], [5], [8], [9]. The start-up behavior of the memory chips, disturbance characteristics, the random decay properties, etc. are the most common techniques to generate responses from memory chips [1]. Previous works on DRAM PUFs (DPUFs) have focused on: (i) retention-based: writing all cells to '1' and disabling the refresh then waiting for half the cells to discharge and reading cell values [2], [2], [13], [14], (ii) start-up based: using the start-up values of the cells to generate the secret key as in [15], [16], and (iii) disturbance-based: disturbance caused by rowhammer [17], [18]. The variations in *activation* latency time have also been used to generate device signatures [19]. In this method, the signature is obtained from the errors generated at the reduced *activation* time during read operation [19].

In PUF-based applications, the responses (i.e., the PUF outputs) have to be robust, fast, random, and unique [5], [20]–[23]. Like other silicon PUFs, the DRAM-based PUF responses are also impacted by external influences such as operating and environmental variations, aging, etc. [24]–[31]. In addition, the existing signature generation schemes from DRAM do not offer impressive throughput; retention-based DPUF requires an order of minutes, and start-up based DPUF needs a power cycle. The destructiveness of the memory contents, disruption of the system, etc. are few other major limitations of existing DRAM-based PUFs (discussed in Section II-E).

The associate editor coordinating the review of this manuscript and approving it for publication was Leandros Maglaras.



(i) DRAM system.

(ii) Close view on a Memory Bank.

FIGURE 1. Organization of a modern memory subsystem [35], [36].

While some applications can tolerate a certain amount of errors, others, such as the generation of cryptographic keys, cannot. To make the PUF output more stable (i.e., to obtain the same response for the applied challenge to a PUF), error correcting code (ECC) and different enrollment schemes are often used but at the expense of additional cost [32]–[34].

In this paper, we propose PreLatPUF that exploits the *precharge* timing latency variations in DRAM to generate device signatures. The main contributions of this paper (i.e., to generate robust device signatures from DRAM) are summarized below.

- We propose *precharge* latency based DRAM PUF (PreLatPUF) that generates device signatures at a much faster rate. We experimentally demonstrate that the faulty read operation at the reduced *precharge* latency can be used to generate unique and random device signatures.
- We characterize the errors at the reduced *precharge* latency to discover cells that are most suitable for robust and reliable PUFs.
- We propose a cell selection algorithm and a registration technique to ensure that the signatures generated at the reduced *precharge* latency are robust, unique, and random.
- We present a quantitative and qualitative comparison between PreLatPUF and some of the previously proposed DRAM-based PUFs. The results show that the proposed PreLatPUF outperforms existing DPUFs in several aspects.
- We evaluate the proposed PreLatPUF using commercially available DDR3 DRAM modules.

The rest of the paper is organized as follows. In Section II, we present the background of DRAM architecture, read/write operation, existing DRAM-based PUFs and major challenges. We propose the latency-based DRAM PUF in Section III. The experimental results and discussions are presented in Section IV. We conclude the paper in Section V.

II. BACKGROUND AND MOTIVATION

In this section, we provide a brief background of the modern memory subsystem and its operation. We also present existing DRAM-based PUFs and their limitations.

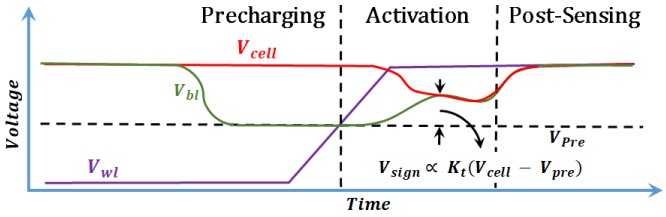
A. DRAM ORGANIZATION

Fig. 1 illustrates the organization of a modern DRAM system, which maintains a hierarchy of channel, rank, bank, DRAM chips, DRAM cells, and memory controller. Depending on the system requirement, different electronic systems can have DRAM modules of different sizes. A DRAM module is divided into one or multiple ranks. The rank is accessed in each reading/writing attempt. Rank, again, consists of several DRAM chips and provides a wide databus together. The same databus is shared among the ranks. A chip select pin is used to choose a particular rank. The width of the databus is usually 64 bits and distributed equally among the chips inside a rank. Each DRAM chip consists of multiple banks to support the parallelism. In a memory bank, the DRAM cells are arranged in a two-dimensional array. The rows and columns of a DRAM are known as *wordline* and *bitline*, respectively. The row of a DRAM is also known as the page. The *bitlines* are connected to the *row-buffer* (a row of *sense-amplifiers*). When a DRAM is read, the *sense-amplifier* senses the stored charge of each memory cell and latches it to a corresponding value (‘1’ or ‘0’). A DRAM cell, the smallest unit, is used to store a single bit (‘1’ or ‘0’). The DRAM cell consists of two components: a capacitor to hold the charge and an access transistor to access the capacitor. The charging state of the capacitor determines the state of the value (‘1’ or ‘0’). A fully charged capacitor is represented by logic ‘1’. On the other hand, logic ‘0’ is the representation of a capacitor with no charge.

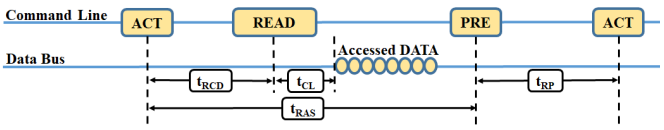
B. DRAM OPERATION

1) READ OPERATION

Fig. 2i presents a simplified DRAM read operation, which consists of several states. In the *precharge* state, the memory controller generates a *precharge* command (*PRE*) to precharge all *bitlines* to $V_{dd}/2$ (green line). This command also deactivates previously activated *wordline*. In the next state (i.e., the *activation* state), the *ACTIVATE* command (*ACT*) from the memory controller activates the target *wordline* by raising the value of *wordline* to V_{dd} (violet line). Once the pass-transistor (connected to the *wordline*) is ON, the charge flows from the capacitor (red line) to the attached



(i) Signal waveform at the reading cycle. [38]



(ii) DRAM timing at the reading cycle. [37].

FIGURE 2. DRAM operation and timing.

bitline if the stored value is '1', and moves from *bitline* to the capacitor if the stored value is '0'. In the final stage, the differential *sense-amplifier* senses the voltage perturbation on the *bitline* and amplifies the *bitline* voltage to a strong logic '1' (or '0'). Then, the *sense-amplifier* latches the logic value from the *bitline*. In the DRAM system, the read operation is destructive; therefore, rewriting after reading is mandatory.

2) WRITE OPERATION

In the *write* operation, initially all *bitlines* are precharged to $V_{dd}/2$ with the *PRE* command. The *ACT* command is applied to write data into a specific *wordline*. The *sense-amplifier* with desired logic value enables the corresponding *bitline* to charge or discharge the connected cell capacitor. After each successful *READ/WRITE* operation, the *bitlines* must be precharged back to $V_{dd}/2$ to access a new set of memory cells from a different *wordline*.

C. DRAM TIMING

Timing is critical for reliable DRAM operation. All major timing parameters of a DRAM module are presented in Fig. 2ii. Initially, all *bitlines* are precharged to $V_{dd}/2$. To access the data from a specific *wordline*, *ACTIVATE* (*ACT*) command is applied to the corresponding *wordline*. Once that is completed, a *READ/WRITE* command is sent from the memory controller to sense the voltage perturbation on *bitlines* or to write a data to the memory cells. The minimum required time interval between *ACT* command and *READ/WRITE* command is defined as the *activation time*, t_{RCD}. The *Column Access Strobe* (*CAS*) latency t_{CL} is the minimum waiting time to get the first data bit on data bus after sending a *READ* command. After a successful *READ/WRITE* operation, *precharge* command (*PRE*) is applied to deactivate the previously activated *wordline* (if any) and precharge the *bitlines* to its initial precharge state (i.e., to $V_{dd}/2$). If the *WRITE* command is applied, the *PRE* command should be further delayed by t_{WR} period (*write recovery* time) at the end the write data burst. The *PRE* command is applied for at least t_{RP} (*precharge* time) duration before sending the next *ACT* command. The duration between the *activation* state to the beginning of the *precharge* state is called *row active* time or *restoration* latency (t_{RAS}). The t_{RAS} + t_{RP} is the total time required to access a single row of a bank and is known as *row cycle* time (t_{RC}). Usually, the t_{RC} is in the order of 50ns for most modern DDR3 DRAMs.

D. EXISTING DRAM-BASED PUFS

1) RETENTION-BASED DRAM PUFs (DPUFs)

Signatures are generated by disabling the refresh interval for a certain and sufficient amount of time [13], [19]. The DRAM cells are leaky, and therefore, the DRAM contents need to be refreshed periodically, usually 64ms or 32ms according to the JEDEC specification [38], to ensure the data integrity [13]. Failing to refresh periodically within this time interval introduces errors due to the leaky property of DRAM cells. The error pattern generated from the retention failure is unique from chip to chip and is used to generate device signatures [13], [19].

The retention-based device signature is promising but suffers from several drawbacks that hinder its use in real applications. First, the periodic refresh operation in most DRAM modules is handled internally by a memory controller. There is no efficient way to control this refresh time for an arbitrarily small region of DRAM module since the granularity for such refresh operation is predefined by the vendors. Some common control signals control memory cells under the same granularity, and therefore changing a timing parameter on one cell affects all other cells as well. On the other hand, two rows from two different granular regions can be accessed independently (but not simultaneously as they may share the same channel). For a retention-based DRAM PUF, an authentication key of sufficient length can be generated by retention failure from a small portion of a DRAM module, but the whole operation may cause unwanted data corruption of other memory cells under the same granularity [38]. Second, a key of sufficient length requires an adequate number of errors; therefore, it might need a long waiting time (order of minutes) to generate a key with desired length and quality [19]. Third, the retention time is heavily temperature dependent, which makes the key sensitive to temperature variations [2], [13], [14], [39]–[41]. Previous studies show that the *bit error rate (BER)* increases exponentially with the temperature; the key generation scheme requires a longer time interval between two refresh operations at a lower temperature [42]. The required time to generate the key is also a function of the size of the memory segment. A smaller segment requires longer evaluation time than a larger one [19].

Therefore, the designer must decide on area vs. time overhead. Several techniques can be used to address the above challenges but with a limited gain [13], [24], [42]–[47].

2) LATENCY-BASED DPUFs

The reduction in t_{RCD} introduces erroneous read/write operation (see Section II-C), which can be used to generate device signatures [19]. This latency-based PUF generates signature at a much faster rate [19]. The reported result shows that the mean evaluation time is ∼88.2ms (outperforms all previously proposed retention-based DPUFs [2], [13], [14]). However, it still requires multiple row cycles to evaluate the PUF response. This latency-based DPUF also needs a filtering mechanism in each access that adds both hardware and computational overheads.

3) START-UP BASED DPUFs

In start-up based DPUF [16], the device signature is generated from the start-up states of DRAM cells. Initially, the bitlines are charged to $V_{dd}/2$. But the process variations on the storage capacitor slightly deviate the *bitline* voltage to $V_{dd}/2+\delta$ or $V_{dd}/2-\delta$, where δ represents a small voltage. The *sense-amplifier* senses the voltage difference to '1' or '0', accordingly. Upon power-up, the DRAM cells generate '1's and '0's randomly. The significant challenges of a start-up based DPUF are: (i) requirement of a power-cycle and (ii) a time gap between the turn-OFF and turn-ON is required to avoid a strong correlation between the data before turn-OFF and the signature.

4) ROWHAMMER DPUFs

The errors caused by the rowhammer disturbance are used to generate device signatures [17], [18]. This technique does not require any additional power cycle. However, the average evaluation time of a rowhammer PUF is in order of minutes and therefore might not be suitable in many applications. Besides, all DRAMs are not vulnerable to rowhammer [17].

E. MOTIVATIONS

Below, we summarize the major motivations of our proposed work.

- **Waste of DRAM Power Cycle:** Start-up based key generation requires a DRAM power cycle to obtain device signatures [16]. Hence, the whole system needs a power cycle (i.e., a turn-off and a turn-on) to obtain the PUF response. Therefore, this type of PUF cannot be evaluated while the system is in operation.
- **Large Evaluation Time:** Rowhammer-based and retention-based key generation techniques require an order of minutes to generate enough bit failures and therefore not suitable for many applications [2], [13], [14], [17], [18], [48]. On the other hand, the existing latency-based DPUF still needs multiple row cycles (reading one data burst at each cycle) to evaluate the PUF key [19] since the reduction in activation time only affects the first few bits in the cache line (see Section II-C).
- **Destructive:** Retention-based key generation is destructive. The DRAM granularity causes random failed bit throughout the smallest granular region (usually a rank). Note that the DRAM refresh can be disabled only at the granularity of channels [38]. A dedicated memory might need to be used to overcome this problem but at the expense of additional hardware. The start-up based and rowhammer-based DPUFs are also destructive.
- **Disruptive:** DRAM granularity keeps the entire DRAM rank busy during each access. Hence, such kind of PUF evaluation blocks the access on the target DRAM region by other applications for a long time. Though the existing latency-based DRAM PUF [19] solves the problem of long evaluation time and unwanted data failure (due to the granularity), it still needs a filtering mechanism to evaluate PUF in each access, which introduces additional computational and timing overheads.

III. PRELATPUF: PRECHAREG LATENCY-BASED PUF

In this section, we present the proposed PreLatPUF, cell characterization, and cell selection algorithm.

A. PRECHARGE LATENCY AND SOURCE OF VARIATIONS

The latency is defined as the time required to move charge during read/write operation. In modern DRAM architecture, multiple DRAM cells are connected to the same *bitline* through access transistors. The DRAM vendor provides the minimum required timing latency to perform a reliable read/write operation. Erroneous read/write operation is observed if the minimum timing latency is not maintained [36]. In our experimental results, the following observations have been discovered that are also consistent with [36] and [49].

- **Observation 1**: A reduced t_{RCD} only affects the first accessed column/cache line.
- **Observation 2**: A reduced t_{RP} might affect almost all cells of a row.
- **Observation 3**: Almost no bit error is introduced at the reduced t_{RAS}.

From the above observations, we can conclude that the reduction in t_{RCD} or t_{RP} can be used to generate device signatures from a DRAM. The t_{RCD}-based PUF has been proposed in [19] that needs an additional filtering mechanism and several row cycles (discussed in Section II-E). In this article, we use the t_{RP} variations to generate device signatures.

The DRAM cell characteristics at the reduced t_{RP} mostly rely on the internal structure of a DRAM module, process variations, layout variations, data dependency, etc. [19], [26], [36], [45], [48], [50]–[52]. Fig. 3 presents a simplified structure of the DRAM precharge circuit [53]. In a DRAM module, each DRAM cell is connected to a *bitline* through an access transistor and each *bitline* has a corresponding $\overline{bitline}$ that provides the complementary data (see Fig. 3). Each *bitline* and $\overline{bitline}$ pair contain a

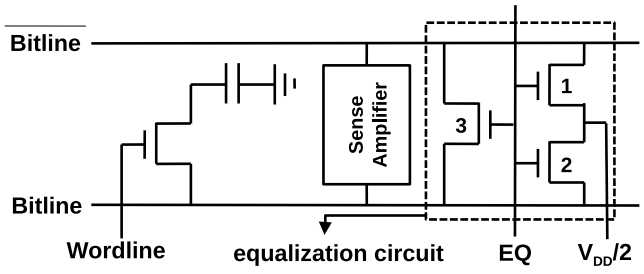


FIGURE 3. Simplified structure of precharge circuit.

sense-amplifier and an equalization circuit. At the precharge state, the transistor 1 and 2 of the equalization circuit create a conducting path with a voltage source $V_{DD}/2$. On the other hand, the transistor 3 of the equalization circuit creates a conducting path between *bitilne* and $\overline{bitline}$. With the proper precharge time, the transistor 1 and 2 get enough time to precharge the *bitline* pair to $V_{DD}/2$, and the transistor 3 further ensures the equalization of voltage on the *bitline* pair. After turning ON the access transistor, the *bitline* voltage is perturbed by the stored charge in the capacitor. Then the perturbed voltage is sensed and amplified with the *sense-amplifier*. However, at the reduced precharge time, the transistor 1 and transistor 2 might not get enough time to precharge the *bitline* pair equally to $V_{DD}/2$. Therefore, the *bitline* and the $\overline{bitline}$ might deviate from VDD/2. The variations on *RC* path delay and the capacitance of the *bitline* follow the Gaussian distribution [54]–[56], and two different DRAM cells of same physical length may have different t_{RP}s.

In addition to this, the process variation also might introduce slight variation on the charge storage capacity of the DRAM cells. Hence, during the READ operation, the intensity of the voltage perturbation on a *bitline* might vary from one memory cell to another memory cell [57]. As a result, these DRAM cells may behave differently at the reduced *precharge* time [37]. In addition, different vendors may follow different kind of configurations (e.g., *open bitline* array structure, *folded bitline* array structure etc. [53]), which may lead to different faulty outputs at the reduced t_{RP}.

Note that the minimum value of t_{RP} is required to deactivate the previously activated row to avoid correlation among the outputs and the contents of the previously activated row. The minimum value of t_{RP} is determined empirically, and it may vary from module to module (discussed in Section IV-B).

B. CHARACTERIZATION

We characterize the DRAM cells to understand the data dependency, spatial correlation, etc. in order to obtain robust PUF signatures. The characterization phase is conducted by observing the outputs with different types of input patterns (e.g., all 1's, all 0's or checkerboard pattern). The term 'input value' or 'input pattern' is used for the pattern that is written in the DRAM memory module with standard timing parameters. On the other hand, the 'output pattern' refers to the output that is read back at the reduced t_{RP}. A particular input pattern is applied several times (more on Section IV) to study the temporal variation (i.e., measurement variation). Based on the data correctness (or incorrect/faulty behavior), we divide the DRAM cells into two major categories:

- **Non-faulty Cells:** These memory cells do not show any errors at the reduced t_{RP} and retain correct data regardless of the input data pattern.
- **Incorrect/Corrupted/Faulty Cells:** These memory cells fail to output the original data (i.e., the input pattern and output pattern are different). The errors might be independent or dependent on the input data.

Based on the temporal variations, again, we categorize the incorrect/faulty cells into the following types:

- **Noisy Cells:** Error pattern varies from measurement to measurement because of internal/external noise for these types of cells. Some of these cells can be useful to generate random number [52]. Some of these cells can be used to create PUF but might require a large ECC [58].
- **Robust/Measurement-invariant Cells:** These cells do not show any temporal variation, i.e., cell outputs are independent of measurements. These cells are tolerant to internal and external noise and ideal for PUF.

In addition, the outputs at the reduced t_{RP} might depend on the memory cell contents (i.e., written patterns) due to the coupling effect of neighborhood cells [51], [59]. Based on the data dependency, we categorize the DRAM cells into following types:

- **Pattern Independent Cells:** These types of cells exhibit the same output (at the reduced t_{RP}) regardless the input patterns. The experimental results show that (details in Section IV), most of the DRAM cells from the major vendors are *pattern independent*. In this paper, we have only focused on the *'pattern independent'* cells for PUF implementation.
- **Pattern Dependent Cells:** The output patterns for these cells are different for different input patterns. Therefore, these cells can be the ideal candidates for the PUF that possesses enhanced challenge-response pair [60]–[62].

C. CELL SELECTION ALGORITHM

In this paper, we only focus on the *pattern independent* cells. The experimental results show that some of the *pattern independent* cells are *strong '1'* and some of them are *strong '0'*. Besides the reproducibility, it is important that the generated key is random and unique as well. Entropy is used to measure the randomness (i.e., the unpredictability) of a bitstream [58], [63]. A binary string of randomly distributed 0's and 1's with equal probability possess high entropy [58], [63], [64]. Not all cells can be used to generate PUF because some DRAM cells create deterministic outputs. We scan each row to find the most suitable cells for generating robust and random keys. We observe that the generated outputs using all *pattern independent* bits of every word (a word is 64 bits wide) suffer from poor entropy. As a part of the

TABLE 1. Selecting appropriate cells with cell selection algorithm.

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}	HW
V_1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	15
V_2	1	0	1	1	0	1	0	0	0	1	0	0	1	0	1	0	7
V_3	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	2
V_4	1	0	0	0	0	1	1	0	1	1	0	1	1	1	0	1	9

entropy test, we count the ratio between the occurrence of 1's and the occurrence of 0's. Our objective is to generate a key that has an equal number of 1's and 0's. The raw outputs show that there is a considerable imbalance between the number of 0's and 1's if we count each failed bit from all words. Therefore, all bits of every word are not suitable for key generation. It is observed that some specific bits of every word of a row produce a predictable outcome. For example, for a particular memory bank, the first bit of every word of a specific row is always read as '0' at a reduced t_{RP}. The binary string (V_1) formed with the first bits of the words cannot be used to generate keys since the *Hamming weight*[1] of the V_1 is 0%. The explanation of this phenomenon as follows: a 64-bit DRAM module is analogous to a combination of 64 2-D memory arrays (distributed into multiple DRAM chips), and each memory array contributes to every word by providing one bit. For example, the 5^{th} memory array is responsible for the 5^{th} bit of the word. The impact of reduced t_{RP} may vary among memory arrays. In our proposed bit selection algorithm, we use an important metric: *Hamming weight*. A 50% of *Hamming weight*, which is ideal for a key, means that the binary string has an equal number of 1's and 0's. Similar to V_1, we create a binary string V_2 with the second bit of each word in a row. Similarly, the binary string generated from the i^{th} bit of each word is V_i. The i^{th} bit of the word is considered as the *eligible bit* if it produces a random binary string V_i with a $\sim$ 50% Hamming weight.

To get the most suitable cells for robust PUF, we propose an algorithm (Algorithm 1) for selecting the qualified memory cells and their locations. In practice, not all binary strings in $\mathcal{V} = \{V_1, V_2, ..., V_{64}\}$ experiences a 50% of *Hamming weight*. Therefore, we choose only those binary strings that fall into a range of allowable Hamming weight (H_{min} to H_{max}). All eligible bits (of words) from a row $\mathcal{R}_x$ can be defined as Eq. (1). Table 1 shows a simplified explanation of selecting eligible bits, where we have presented all memory cells from an imaginary row that has 4-bit (V_1 to V_4) wide 16 words (W_1 to W_{16}). We have produced the first string V_1 by only taking the first bit from each word, V_2 by only taking the second bit from each word and so on. The rightmost column of Table 1 presents the Hamming weight (HW) of each string. For better randomness, the *Hamming weight* of each string should be 50% (8 in this case). However, the silicon results show that it is not always achievable. Therefore, we have to choose a lower limit (H_{min}) and an upper limit (H_{max}) of *Hamming weight*. Let's assume, the chosen values of H_{min} and H_{max} are 5 and 11, respectively. As a result, only cells under the V_2 and V_4 can be used for PUF operation (as Hamming weight of V_2 and V_4 are between 5 and 11, see Table 1). So, according to the Eq. (1), the set of eligible bits is $\beta_{\mathcal{R}_x} = \{2, 4\}$.

If the row $\mathcal{R}_x$ consists of n words, then we can create a binary string from each word by only considering the qualified bits (i.e., the cells that satisfy Eq. (1)). For example, if we consider the i^{th} word W_i from row $\mathcal{R}_x$, then, $W_i^{\beta_{\mathcal{R}_x}}$ is a binary string by taking bits which are the elements of $\beta_{\mathcal{R}_x}$. So, all allowable data bits from the $\mathcal{R}_x$ can be presented as the Eq. (2). Here, $\mathcal{M}_{\mathcal{R}_x}$ is a single dimensional binary string containing all eligible data bits from $\mathcal{R}_x$. According to the Table 1, and Eq. (2), $\mathcal{M}_{\mathcal{R}_x} =$ [11, 00, 10, 10, 00, 11, 01, 00, 01, 11, 00, 01, 11, 01, 10, 01].

$$\beta_{\mathcal{R}_x} = \{b \in \{1, 2, 3, ..., 64\} : H_{min} < Hamming\_weight\_of(V_b) < H_{max}\} \quad (1)$$

$$\mathcal{M}_{\mathcal{R}_x} = [W_1^{\beta_{\mathcal{R}_x}}, W_2^{\beta_{\mathcal{R}_x}}, W_3^{\beta_{\mathcal{R}_x}}, ..., W_n^{\beta_{\mathcal{R}_x}}] \quad (2)$$

However, the length of the key can be larger than the number of qualified memory cells in a binary string $\mathcal{M}_{\mathcal{R}_x}$. In this case, we will have to use more than one binary string from the multiple rows. Algorithm 1 is designed to select the *qualified bits* (i.e., the cells that satisfy Eq. (1)) from each row. From now on to the rest of our discussion, the b^{th} bit of the 64-bit data word, accessed from the location *(r,c)*, will be noted as *(r,c,b)* where, r is the row number (or page number), and c is the column number (c^{th} word of the row r). In Algorithm 1, R_n, C_n, and B_n are the total number of rows, total number of columns, and the word width respectively (constant for a specific memory module). In our experiment, we have used 1GB memory modules, where, $R_n = 16384$, $C_n = 1024$, and $B_n = 64$).

In the proposed Algorithm 1, a one-dimensional array $\mathcal{R}$ and a two-dimensional array β together hold the memory locations of the qualified DRAM cells. The $\mathcal{R}$ holds all eligible row (or page) addresses and β holds corresponding *qualified bit* number of the row. For example, $\mathcal{R} = 1, 3, 4, 7$ represents that 1^{st}, 3^{rd}, 4^{th}, and 7^{th} rows (or pages) are marked as the qualified rows (see Fig. 4). β (on right side) of the fig. 4 represents corresponding locations of the eligible bits. For example, for $\mathcal{R} = 1$, the $\beta = \{2, 5, 8\}$. i.e. 2^{nd}, 5^{th} and 8^{th} bit of all words from row 1 can be used to generate key.

[1]The *Hamming weight* is defined as the total number 1's (or 0's) in a bitstream.

Algorithm 1 Selecting Qualified Memory Cells

Input:

mem_data: A $R_n \times C_n \times B_n$ matrix, containing pattern independent data. An element of *mem_data* can be empty (if the corresponding memory cell is not pattern independent) or '0' or '1'.

H_{min} & H_{max}: Minimum and maximum allowable Hamming weight as described in sec III-C.

Output:

$\mathcal{R}$: 1D array, contains the list of qualified rows which holds *qualified bits* for PUF generation

β: 2D array, i^{th} row is associated with the i^{th} row of $\mathcal{R}$. Each row of β contains all *qualified bits* from each word of the corresponding row.

```
β = [ ]; // β initialized with empty matrix
R = [ ]; // R initialized with empty matrix
bit_count = 0;
row_count = 0;
row_flag = false;
for r = 1 to R_n do
    for b = 1 to B_n do
        V_b = [ ];
        k = 0;
        for i = 1 to C_n do
            temp = mem_data (r, i, b) ;
            if is_pattern_independent(temp) == true then
                V_b (k) = temp;
                k = ++;
            end if
        end for
        h = Hamming_weight_of (V_b) ;
        if h > H_min && h < H_max then
            row_flag = true;
            β (row_count, bit_count) = b;
            bit_count + +;
        end if
    end for
    if row_flag == true then
        R (row_count) = r;
        row_count + +;
        bit_count = 0;
        row_flag = false;
    end if
end for
```

D. REGISTRATION

In the *registration* phase, we generate a golden data set (i.e. challenge-response data set) using Algorithm 2, which can be used to generate robust signatures. We assume that the golden data set is created and stored in a trusted environment. In the Algorithm 2, we use qualified memory cells that are obtained using Algorithm 1. In Algorithm 2, the *goldenDataLoc* holds the logical locations of eligible memory cells and the *goldenData* saves the outputs that are accessed from the corresponding locations at the reduced t_{RP}. The *goldenDataLoc*, *goldenData*, and the reduced value of t_{RP} will be used as the golden data set for future authentication.

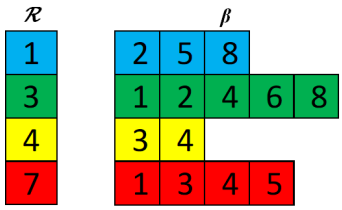


FIGURE 4. Qualified row position and corresponding bit position in words.

Algorithm 2 Generating Golden Data

Input:

mem_data: A $R_n \times C_n \times B_n$ matrix, containing pattern independent data. An element of *mem_data* can be empty (if the corresponding memory cell is not pattern independent) or '0' or '1' .

β & $\mathcal{R}$: generated from algorithm 1.

Output:

goldenDataLoc: A boolean matrix of size $R_n \times C_n \times B_n$. *goldenDataLoc(r,c,b)* is *true* if corresponding memory cell qualified for the PUF application

goldenData: Matrix of size $R_n \times C_n \times B_n$, contains pattern independent output of those memory cells that are marked as *true* in *goldenDataLoc* matrix.

```
goldenDataLoc = boolean_matrix (Rn, Cn, Bn) ;
goldenData = matrix (Rn, Cn, Bn) ;
for i = 1 to length(R) do
    for j = 1 to length(β (i, 1 to end)) do
        for k = 1 to C_n do
            temp = mem_data (R (i) , k, β (i, j)) ;
            if is_pattern_independent(temp) == true then
                goldenDataLoc (R (i) , k, β (i, j)) = true;
                goldenData (R (i) , k, β (i, j)) = temp;
            end if
        end for
    end for
end for
```

IV. RESULT AND ANALYSIS

Our results are based on experiments conducted with six memory banks from two commercial DDR3 memory modules of two major memory vendors[2] (namely *A* and *B*). We used *SoftMC* (Soft Memory Controller [50]) along with the *Xilinx ML605* Evaluation Kit which is embedded with *Virtex-6* FPGA. *SoftMC* uses *Riffa* [65] framework to establish communication between a host PC and the evaluation board through x8 PCIe bus. To check the design reliability

[2]vendor *A*: *Micron*, vendor *B*: *Samsung*

against voltage variation, we used a USB interface adapter evaluation module [66] for controlling the voltage of the memory module very precisely.

The experiment was performed in two steps. First, an 8-bit pattern was written at the regular timing parameter and then read it back at the reduced timing parameter. The reading operation was done in a single row cycle, i.e., we activated one *wordline* at a time and then read all *bitlines* with consecutive burst. Here, each data burst was able to capture the data from successive 8 *bitlines*. This whole process was done at the nominal operating voltage and room temperature (i.e., 25°C and 1.5V for all modules). To obtain and analyze the error pattern, we first checked the *Hamming Distance* between the written pattern (input pattern) and the pattern that was read out (output pattern) with the reduced timing parameter. Then, failed bits were analyzed for additional information (e.g., spatial distribution, pattern dependency, etc.). Four sets of 8-bit input patterns (*0xFF*, *0xAA*, *0x55*, *0x00*) were used to characterize the DRAM cells. For each set of the input pattern, we repeated our experiment five times (hence, produced 20 sets of data) to study the temporal variation. Independent analysis is done by choosing random memory banks (four from vendor *A* and two from *B*; each consists 128MB memory cells).

We conducted our experiment on DRAM memory module by changing the *activation* time (t_{RCD}), *restoration* time (t_{RAS}), and *precharge* time (t_{RP}).

A. REDUCED LATENCY: ACTIVATION TIME VS. PRECHARGE TIME

We read a whole row in a single row-cycle to evaluate the error patterns generated at the reduced t_{RCD}. Two 32-byte (double-data rate) memory chunks were read with each burst (with 8-bit burst length, i.e., eight words can be accessed at a time while each word corresponds to 64-bit data). From now on to rest of our discussion, we will use the notation $t_{A,x}$ to present the reduced timing parameter t_A, where x is the reduced value of the timing parameter in nanosecond. At a reduced activation time (e.g., at $t_{RCD,5.0}$), failed bits were only observed at the first accessed cache line (i.e., in the first 64-byte data) and therefore it needs several read cycles. All memory banks from our selected manufacturers exhibit similar characteristics. Such behavior is observed because the target *wordline* is fully activated before accessing the second content of the cache line (see appendix). Note that [36] and [19] also presented similar observation.

On the other hand, the experimental results show that enough reduction in t_{RP} creates errors uniformly across the whole word. In addition, it requires only a single row-cycle. Fig. 5 shows that the percentage of failed bits in two random banks from two vendors for different input patterns at the reduced t_{RP}. We observed the first error(s) at $t_{RP,7.5}$. We reduced the t_{RP} to $t_{RP,7.5}$, $t_{RP,5.0}$, and $t_{RP,2.5}$ to observe the behavior of failed bits. The results show that the total number of failed bits are $\ll 1\%$ at $t_{RP,5.0}$ for vendor *A*. The rate of the failed bits increases at a much faster rate as we decrease

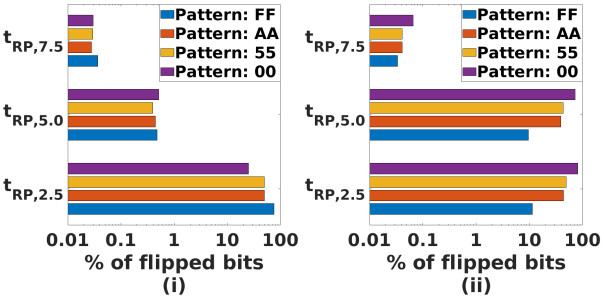


FIGURE 5. t_{RP} vs. % of failed bits- (i) from vendor *A* and (ii) from vendor *B* (the horizontal axis shown in logarithmic scale).

the t_{RP} further. For *vendor B*, the total number of failed bits are $\ll 1\%$ at $t_{RP,7.5}$ but increase significantly at $t_{RP,5.0}$.

We also discover that DRAMs from different manufacturers react differently for a given input pattern. Fig. 5 (left) shows that most of the cells produce faulty outputs with the input pattern that has all 1's, but most of the bits are faultless when the input pattern is all 0's. On the other hand, we observe in Fig. 5 (right) that most of the bits are failed when the input pattern is all 0's but most of the bits are seemed to be correct when the input pattern is all 1's. In the left figure, the number of failed bits for input pattern *0xFF* is higher because the pattern independent '0' (output always '0' regardless of the input pattern) cells are dominant for this module. In the right figure, the number of failed bits for the input pattern *0x00* is higher as the pattern independent '1' (output always '1' regardless of the input pattern) cell is dominant for this module.

We can conclude from the results that (i) reducing *precharge* time is superior to the reducing *activation time* for generating quality signatures in a single row-cycle, and (ii) the erroneous behavior depends on the input pattern, the DRAM architecture, process variations, the amount of reduction in t_RP, etc.

B. CELL CHARACTERIZATION

The silicon results show that a different number of faulty outputs are generated at different reduced t_{RP}s. In addition, we must ensure that the reduced t_{RP} is also capable of closing the previously activated row (as discussed in Sec. III-A). The chosen value of t_{RP} is empirical (denoted as $t_{RP,PUF}$ for clarification), which used to characterized DRAM cells, and to evaluate the PUF responses. Note that the $t_{RP,PUF}$ might vary from module to module. Also, the cell characterization is done at nominal voltage and room temperature (i.e., 25°C and 1.5V for all modules).

1) **Pattern Independent:** Memory cells from this category always flip to a fixed value (either to '0' or to '1') regardless of the input pattern (i.e., originally written value to the DRAM cells). Fig. 6, a 3D histogram plot to describe the spatial locality of the pattern independent cells, shows the spatial locality (along 16384 rows and 1024 columns) of output '0' (left) and '1' (right) across

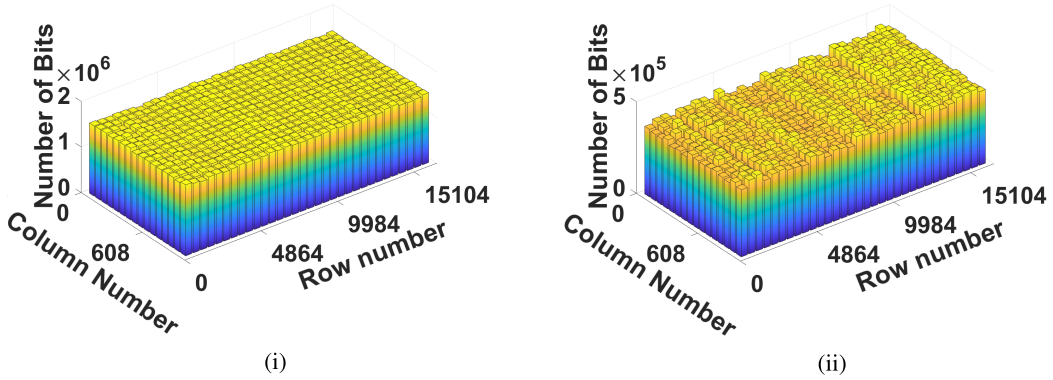


FIGURE 6. Spatial location of pattern independent cells (at $t_{RP,PUF} = 2.5$ns), (i) bit '0', (ii) bit '1'.

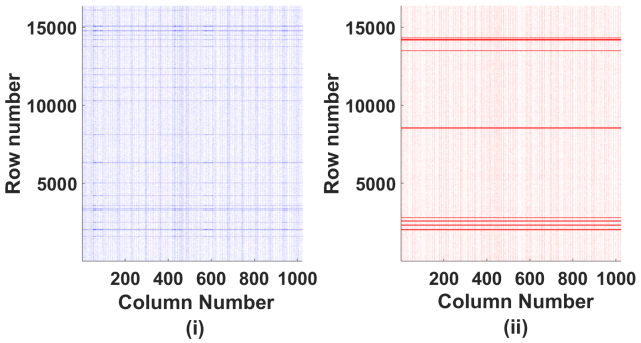


FIGURE 7. Pattern dependent cells (at $t_{RP,PUF} = 2.5$ns), (i) failed to '0', and (ii) failed to '1'.

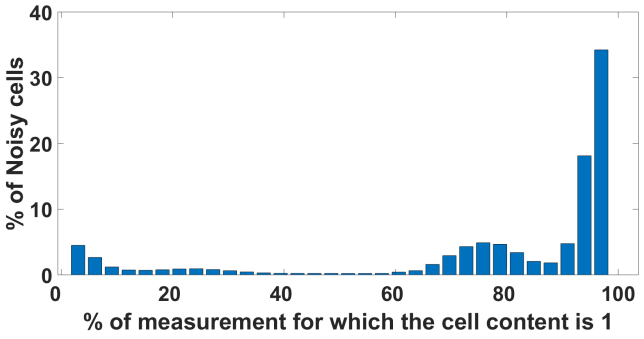


FIGURE 8. Noisy cell characteristics. Most of the cells are biased to '1'.

a random DRAM bank from vendor *A*. The results show that pattern independent 1's and 0's are uniformly distributed. All memory banks from vendor *B* also show similar spatial distribution (not shown in the figure). Therefore, the reduction of t_{RP} is a better candidate to generate device signatures.

2) **Pattern Dependent:** The outputs of these type of cells depend on the input patterns written into the DRAM cells. The outputs are affected by the cumulative voltage of partially precharged *bitline*, stored values, the coupling effect of neighbor cells, etc. We consider a memory cell as a pattern dependent if it provides different outputs for different inputs. These cells are also measurement invariant for at least one input pattern. Fig. 7 shows the DRAM cells that are dependent on input patterns *0xAA*. Pattern dependent cells can be used for PUF with an enhanced challenge-response pair (CRP) space. Besides, spatial locality along both row and column are visible in Fig. 7. The darker line in the Fig. 7 (both horizontal and vertical) represents rows and columns with the pattern dependent cells. A darker line signifies that it has more pattern dependent cells. The spatial locality might reveal the physical to logical address mapping [45]. Fig. 7 is captured from a random bank of vendor *B*, a similar type of spatial locality was found in all memory banks from all vendors. The third column (from right) in Table 2 shows the percentage of pattern dependent cells from each bank.

3) **Noisy Cells:** With partially *precharged* bitlines, outputs of these cells vary from measurement to measurement. Hence, these noisy cells are not suitable to be used as PUF. The second column (from right) in Table 2 represents the percentage of noisy cells from each bank. In Fig. 8, we demonstrate the distribution of noisy memory cells for a random bank from the vendor *B*. The results show that the noisy cells are not entirely random (in this case, most of the cells are biased to '1'). Similar characteristics were found in other memory banks from both vendors (i.e. most of the noisy cells are biased to either '0' or '1'). Large ECC might be required if these cells are used as a PUF [32], [33] because of their poor reproducibility. We also found that the spatial locality of noisy cells from one bank to another is random. Therefore, a proper subset of such cells can also be used to generate a random number [52].

The complete distribution of these three types of DRAM cells along the *bitline* is presented in Fig. 9 for a given bank of *vendor A*. In this figure, we presented only 128 *bitlines* of two consecutive 64-bit words, where each *bitline* consists of 16384 memory cells (since the total number of

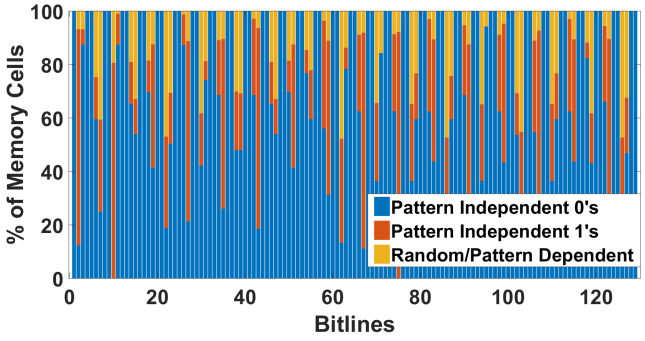


FIGURE 9. Cell distribution among *bitlines*.

rows is 16384). The figure shows that all memory cells from $4n^{th}$ and $(4n+1)^{th}$ (where, $n = 1, 2, 3, ...$) bits of a word generate '0' regardless of the input patterns. One of the possible reasons is that, for these *bitlines*, a large voltage difference with corresponding $\overline{bitlines}$ causes the *sense-amplifier* to deviate towards a specific logic level (either '0' or '1') (see Sec. III-A). Therefore, the generation of key from such memory cells reduces the overall entropy of the key. The proposed Algorithm 1 eliminates such memory cells and improves the entropy.

TABLE 2. Distribution of memory cells at the partial *precharge* state.

Vendor	Memory Bank ID	$t_{RP,PUF}$ (ns)	Pattern Independent		Pattern Dependent (%)	Noisy (%)	Vaid bits (%)
			0 (%)	1 (%)			
A	a	2.5	85.825	12.631	0.006	1.537	0.000
	b	2.5	72.663	18.790	0.135	8.413	0.000
	c	2.5	72.793	17.202	0.133	9.872	0.000
	d	5	7.820	10.560	0.310	81.030	0.290
B	a	2.5	8.226	63.674	0.519	27.580	0.001
	b	2.5	6.339	53.530	0.113	40.017	0.001

Table 2 summarizes the distribution of the cells of two different vendors (vendor *A*, and vendor *B*) at $t_{RP,PUF}$. The results show that more than 90% cells from each bank of vendor *A* (except the bank *d*) are pattern independent while it is $< 75\%$ for the vendor *B*. However, we found an exception for the bank *d* of vendor *A* because the previously activated row fails to close at $t_{RP,2.5}$. To avoid this issue, we characterized memory cells of this bank with $t_{RP,5.0}$. For this particular memory bank, we found that the independent pattern cells are fewer in numbers than the other memory banks. We also found that the number of noisy cells increases by a significant margin than the other memory banks.

C. PRELATPUF EVALUATION

We use *diffuseness, uniqueness, and reliability*, three major PUF performance metrics [22], [23], [67], to quantify and compare (with other DPUFs) the quality of the proposed PreLatPUF. The proposed Algorithm 1, presented in Section III, is used to obtain the logical locations of the qualified memory cells. In this algorithm, we used $H_{min} = 0.25$ and $H_{max} = 0.75$ as the input parameters. Ideally, the Hamming distance should be 0.5. A Hamming distance of 0 represents that the PUF is not unique. We completed the registration (i.e., creating the golden data set) using the proposed Algorithm 2. We generated at least one 1024-bit key from each qualified row (or page). However, it is possible to generate multiple keys from each row since the number of qualified memory cells was more than 1024. To keep it simple, we obtained only one key from each row to test the PUF performance. The key generated from the golden data set is used as the *reference key*. We refer the corresponding address for generating a *reference key* as the *key address*. To evaluate the performance of our proposed PreLatPUF, we created four sets of test data in four different operating conditions (will be discussed in IV-C.3). We measured the output for the four different input patterns (*0xFF*, *0xAA*, *0x55*, and *0x00*) and took the average. The outputs from different operating conditions were compared with the *reference key* to ensure the robustness of our proposed key generation methodology. We present the major performance metrics below.

1) DIFFUSENESS

PUF device should be able to generate distinguishable responses with different challenges. For PreLatPUF, we consider the address as the challenge and corresponding cell content at reduced $t_{RP,PUF}$ as the response. To check the diffuseness, we measured inter Hamming Distance (inter HD) among the reference keys from each bank (i.e., intra-bank but inter-reference key). A 50% of inter HD signifies that a unique key can be generated from each row (i.e., address). The average Hamming weight of 50% also represents that the keys are random. Table 3 shows the average Hamming weight of each key and average Hamming distance among the different keys generated from each bank. Though the average HD and Hamming weight for a few banks deviate from 50%, the silicon results from all rows of each bank show that the keys generated from a distant row of the same memory bank are not repetitive.

TABLE 3. Average Hamming weight and average Hamming distance among the keys generated from each bank.

Vendor	Memory Bank ID	#Qualified row (%)	Average Hamming Distance (%)	Average Hamming weight (%)
A	a	100.00	48.87	54.23
	b	92.31	49.35	53.29
	c	92.30	49.24	49.24
	d	67.82	28.97	53.98
B	a	74.84	42.28	68.19
	b	63.99	38.06	70.31

2) UNIQUENESS

Responses from different devices should be unique. This metric tells that the PUF 1 is different from the PUF 2. To quantify the uniqueness, we measured the inter Hamming

Distance (inter HD) of the key from different memory banks, i.e. the HD between the two keys of two banks generated from each key address. We checked the inter HD for the following combinations by taking account following scenarios:

- A different pair of banks that are from the same module.
- A different pair of banks that are from different modules but from the same vendor.
- A different pair of banks that are from two different vendors.

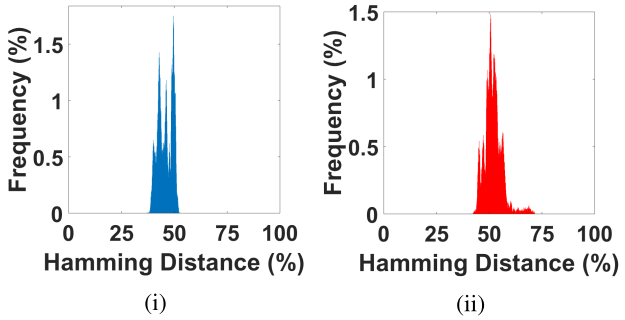


FIGURE 10. Inter Hamming distance (at $t_{RP,PUF} = 2.5$ns) for the worst case from (i) vendor *A* and (ii) vendor *B*.

Fig. 10 shows the inter HD at the worst case (i.e., the largest deviation from 50% inter HD) scenario for both vendors. In the worst scenario, for the vendor *A*, the average, minimum, and maximum inter HD are 45.78%, 37.05%, and 52.5% respectively. For the vendor *B*, the mean, minimum, and maximum inter HD are 51.91%, 40.92%, and 72.23%, respectively. Therefore, we can conclude that the key generated from the proposed PreLatPUF is unique.

3) RELIABILITY

Same response (i.e., PUF output) should be generated to its entire lifetime at any operating condition. The reproducibility at different operating conditions is presented in Fig. 11. This figure presents only the worst results from each vendor (i.e., memory bank with the most significant deviation from 0% intra HD). To examine the robustness of the proposed PreLatPUF at extreme operating conditions, we collected results at four different operating conditions: (i) nominal voltage and room temperature (NVRT), (ii) low-voltage and room temperature (LVRT), (iii) high-voltage and room temperature (HVRT), and (iv) nominal voltage and high temperature (NVHT). The results show that the memory module from vendor *A* is less robust than the vendor *B* at the reduced operating voltage. For vendor *A*, we can only change the operating voltage by $-20mv$ without causing an excessive error on PUF response. On the other hand, the DRAM module from vendor *B* can tolerate $-55mv$ change in operating voltage. Table 4 presents the intra HD at different operating conditions. The column 4 of Table 4 represents change in operating voltage from the nominal (1.5V) and the column 5 represents change in temperature from room temperature (25°C). The results show that all memory banks from both vendors are robust against temperature variations.

TABLE 4. Intra HD at different operating conditions.

Vendor	Memory Bank ID	Operating Codition	ΔV (mV)	ΔT (° C)	Intra HD		Key with Intra HD	
					μ	σ	$> 1\%$	$> 30\%$
A	a	NVRT	0	0	0.48	0.07	0.00	0.00
		LVRT	−20	0	0.05	0.08	0.00	0.00
		HVRT	+55	0	0.07	0.09	0.00	0.00
		NVHT	0	+20	0.06	0.09	0.00	0.00
	b	NVRT	0	0	0.47	3.17	1.57	0.00
		LVRT	−20	0	2.94	10.55	7.81	2.91
		HVRT	+55	0	0.09	0.10	0.00	0.00
		NVHT	0	+20	0.67	3.84	2.34	0.01
	c	NVRT	0	0	0.49	3.34	1.54	0.03
		LVRT	−20	0	7.77	12.38	27.95	0.46
		HVRT	+55	0	0.09	0.12	0.01	0.00
		NVHT	0	+20	0.52	3.38	1.54	0.02
	d	NVRT	0	0	1.54	9.02	4.37	2.74
		LVRT	−20	0	1.69	8.87	8.87	2.66
		HVRT	+55	0	1.47	8.73	4.29	2.64
		NVHT	0	+20	4.72	8.36	9.35	2.62
B	a	NVRT	0	0	1.97	10.25	3.37	3.25
		LVRT	−55	0	2.11	10.19	3.36	3.17
		HVRT	+55	0	1.92	10.02	3.53	3.17
		NVHT	0	+20	2.13	10.23	3.76	3.26
	b	NVRT	0	0	1.93	10.55	3.24	2.62
		LVRT	−55	0	2.22	10.30	5.68	2.52
		HVRT	+55	0	1.95	10.35	3.18	2.53
		NVHT	0	+20	1.99	10.55	3.39	2.74

The rightmost column of Table 4 shows that the robustness of the PUF output improves as we increase the operating voltage for those banks that possess dominant pattern independent '0' cells at the reduced t_{RP}. An increase in the voltage makes these cells more immune to noise. On the other hand, the banks with dominant pattern independent '1' cells show the opposite behavior (i.e., the robustness of the PUF output increases as we reduce the operating voltage). A decrease in the voltage makes these cells more immune to noise. However, the bank *d* from the vendor *A* produces a slight robust output with the change in the voltage (increased or decreased) because this bank produces noisier cells than other banks.

D. PERFORMANCE COMPARISON

1) EVALUATION TIME

We use two approaches to quantify and compare (with other DPUFs) the evaluation time. Eq. (3) (the first approach) and Eq. (4) (the second approach) are used to compare the time overhead required for the Key generation. The first approach measures the time required to receive the response after sending the challenge from the host. The second approach, on the other hand, is intended to measure the required time to produce the key in the evaluation board (t_{exec}).[3]

$$\mathcal{T}_{eval1} \approx t_{host\_send} + t_{exec} + t_{host\_receive} + t_{store} \quad (3)$$

$$\mathcal{T}_{eval2} \approx t_{exec} + t_{host\_receive} \quad (4)$$

[3]The current implementation does not support a separate measurement of t_{exec} and $t_{host\_receive}$.

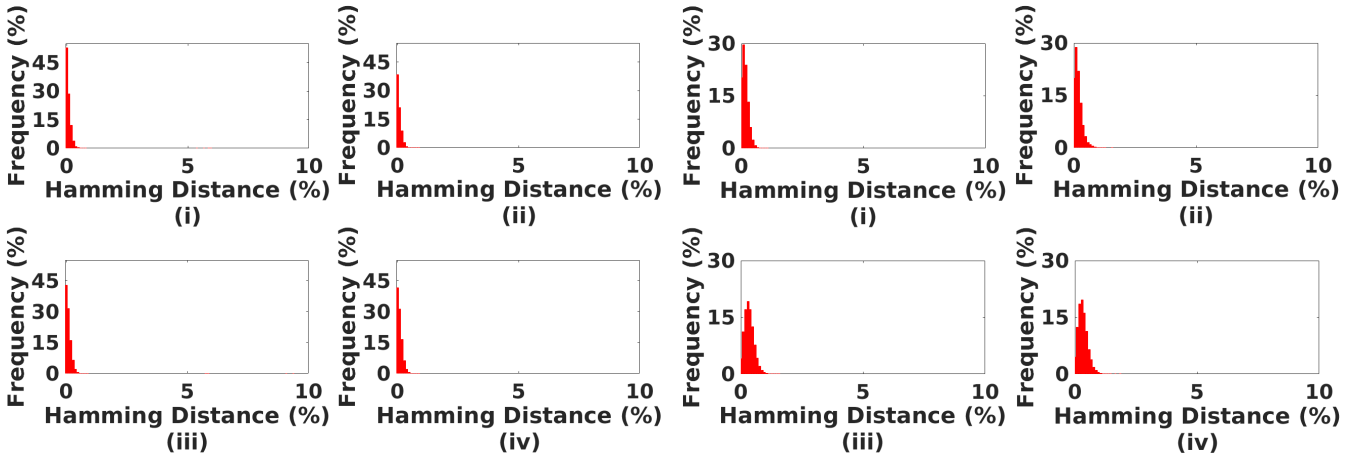


FIGURE 11. Intra HD for the worst case from- (a) vendor *A* (at $t_{RP,PUF} = 2.5$ns), (b) vendor *B* (at $t_{RP,PUF} = 2.5$ns) with (i) NVRT, (ii) LVRT, (iii) HVRT, and (iv) NVHT.

where,
$\mathcal{T}_{eval1}$ = evaluation with the first approach,
$\mathcal{T}_{eval2}$ = evaluation with the second approach,
t_{host_send} = time required to send the command to the evaluation board from the host computer,
t_{exec} = time required to execute the command in the evaluation board,
$t_{host\_receive}$ = time required to send back the read data to the host computer from the evaluation board, and
t_{store} = time required to store the read data to a storage device.

TABLE 5. Average PreLatPUF evaluation time.

Vendor	Memory Bank ID	#Required Burst (mean)	Mean Evaluation time (ms)
A	a	9.00	0.51
	b	6.43	0.41
	c	7.19	0.47
	d	16.10	0.93
B	a	28.15	1.59
	b	24.18	1.34

With the first approach, the worst average time is 1.59ms (worst among all banks, see Table 5), which is 74us with the second approach. However, the evaluation time can be measured more accurately by inserting a local counter inside the FPGA. Note that we did not include the characterization phase in the evaluation time (see Eq. (3) and Eq. (4)) since the cell characterization is performed once during the registration. We, also, did not include the time to write a specific data pattern because we did not consider pattern dependent cells in this paper. However, the writing time needs to be added if pattern dependent cells are considered to generate a large CRP space. The average system-level evaluation time of reduced t_{RCD}-based DPUF is 88.2ms [19], which is still $\sim 1,192X$ slower (considering the worst evaluation time with the second approach) than our proposed method. On the other hand, the retention-based DPUF takes order of minutes to generate a device signature with enough retention failures [13].

2) SYSTEM LEVEL DISRUPTION

For most of the DRAM modules, the granularity of refreshing the DRAM contents is rank. Therefore, we need to increase the refresh interval for entire memory rank for evaluating retention-based DPUF. As a result, it causes random data corruption over the whole rank. Also, due to the long evaluation time of the retention-based DPUF, the particular DRAM rank becomes unavailable for other applications for a long time. In the proposed PreLatPUF, the reduced t_{RP} only affects the cells that are being accessed. We also checked the interference to the neighborhood rows of the target row that is being accessed for key generation. To do so, we arbitrarily selected consecutive 1000 rows from each memory bank. Then, we read the data from all odd-numbered rows at the $t_{RP,PUF}$ and investigated the impact on the memory cells of the even-numbered row with nominal t_{RP}. Our results show that there is no data corruption in the adjacent rows.

However, though the latency-based DPUF of [19] at the reduced t_{RCD} is fast, this type of DPUF evaluation needs a filtering mechanism upon each access, which causes both computational and hardware overheads. In our proposed mechanism, we register the eligible PUF cells once in its entire lifetime (see Section III-C). Once the suitable cells for PUF operation are determined, the evaluation of our proposed PUF is straight-forward (i.e. request the response by sending an address and then compare only the eligible cells' content with the golden data). The proposed PUF evaluation has the least evaluation time. Therefore, the proposed PreLatPUF can be used in run-time, which is impossible in many existing DPUFs [2], [13], [14].

3) ROBUSTNESS

The robustness (i.e., the effect of different operating conditions and environmental variations) of the proposed PreLatPUF is shown in Table 4. The impact of operating voltage and temperature variations in DPUFs have been explored before [13], [19]. In this paper, we compared the robustness between the proposed PreLatPUF and retention-based DPUF at different operating conditions. To accumulate the retention-based failures, we chose a random memory segment with 1000 rows from each bank. At first, we stored logic '1' to all memory cells under the segment, and then the refresh interval was prolonged until we obtained at least ∼2% failures at the NVRT. For a specific bank, same refresh interval was maintained for all other operating conditions. For the proposed PreLatPUF, we measured the data errors with four input patterns (*0xFF*, *0xAA*, *0x55*, and *0x00*) at $t_{RP,PUF}$ for the same 1000 rows. We used the *Jaccard Index* to compare the robustness of our proposed PreLatPUF with the retention-based PUF. For the retention-based DPUF, the PUF characteristics are evaluated from the location of the failed bits. For example, in our case, retention-based failed bits are always failed from logic '1' to logic '0'. But the location of the failed bits differs from one device to another. For the two sets of the measurements (M_1, M_2), the *Jaccard Index* is measured as $\frac{M_1 \cap M_2}{M_1 \cup M_2}$, where $M_1 \cap M_2$ is the total matched failed bits and $M_1 \cup M_2$ is the total failed bits from two measurements M_1 and M_2 [17], [19]. For better reproducibility, the intra *Jaccard Index* should be ∼1. Table 6 shows the comparison between PreLatPUF and retention-based PUF. The results show that the proposed PreLatPUF is more robust than the retention-based DPUF. The retention-based PUF is more susceptible to the temperature variation compared to the PreLatPUF. This is because the retention-based failed bit is mostly emphasized by the charge leakage rate of DRAM cells, which has a strong exponential dependence on the temperature [2], [13], [14], [39]–[41]. On the other hand, the change in t_{RP} is very negligible as temperature changes. The t_{RP} changes only (∼3%) as temperature changes from 27°C to 85°C [25].

TABLE 6. ***Jaccard Index*** **at different operating conditions for the PreLatPUF and the retention-based DPUF.**

Vendor	Memory Bank ID	M_1, M_2	Jaccard Index	
			Proposed PreLatPUF	Retention Based DPUF
A	a	NVRT, LVRT	0.997	0.926
		NVRT, HVRT	0.997	0.968
		NVRT, NVHT	0.997	0.349
	b	NVRT, LVRT	0.980	0.902
		NVRT, HVRT	0.997	0.970
		NVRT, NVHT	0.986	0.356
	c	NVRT, LVRT	0.929	0.930
		NVRT, HVRT	0.997	0.960
		NVRT, NVHT	0.985	0.355
	d	NVRT, LVRT	0.994	0.941
		NVRT, HVRT	0.983	0.968
		NVRT, NVHT	0.996	0.279
B	a	NVRT, LVRT	0.968	0.962
		NVRT, HVRT	0.961	0.847
		NVRT, NVHT	0.968	0.421
	b	NVRT, LVRT	0.965	0.952
		NVRT, HVRT	0.968	0.950
		NVRT, NVHT	0.971	0.457

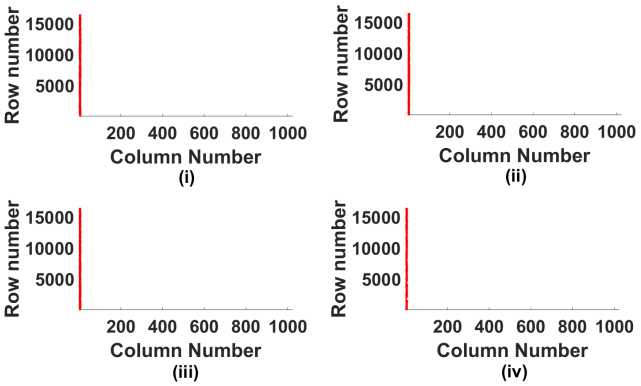


FIGURE 12. **Failed bits at $t_{RCD,5.0}$ with input pattern- (i)** ***0x00***, **(ii)** ***0x55***, **(iii)** ***0xAA***, **and (iv)** ***0xFF***.

For the t_{RCD}-based DPUF, the results shown in [19] suggest that it can tolerate only a small change in temperature (e.g., 5°C). On the other hand, for the PreLatPUF, we increased the temperature by 20° and found a negligible change in the signature. The results presented in [25] also suggest that the temperature dependency of t_{RCD} is stronger than the temperature dependency of t_{RP}.

V. CONCLUSION

In this paper, we proposed a DRAM-based PUF that exploits the precharge-latency variations in DRAM cells. We characterized DRAM cells' errors at the reduce precharge-latency to find the most suitable DRAM cells in order to produce random, unique, and reliable device signatures. The silicon results from commercially available DRAM modules show that the proposed device signature scheme and algorithm can generate robust PUF outputs at a much faster rate.

APPENDIX
IMPACT OF REDUCED *ACTIVATION* TIME

In figure 12, red spots represent the failed bits at the reduced activation time ($t_{RCD,5.0}$) for a DRAM bank. The results show that the failed bits are only observed at the first accessed cache line (i.e., just in the first column). A similar observation was concluded in [19] and [36].

ACKNOWLEDGMENT

The authors would like to thank Hasan Hassan (ETH Zürich) & CMU for the SoftMC software.

REFERENCES

[1] C. Herder, M.-D. Yu, F. Koushanfar, S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proc. IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.
[2] C. Keller, F. Gürkaynak, H. Kaeslin, and N. Felber, "Dynamic memory-based physically unclonable function for the generation of unique identifiers and true random numbers," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Melbourne VIC, Australia, Jun. 2014, pp. 2740–2743.

[3] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[4] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.

[5] A. Hosey, M. T. Rahman, K. Xiao, D. Forte, and M. Tehranipoor, "Advanced analysis of cell stability for reliable SRAM PUFs," in *Proc. IEEE 23rd Asian Test Symp.*, Nov. 2014, pp. 348–353.

[6] M. T. Rahman, D. Forte, Q. Shi, G. K. Contreras, and M. Tehranipoor, "CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2014, pp. 46–51.

[7] G. K. Contreras and M. T. R. M. Tehranipoor, "Secure split-test for preventing IC piracy by untrusted foundry and assembly," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2013, pp. 46–51.

[8] A. Basak, F. Zhang, and S. Bhunia, "PiRA: IC authentication utilizing intrinsic variations in pin resistance," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2015, pp. 1–8.

[9] A. Hennessy, Y. Zheng, and S. Bhunia, "JTAG-based robust PCB authentication for protection against counterfeiting attacks," in *Proc. 21st Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2016, pp. 56–61.

[10] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 3, p. 67, 2017.

[11] B. Halak, M. Zwolinski, and M. S. Mispan, "Overview of PUF-based hardware security solutions for the Internet of Things," in *Proc. IEEE 59th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Oct. 2016, pp. 1–4.

[12] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, and M. M. Prabhu, "Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 3, pp. 424–437, May/Jun. 2019.

[13] S. Sutar, A. Raha, D. Kulkarni, R. Shorey, J. Tew, and V. Raghunathan, "D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication and random number generation," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 1, p. 17, Dec. 2017.

[14] W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Run-time accessible DRAM PUFs in commodity devices," in *Cryptographic Hardware and Embedded Systems–CHES* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2016, pp. 432–453.

[15] F. Tehranipoor, N. Karimian, K. Xiao, and J. Chandy, "DRAM based intrinsic physical unclonable functions for system level security," in *Proc. 25th Ed. Great Lakes Symp. VLSI (GLSVLSI)*. New York, NY, USA: ACM, 2015, pp. 15–20.

[16] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "DRAM-based intrinsic physically unclonable functions for system-level security and authentication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 3, pp. 1085–1097, Mar. 2017.

[17] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Intrinsic Rowhammer PUFs: Leveraging the Rowhammer effect for improved security," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2017, pp. 1–7.

[18] N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, A. Schaller, W. Xiong, M. Jain, M. U. Saleem, J. Lotichius, S. Gabmeyer, J. Szefer, and S. Katzenbeisser, "Intrinsic run-time row hammer PUFs: Leveraging the row hammer effect for run-time cryptography and improved security," *Cryptography*, vol. 2, no. 3, p. 13, 2018.

[19] J. S. Kim, M. Patel, H. Hassan, and O. Mutlu, "The DRAM latency PUF: Quickly evaluating physical unclonable functions by exploiting the latency-reliability tradeoff in modern commodity DRAM devices," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Vienna, Austria, Feb. 2018, pp. 194–207.

[20] A. Mazady, M. T. Rahman, D. Forte, and M. Anwar, "Memristor PUF—A security primitive: Theory and experiment," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 222–229, Jun. 2015.

[21] K. Xiao, M. T. Rahman, D. Forte, Y. Huang, M. Su, and M. Tehranipoor, "Bit selection algorithm suitable for high-volume production of SRAM-PUF," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, May 2014, pp. 101–106.

[22] M. T. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, "ARO-PUF: An aging-resistant ring oscillator PUF design," in *Proc. Conf. Design, Automat. Test Eur. (DATE) Eur. Design Automat. Assoc.*, vol. 3001, Leuven, Belgium, 2014, p. 69.

[23] M. T. Rahman, F. Rahman, D. Forte, and M. Tehranipoor, "An aging-resistant RO-PUF for reliable key generation," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 3, pp. 335–348, Jul./Sep. 2016.

[24] K. K. Chang, A. G. Yağlikçi, S. Ghose, A. Agrawal, N. Chatterjee, A. Kashyap, D. Lee, M. O'Connor, H. Hassan, and O. Mutlu, "Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, p. 10, Jun. 2017.

[25] K. Chandrasekar, S. Goossens, C. Weis, M. Koedam, B. Akesson, N. Wehn, and K. Goossens, "Exploiting expendable process-margins in DRAMs for run-time performance optimization," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2014, p. 173.

[26] D. Lee, S. Khan, L. Subramanian, S. Ghose, R. Ausavarungnirun, G. Pekhimenko, V. Seshadri, and O. Mutlu, "Design-induced latency variation in modern DRAM chips: Characterization, analysis, and latency reduction mechanisms," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*, 2017, p. 7.

[27] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "Investigation of DRAM PUFs reliability under device accelerated aging effects," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[28] K. Kim, I. Chung, D. Sun, S. Rhe, I. Kim, H. Hwang, K. Cho, and G. Jin, "Study on off-state hot carrier degradation and recovery of NMOSFET in SWD circuits of DRAM," in *Proc. IEEE Int. Integr. Rel. Workshop (IIRW)*, Oct. 2016, pp. 91–94.

[29] D. Ganta and L. Nazhandali, "Study of IC aging on ring oscillator physical unclonable functions," in *Proc. 15th Int. Symp. Qual. Electron. Design*, Mar. 2014, pp. 461–466.

[30] M. Hasanuzzaman, S. K. Islam, and L. M. Tolbert, "Effects of temperature variation (300–600 K) in MOSFET modeling in 6H–silicon carbide," *Solid-State Electron.*, vol. 48, no. 1, pp. 125–132, 2004.

[31] Z. Guo, X. Xu, M. T. Rahman, M. M. Tehranipoor, and D. Forte, "SCARe: An SRAM-based countermeasure against IC recycling," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 4, pp. 744–755, Apr. 2018.

[32] M. Hiller, D. Merli, F. Stumpf, and G. Sigl, "Complementary IBS: Application specific error correction for PUFs," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust*, Jun. 2012, pp. 1–6.

[33] M.-D. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 48–65, Jan./Feb. 2010.

[34] J. Kim, M. Sullivan, S.-L. Gong, and M. Erez, "Frugal ECC: Efficient and Versatile Memory Error Protection through Fine-Grained Compression," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, Nov. 2015, pp. 1–12.

[35] Q. Deng, L. Ramos, R. Bianchini, D. Meisner, and T. Wenisch, "Active low-power modes for main memory with MemScale," *IEEE Micro*, vol. 32, no. 3, pp. 60–69, May/Jun. 2012. doi: 10.1109/MM.2012.21.

[36] K. K. Chang, A. Kashyap, H. Hassan, S. Ghose, K. Hsieh, D. Lee, T. Li, G. Pekhimenko, S. Khan, and O. Mutlu, "Understanding latency variation in modern DRAM chips: Experimental characterization, analysis, and optimization," *SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 323–336, Jun. 2016.

[37] D. Lee, Y. Kim, G. Pekhimenko, S. Khan, V. Seshadri, K. Chang, and O. Mutlu, "Adaptive-latency DRAM: Optimizing DRAM timing for the common-case," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2015, pp. 489–501.

[38] *DDR3 SDRAM Standard*, document JESD79-3F, JEDEC Solid State Technology Association, Jul. 2012.

[39] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 60–71, 2013.

[40] H. Hassan, G. Pekhimenko, N. Vijaykumar, V. Seshadri, D. Lee, O. Ergin, and O. Mutlu, "ChargeCache: Reducing DRAM latency by exploiting row access locality," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2016, pp. 581–593.

[41] Y. Katayama, E. J. Stuckey, S. Morioka, and Z. Wu, "Fault-tolerant refresh power reduction of DRAMs for quasi-nonvolatile data retention," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (EFT)*, Albuquerque, NM, USA, Nov. 1999, pp. 311–318.

[42] S. Govindavajhala and A. W. Appel, "Using memory errors to attack a virtual machine," in *Proc. Symp. Secur. Privacy*, May 2003, pp. 154–165.

[43] X. Zhang, Y. Zhang, B. R. Childers, and J. Yang, "Restore truncation for performance improvement in future DRAM systems," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2016, pp. 543–554.
[44] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," in *Proc. 40th Annu. Int. Symp. Comput. Archit. (ISCA)*. New York, NY, USA: ACM, 2013, pp. 1–12.
[45] S. Khan, D. Lee, and O. Mutlu, "PARBOR: An efficient system-level technique to detect data-dependent failures in DRAM," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Toulouse, France, Jun./Jul. 2016, pp. 239–250.
[46] Y. Wang, A. Tavakkol, L. Orosa, S. Ghose, N. M. Ghiasi, M. Patel, J. S. Kim, H. Hassan, M. Sadrosadati, and O. Mutlu, "Reducing DRAM latency via charge-level-aware look-ahead partial restoration," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2018, pp. 298–311.
[47] M. Patel, J. S. Kim, and O. Mutlu, "The reach profiler (REAPER): Enabling the mitigation of DRAM retention failures via profiling at aggressive conditions," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 255–268, 2017.
[48] M. K. Qureshi, D.-H. Kim, S. Khan, P. J. Nair, and O. Mutlu, "AVATAR: A variable-retention-time (VRT) aware refresh for DRAM systems," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2015, pp. 427–437.
[49] J. Kim, M. Patel, H. Hassan, and O. Mutlu, "Solar-DRAM: Reducing DRAM access latency by exploiting the variation in local bitlines," in *Proc. IEEE 36th Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 282–291.
[50] H. Hassan, N. Vijaykumar, S. Khan, S. Ghose, K. Chang, G. Pekhimenko, D. Lee, O. Ergin, and O. Mutlu, "SoftMC: A flexible and practical open-source infrastructure for enabling experimental DRAM studies," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Austin, TX, USA, Feb. 2017, pp. 241–252.
[51] S. Khan, C. Wilkerson, Z. Wang, A. R. Alameldeen, D. Lee, and O. Mutlu, "Detecting and mitigating data-dependent DRAM failures by exploiting current memory content," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2017, pp. 27–40.
[52] B. M. S. B. Talukder, J. Kerns, B. Ray, T. Morris, and M. T. Rahman, "Exploiting DRAM latency variations for generating true random numbers," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–6.
[53] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk*. San Francisco, CA, USA: Morgan Kaufmann, 2010.
[54] Y. Chen, A. B. Kahng, B. Liu, and W. Wang, "Crosstalk-aware signal probability-based dynamic statistical timing analysis," in *Proc. 16th Int. Symp. Qual. Electron. Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers, Mar. 2015, pp. 424–429.
[55] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A model of process variation and resulting timing errors for microarchitects," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 3–13, Feb. 2008.
[56] W. Zhang, W. Yu, Z. Wang, Z. Yu, R. Jiang, and J. Xiong, "An efficient method for chip-level statistical capacitance extraction considering process variations with spatial correlation," in *Proc. Design, Automat. Test Eur.*, 2008, pp. 580–585.
[57] W. Shin, J. Yang, J. Choi, and L.-S. Kim, "NUAT: A non-uniform access time memory controller," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2014, pp. 464–475.
[58] M. T. Rahman, A. Hosey, Z. Guo, J. Carroll, D. Forte, and M. Tehranipoor, "Systematic correlation and cell neighborhood analysis of SRAM PUF for robust and unique key generation," *J. Hardw. Syst. Secur.*, vol. 1, no. 2, pp. 137–155, 2017.
[59] Y. Li, H. Schneider, F. Schnabel, R. Thewes, and D. Schmitt-Landsiedel, "DRAM yield analysis and optimization by a statistical design approach," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 12, pp. 2906–2918, Dec. 2011.
[60] 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," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr./May 2017, pp. 1–4.
[61] D. Ganta and L. Nazhandali, "Easy-to-build arbiter physical unclonable function with enhanced challenge/response set," in *Proc. Int. Symp. Qual. Electron. Design (ISQED)*, 2013, pp. 733–738.
[62] A. Maiti, I. Kim, and P. Schaumont, "A robust physical unclonable function with enhanced challenge-response set," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 333–345, Feb. 2012.
[63] M. T. Rahman, K. Xiao, D. Forte, X. Zhang, J. Shi, and M. Tehranipoor, "TI-TRNG: Technology independent true random number generator," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, pp. 1–6.
[64] C. E. Shannon, "Prediction and entropy of printed English," *Bell Syst. Tech. J.*, vol. 30, no. 1, pp. 50–64, Jan. 1951.
[65] M. Jacobsen, D. Richmond, M. Hogains, and R. Kastner, "RIFFA 2.1: A reusable integration framework for FPGA accelerators," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 8, no. 4, p. 22, Sep. 2015.
[66] *USB Interface Adapter EVM USB-TO-GPIO (ACTIVE)*. Accessed: May 3, 2019. [Online]. Available: http://www.ti.com/tool/usb-to-gpio
[67] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded Systems Design with FPGAs*. New York, NY, USA: Springer, 2012, pp. 245–267.

B. M. S. BAHAR TALUKDER (S'18) received the bachelor's degree from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh. He is currently pursuing the Ph.D. degree in computer engineering with The University of Alabama in Huntsville. His primary research interests include hardware security, secured computer architecture, and emerging memory technologies.

BISWAJIT RAY (S'12–M'16) received the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 2013. He was with SanDisk Corporation, CA, USA, developing 3-D NAND flash memory technology. He is currently an Assistant Professor of electrical and computer engineering with The University of Alabama in Huntsville, AL, USA, where he leads the Hardware Reliability Laboratory. He holds eight U.S. issued patents on non-volatile memory devices and systems, published more than 40 research papers in international journals and conferences. His current research spans the boundaries of electron devices and systems for addressing the challenges in hardware security and reliability.

DOMENIC FORTE (S'09–M'13–SM'18) received the B.S. degree in electrical engineering from the Manhattan College, Riverdale, NY, USA, in 2006, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 2010 and 2013, respectively. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, University of Florida, Gainesville, FL, USA. His current research interests include the domain of hardware security, including the investigation of hardware security primitives, hardware Trojan detection and prevention, electronics supply-chain security, and antireverse engineering. He was a recipient of the Young Investigator Award from the Army Research Office, the NSF CAREER Award, and the George Corcoran Memorial Outstanding Teaching Award from the Electrical and Computer Engineering Department, University of Maryland.

MD TAUHIDUR RAHMAN (S'12–M'18) received the Ph.D. degree from the University of Florida, in 2017, and the master's degree from the University of Connecticut, in 2015. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, The University of Alabama in Huntsville, USA, where he directs the SeRLoP Research Laboratory. His current research interests include hardware security and trust, security of memory organization and modern architectures, embedded security, and reliability. He received the NSF CRII Award, in 2019.