The DRAM Latency PUF:
Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim[†§] Minesh Patel[§] Hasan Hassan[§] Onur Mutlu[§†]
[†]Carnegie Mellon University [§]ETH Zürich

Physically Unclonable Functions (PUFs) are commonly used in cryptography to identify devices based on the uniqueness of their physical microstructures. *DRAM-based PUFs have numerous advantages over PUF designs that exploit alternative substrates: DRAM is a major component of many modern systems, and a DRAM-based PUF can generate many unique identifiers. However, none of the prior DRAM PUF proposals provide implementations suitable for runtime-accessible PUF evaluation on commodity DRAM devices. Prior DRAM PUFs exhibit unacceptably high latencies, especially at low temperatures (e.g., >125.8s on average for a 64KiB memory segment below 55°C), and they cause high system interference by keeping part of DRAM unavailable during PUF evaluation.*

In this paper, we introduce the DRAM latency PUF, a new class of fast, reliable DRAM PUFs. The key idea is to reduce DRAM read access latency below the reliable datasheet specifications using software-only system calls. Doing so results in error patterns that reflect the compound effects of manufacturing variations in various DRAM structures (e.g., capacitors, wires, sense amplifiers). Based on a rigorous experimental characterization of 223 modern LPDDR4 DRAM chips, we demonstrate that these error patterns 1) satisfy runtime-accessible PUF requirements, and 2) are quickly generated (i.e., at 88.2ms) irrespective of operating temperature using a real system with no additional hardware modifications. We show that, for a constant DRAM capacity overhead of 64KiB, our implementation of the DRAM latency PUF enables an average (minimum, maximum) PUF evaluation time speedup of 152x (109x, 181x) at 70°C and 1426x (868x, 1783x) at 55°C when compared to a DRAM retention PUF and achieves greater speedups at even lower temperatures.

1. Introduction

A Physically Unclonable Function (PUF) maps a set of *input parameters* to unique, device-specific signatures that can be generated *repeatably* and *reliably*. We refer to the process of generating a signature using a given set of input parameters as the *evaluation* of a PUF. The resulting signature reflects a device's inherent, random physical variations introduced during manufacturing. This property guarantees that the signature is practically impossible to predict or replicate without access to the device itself [28, 136]. These characteristics enable PUFs to be frequently used in security applications such as low-cost authentication mechanisms against system security attacks and prevention of integrated circuit (IC) counterfeiting [120, 135].

PUFs are generally used in a challenge-response (CR) protocol [120], in which a trusted server gives a device a *challenge* (i.e., a set of input parameters and conditions with which to evaluate a PUF), and verifies the device's *PUF response* (i.e., the signature generated by the PUF). A CR protocol generally consists of two phases: *enrollment* and *authentication*. Enrollment is a one-time setup phase in which a given device is analyzed, and all possible PUF responses are stored in the trusted server. Authentication occurs when an application running on the enrolled device requests escalated permissions from the trusted server to perform a secure action. The server provides a challenge to the application, which then evaluates the PUF with the requested parameters and returns the PUF response. If the response matches with the previously-enrolled response for the challenge, i.e., the *golden key*, authentication is successful. The CR can be done *statically*, where the PUF is evaluated only once before runtime (e.g., at bootup) or at *runtime*, where an application running on the enrolled device can evaluate a PUF on-demand [135].

PUFs for silicon devices were first introduced as a method for integrated circuit (IC) identification, exploiting manufacturing process variation among devices for *disambiguating* different devices [83]. Since then, many prior works have proposed PUF evaluation techniques for different substrates (e.g., ASICs, FPGAs, memories), exploiting manufacturing variation in different components such as emerging memory technologies [45, 64, 107, 128], flash memory [132], Application Specific Integrated Circuit (ASIC) logic [28, 29, 33, 36, 40, 65, 72, 75, 84, 88, 98, 99, 108, 116, 119, 125, 127, 139], Static Random Access Memory (SRAM) [6, 9, 20, 32, 41, 42, 134, 145], and Dynamic Random Access Memory (DRAM) [37, 53, 103, 120, 121, 122, 123, 124].

PUFs must satisfy *five* key characteristics to be effective in security applications [37, 85, 120, 123, 135]. We describe these characteristics in detail in Section 3.1. PUFs satisfying these characteristics 1) guarantee a level of robustness for *disambiguating* many devices and 2) are practically impossible for an attacker to duplicate *without* access to the physical device itself. In addition to these properties, a *runtime-accessible* PUF, i.e., a PUF that is accessible online to an application running on an enrolled device, must 1) be easily evaluated with *low latency* to prevent unnecessary slowdown of the application requesting authentication, and 2) provide *low system interference*, i.e., minimize the disturbance PUF evaluation causes to other applications running on the same system. Section 3.2 describes the characteristics of *ideal runtime-accessible PUFs*.

DRAM-based PUFs, henceforth called *DRAM PUFs*, have recently attracted significant interest for two key reasons: 1) DRAM is already widely used in a wide variety of modern systems [90, 94], ranging from embedded to server, and 2) DRAM's large address space, which is on the order of Giga- or Tera-bytes, makes it naturally suitable for CR applications by providing a greater CR space relative to smaller components (e.g., SRAMs) [6, 9, 20, 32, 33, 41, 42, 134, 145]. Prior DRAM PUF proposals exploit variations in DRAM start-up values [123], DRAM write access latencies [37], and DRAM cell retention failures [53, 82, 120, 135] to generate reliable PUF responses.

Unfortunately, these prior DRAM PUF proposals have significant drawbacks that make them unsuitable as *runtime-accessible* PUFs. PUFs that use DRAM start-up values [123] preclude runtime-accessible PUF evaluation by requiring a DRAM power cycle for *every* authentication. This requires either interrupting other applications using DRAM or restarting the entire system, which is likely infeasible at runtime. On the other hand, PUFs that exploit variation in write access latencies [37] *can be* evaluated at runtime. However, [37]'s proposal requires additional circuitry in a DRAM chip to allow fine-grained manipulation of write latency [37]. This requires changes to DRAM chips, rendering such proposals inapplicable to devices used in the field today. In this paper, we would like to design a new runtime-accessible PUF *without* modifying commodity DRAM chips.

Using cell charge retention *failures* and their resulting *error patterns* [34, 54, 78, 79, 101, 102] is the best candidate for runtime-accessible DRAM PUF evaluation in commodity devices today, since it does *not* require a power cycle or any modifications to DRAM chips. Unfortunately, such *DRAM* *retention PUFs* impose two major overheads. First, due to the 1) wide distribution of charge retention times across DRAM cells [34, 54, 78, 101, 102] and 2) roughly-uniform spatial distribution of retention failures across a chip [7, 115], we find that the evaluation time of a DRAM retention PUF takes on the order of *minutes* at 55°C to identify enough retention failures. The evaluation time increases exponentially as temperature decreases. Second, this means that DRAM refresh *must* be disabled for long periods of time. Because DRAM refresh can *only* be disabled for large regions of DRAM [14], evaluating a DRAM retention PUF on a small region of memory, i.e., a *PUF memory segment*, requires disabling refresh on the *entire* large memory region containing the PUF memory segment. However, to maintain the integrity of data inside the large region but outside of the PUF memory segment, *all* such data must be *continuously* refreshed with additional DRAM commands, which results in significant system interference [135]. Based on extensive experimental analysis using 223 state-of-the-art LPDDR4 DRAM devices, we find that DRAM retention PUFs are too slow for reasonable runtime operation, e.g., they have average evaluation times of 125.8s at 55°C and 13.4s at 70°C using a 64KiB memory segment (Section 5).

Our goal in this work is to develop a new runtime-accessible PUF that 1) uses *existing* commodity DRAM devices, 2) satisfies all characteristics of an effective *runtime-accessible* PUF, and 3) provides *low-latency* evaluation with *low system interference* across *all operating conditions*.

Our key idea is to reduce DRAM read access latency below the reliable datasheet specifications using software-only system calls in order to exploit the resulting error patterns as unique identifiers. We refer to our proposal as *the DRAM latency PUF*. We experimentally demonstrate, using 223 modern LPDDR4 DRAM chips, that the DRAM latency PUF satisfies all of the requirements of an effective runtime-accessible PUF. In particular, a DRAM latency PUF can be evaluated in 88.2ms on average across all devices at all operating temperatures. We show that, for a constant DRAM capacity overhead of 64KiB, the DRAM latency PUF's average (minimum, maximum) evaluation time speedup over the DRAM retention PUF is 152x (109x, 181x) at 70°C and 1426x (868x, 1783x) at 55°C, with exponentially increasing speedups at even lower temperatures.

Our key contributions are as follows:

1. We introduce the DRAM latency PUF, a new class of DRAM PUFs, that is based on the deliberate violation of manufacturer-specified DRAM latency parameters. DRAM latency PUFs can be implemented *with no additional hardware overhead* on any commodity off-the-shelf (COTS) system that permits software-controlled manipulation of DRAM access latencies at the memory controller (e.g., [1, 3, 67]).
2. Using experimental data from 223 real LPDDR4 DRAM chips, we extensively analyze both DRAM latency PUFs and DRAM retention PUFs. We show that DRAM latency PUFs 1) satisfy all characteristics of an effective PUF, and 2) are suitable for use as runtime-accessible PUFs across a *wide range* of temperatures. We also present an extensive characterization of DRAM retention PUFs under a wide range of temperatures. We show that while DRAM retention PUFs can be evaluated faster at higher temperatures, their evaluation time at temperatures even as high as 70°C is *prohibitively* slow.
3. We experimentally show that the DRAM latency PUF significantly outperforms DRAM retention PUFs, achieving an average speedup of 152x/1426x at 70°C/55°C when evaluating PUFs with a constant DRAM capacity overhead of 64KiB. We also find that while DRAM retention PUFs suffer from *temperature-dependent evaluation times*, the DRAM latency PUF provides a consistently low average evaluation time of 88.2ms at all operating temperatures.

2. Background

In this section, we describe the high-level mechanisms enabling DRAM retention and DRAM latency PUFs. We begin by briefly describing DRAM organization, but refer the reader to past works [12, 13, 15, 38, 39, 55, 59, 62, 63, 67, 68, 69, 70, 71, 111, 112, 113, 142] for more detail.

2.1. DRAM Organization

As illustrated in Figure 1a, the lowest level of DRAM organization consists of 2-dimensional arrays of cells in which each cell (shown as a dashed circle) stores one bit of information in a *storage capacitor*. Data is encoded using logic "high" (V_{dd}) as one binary value and logic "low" (V_{ss}) as its inverse. The cell is written to and read from via an *access transistor*. Within the same array, cells in the same *row* are connected by a wire called the *wordline*, and cells of the same *column* are connected by a wire called the *bitline*. The sizes of the arrays and other hierarchical elements are manufacturer dependent [62, 69].

Many of these 2-dimensional arrays are tiled to create a larger array referred to as a *bank* (Figure 1b). Banks are striped across a set of *chips*, which together form a single *rank* that operates in unison (Figure 1c). At the highest level of the hierarchy, a *channel* consists of one or more ranks sharing a single command, address, and data interface to a *memory controller* (Figure 1d).

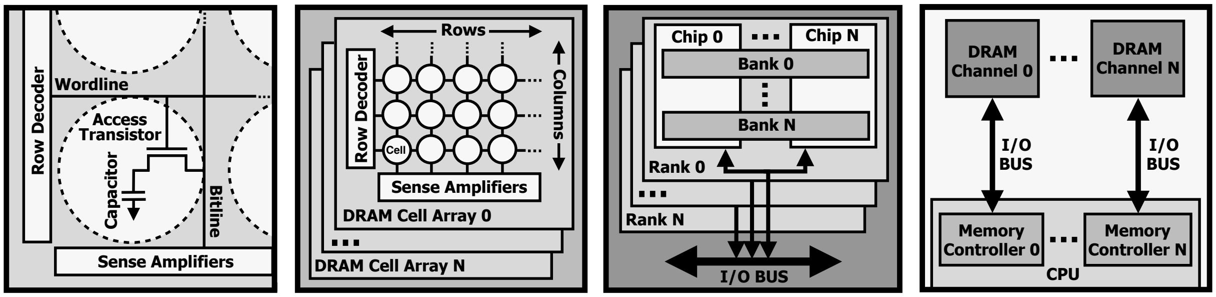
The memory controller interfaces with a single DRAM rank at a time and is responsible for sending commands and receiving data over the buses associated with that channel. To access a cell (i.e., to read from or to write to it), the row containing the cell must first be *activated*. To activate a row, the memory controller sends an *ACT* command along with a row address to a target DRAM rank. In response, the target rank asserts every wordline corresponding to the requested row address in the corresponding bank across all chips. The activation process causes the row's stored data to be transferred to the row's *sense amplifiers*. At this point, the row is considered to be *open*. Only then can the memory controller read out the data with a *RD* command or write to it with a *WR* command. After all read or write operations to the open row are complete, the memory controller can *precharge* the bank containing the row using a *PRE* command, which *closes* the row.

2.2. DRAM Refresh and DRAM Retention PUFs

Over time, charge leaks from a DRAM cell [58, 78, 79]. To keep data intact, DRAM cells must be periodically *refreshed*. To achieve this, the memory controller sends a *REF* command at regular intervals to instruct a DRAM rank to refresh its cells. Because the refresh operation is handled *internally* to the DRAM chip [14, 47, 48, 49] for modern DRAM devices, there is no efficient way to *selectively* refresh DRAM regions smaller than the granularity provided by the refresh operation. For current DDR DRAM, the granularity is one rank, but for LPDDR DRAM [48, 49], a finer granularity of one bank is available [14].

Many prior works show that the rate of charge leakage across different cells 1) varies widely due to manufacturing process variation and 2) changes at runtime depending on many factors, including the device's operating temperature [34, 54, 78, 79, 101], operating voltage [16], and the data values stored in the device itself [54, 55, 56, 57, 59, 73, 74, 78, 101]. For modern DDR DRAM devices, JEDEC specifies a *refresh interval* of 64ms to account for the worst case cell at 85°C [50].

If a cell leaks too much charge before being refreshed, the stored value can be corrupted and a read to the cell may result



(a) DRAM Cell Array **(b)** DRAM Bank **(c)** DRAM Channel **(d)** DRAM-Based System

Figure 1: DRAM organization.

in incorrect data. This is referred to as a *retention failure*, and the period of time that a cell can retain correct data is referred to as the cell's *retention time*. Prior work shows that 1) process variation results in a wide distribution of cell retention times across a single DRAM chip [34, 51, 54, 58, 74, 78, 79, 101], and 2) for any given refresh interval, the spatial distribution of retention failures is distributed roughly uniform-randomly across a chip [7, 34, 102, 115].

Several prior works [53, 120, 135] exploit these data retention properties of DRAM cells to devise PUFs (called *DRAM retention PUFs*) that are evaluated by analyzing the distribution of charge retention times across a chip. Ideally, such a PUF evaluation would consist of *measuring* each cell's rate of charge leakage within a specified PUF memory segment. However, because this is a complex and time-consuming procedure, prior proposals [53, 82, 109, 120, 122, 135] rely on simply determining the set of cells that fail at a longer refresh interval. A longer refresh interval results in a set of cells that is unique to a chip, and given a *large-enough* PUF memory segment or a *long-enough* refresh interval, the magnitude of the set of failures becomes large enough to satisfy the characteristics of an effective PUF. Section 5 presents our experimental evaluation of DRAM retention PUFs on modern LPDDR4 DRAM devices. Section 8 provides a more comprehensive description of the different proposals for various DRAM retention PUFs.

2.3. DRAM Operation

The timing of DRAM commands is guided by a set of manufacturer-specified *timing parameters* [13, 16, 50, 62, 67, 69, 70], which account for the latency of different circuit-level DRAM operations. These timing parameters are provided to guarantee correct DRAM operation, and it is up to the memory controller to obey them. If the memory controller violates a timing parameter, correct DRAM operation is no longer guaranteed, and thus data loss or corruption can occur [11, 13, 16, 67, 69]. Our proposal, the DRAM latency PUF, exploits this behavior to *deliberately* cause DRAM timing-related failures and uses the resulting error patterns as unique identifiers.

2.3.1. DRAM Timing Parameters. We examine the key timing parameters governing DRAM access. DRAM reads and writes consist of three major sequential steps: 1) activation, 2) read/write, and 3) precharge, each of which is defined as a *DDR command* by the JEDEC DDR specification [50].

As detailed in Section 2.1, the ACT command opens a row and prepares it for accesses. The timing parameter t_{RCD} governs the amount of time required for the activation process. This means that after issuing an ACT command to a row, the memory controller must wait for a delay of t_{RCD} before issuing a subsequent RD or WR command to the row. This delay allows time for 1) the internal DRAM circuitry to assert the correct wordline, 2) the cell capacitors to share charge with their respective bitlines, and 3) the sense amplifiers to finish sensing and capturing the values stored in the cells. Violating t_{RCD} can result in insufficient time for *any* of these internal processes to complete, and thus result in incorrect operation or incorrect data to be read [13, 69].

The *RD* and *WR* commands are responsible for reading from and writing to the open DRAM row and are governed by a number of different timing parameters (e.g., t_{CL}, t_{CWL}, t_{RAS}). These parameters ensure that enough time passes after the RD/WR command is issued such that the memory controller can reliably read data stored in the sense amplifiers or reliably write data into the DRAM cells [62, 70].

The *PRE* command initiates the precharge operation, and it is governed by the t_{RP} timing parameter. This parameter allows sufficient time for closing the currently-open row and re-initializing the bitlines.

Additional timing parameters (e.g., t_{WR}, t_{WTR}, t_{RTW} [62, 66, 70]) govern other DDR commands. In general, each parameter ensures that enough time has passed after a certain action such that DRAM operates correctly and provides data reliably. The memory controller is responsible for scheduling DRAM commands according to these timing parameters in order to maintain correct and reliable device operation [4, 44, 66, 93, 106, 117, 118, 126].

2.3.2. Violating Manufacturer-Specified Timing Parameters. Different cells in the same DRAM chip have different reliable operation latencies (for each timing parameter) due to two major reasons: 1) design (architectural) differences [69], and 2) process variation [67]. For example, a cell located closer to the sense amplifiers than an otherwise-equivalent cell can operate correctly with a lower t_{RCD} constraint [69] because the inherent latency to access a cell close to the sense amplifiers is lower. Similarly, a cell that happens to have a larger capacitor (due to manufacturing process variation) can operate reliably with tighter timing constraints than a smaller cell elsewhere in the same chip [67].

Because manufacturing process variation occurs in random and unpredictable locations within and across chips [12, 13, 16, 25, 58, 59, 67, 68, 69, 74, 143], the manufacturer-published timing parameters are chosen to ensure reliable operation of the *worst-case* cell in any acceptable device at the *worst-case* operating conditions (e.g., highest supported temperature, lowest supported voltage). This results in a large *safety margin* (or, *guardband*) for each timing parameter, which prior work shows can often be reliably reduced at *typical* operating conditions [11, 13, 67].

Prior work also shows that decreasing the timing parameters *too aggressively* results in failures, with increasing error rates observed for larger reductions in timing parameter values [13, 16, 38, 39, 54, 55, 56, 57, 67, 78, 101, 102]. Errors occur because, with reduced timing parameters, the internal DRAM circuitry is *not* allowed *enough time* to properly perform its functions and stabilize outputs before the memory controller issues the next command (Section 2.3.1). The DRAM latency PUF exploits the resulting error patterns to uniquely identify a device. In Section 6, we show experimental data demonstrating how to generate unique signatures with aggressively reduced DRAM timing parameters.

3. Properties of a Runtime-Accessible PUF

In this section, we examine the desired properties of an *effective runtime-accessible PUF*. Prior works present various different metrics for defining the effectiveness of a PUF [37, 85, 120, 123, 135]. We consolidate these metrics into five key properties below. We then discuss two properties that we consider necessary for an effective *runtime-accessible* PUF. We refer to these seven properties when analyzing DRAM PUFs (Section 5 and 6). In Section 6, we show how DRAM latency PUFs overcome the weaknesses of DRAM retention PUFs based on a comparison of these properties between the two types of PUFs.

3.1. Characteristics of a Desirable PUF

The following five key properties must be provided by any effective *PUF* that can be evaluated across a set of devices:

1. *Diffuseness*: a single device is able to generate many unique and independent responses to different input parameters [6, 22, 27, 43].
2. *Uniqueness*: a single device can be uniquely identified among the set of devices [37, 43, 77, 123, 133, 135].
3. *Uniform Randomness*: all possible PUF responses must be equally different from each other [43, 77, 123, 135].
4. *Unclonability*: it should be practically impossible for an adversary to construct a device that exhibits the same properties as another [52, 85, 133].
5. *Repeatability*: given a set of input parameters, PUF evaluation results in the same PUF regardless of external conditions (e.g., temperature, aging) [37, 43, 77, 123, 133, 135].

These five properties ensure that a PUF can be used effectively for challenge-response authentication.

3.2. Characteristics of a *Runtime-Accessible* PUF

There are many important use cases for runtime-accessible PUFs. Examples include 1) systems that employ remote communication protocols to access devices via remote direct memory access (RDMA [2]) or to perform functions on remote devices (e.g., remote servers), 2) systems that have interchangeable/broken system components (e.g., SSD drives, external sensors, peripheral devices). In each of these systems, a connection/component can be maliciously swapped out *during runtime* so that a *malicious* device can be swapped in. One way to avoid such an attack is to utilize a low-overhead runtime-accessible PUF-based challenge-response mechanism that frequently authenticates the communicating devices. This enables re-authentication of the system components during each step of communication rather than just once at bootup time. More generally, a fast *runtime-accessible* PUF enables the protection of a system from attacks that exploit the fact that the time of check is different from the time of use [135].

In order to be useful for *runtime* authentication, a PUF must be effectively *usable* while the system is running *without* significantly interfering with application execution and system operation. Thus, a runtime-accessible PUF must possess the following two key properties:

1. *Low Latency*: PUF evaluation must be fast so that the application requesting authentication stalls for the *smallest possible amount of time.*
2. *Low System Interference*: PUF evaluation must *not* significantly slow down concurrently-running applications.

4. Testing Environment

To analyze DRAM behavior with both reduced refresh rates and reduced timing parameters, we developed an infrastructure to characterize modern LPDDR4 [50] DRAM chips. Our testing environment gives us precise control over the DRAM commands and DRAM timing parameters as verified with a logic analyzer probing the command bus.

We perfom all tests, unless otherwise specified, using a total of 223 2y-nm LPDDR4 DRAM chips from three major manufacturers in a thermally-controlled chamber held at 45°C. For consistency across results, we stabilize the ambient temperature precisely using heaters and fans controlled via a microcontroller-based proportional-integral-derivative (PID) loop to within an accuracy of 0.25°C and a reliable range of 40°C to 55°C. We maintain DRAM temperature at 15°C above ambient temperature using a separate local heating source. We utilize temperature sensors to smooth out temperature variations caused by self-induced heating.

5. DRAM Retention PUFs: Analysis

Recent works [103, 120, 121, 124, 135] propose DRAM retention PUFs, which require no modifications to commodity DRAM chips (Section 2.2). These works evaluate their proposals using DDR3 DRAM modules and find that while the use of charge retention times in DRAM cells can result in repeatable PUFs, delays on the order of *minutes* are required to produce enough failures for uniquely identifying many devices.

In this section, we evaluate prior proposals using our own infrastructure with 223 modern LPDDR4 DRAM modules. Our experimental results (Section 5.2) confirm that DRAM retention PUFs can be effectively implemented with commodity LPDDR4 DRAM devices. However, similarly to prior work [120, 135], we find that the time required to evaluate retention PUFs is prohibitively long (e.g., on the order of minutes) *at temperatures* that are likely encountered under common-case operating conditions (e.g., 35°C-55°C) [19, 26, 67].

5.1. Evaluating Retention PUFs

We evaluate DRAM retention PUFs on our modern LPDDR4 devices, as shown in Algorithm 1. The DRAM retention PUF disables refresh for a period indicated by the *wait_time* input parameter on a memory segment indicated by the segment ID (*seg_id*) input parameter (line 3). In order to constrain retention failures to the PUF memory segment, the user must refresh the rows contained in the DRAM rank, but *not* in the PUF memory segment during the *wait_time* interval (line 5-8). The resulting data in the memory segment after the *wait_time* interval is the PUF response that is returned for authentication (line 10). This PUF response is uniquely represented by the pattern of DRAM cells that fail in the memory segment after *not* being refreshed during the *wait_time* interval.

As discussed in Section 2.2, the memory controller can disable refresh only at the granularity of DRAM ranks or banks [50]. Therefore, in order to prevent potential data loss, evaluation of a *runtime-accessible* DRAM retention PUF using a given DRAM memory segment *requires* continuous refreshing of all rows that are within the same rank or bank but outside of the PUF memory segment. Doing so results in high system interference (see, e.g., [14, 79]) that is greatly exacerbated by the long refresh intervals (e.g., *60s* vs. the standard *64ms*) required for repeatable retention PUF evaluation at common-case temperatures (e.g. 35°C-55°C).

5.2. Evaluation Times of Retention PUFs

In this section, we explore the effects of DRAM temperature during DRAM retention PUF evaluation on the DRAM retention PUF evaluation time. Based on extensive experimental data from 223 LPDDR4 DRAM chips, we find that the evaluation time of a DRAM retention PUF exhibits a strong dependence on DRAM temperature during evaluation. With even just a 10°C decrease in DRAM temperature, the evalu-

Algorithm 1: Evaluate Retention PUF [103, 120, 121, 124, 135]

```
evaluate_DRAM_retention_PUF(seg_id, wait_time):
  rank_id ← DRAM rank containing seg_id
  disable refresh for Rank[rank_id]
  start_time ← current_time()
  while current_time() - start_time < wait_time:
    foreach row in Rank[rank_id]:
      if row not in Segment[seg_id]:
        issue refresh to row                    // refresh all other rows
  enable refresh for Rank[rank_id]
  return data at Segment[seg_id]
```

ation time for the *same* PUF memory segment increases by 10x [120, 135]. This is due to the direct correlation between retention failure rate and temperature. We reproduce the *bit error rate* (BER) vs. temperature relationship studied for DDR3 [78] and LPDDR4 [101] chips using our own LPDDR4 chips. We find that below refresh intervals of 30s, there is an exponential dependence of BER on temperature with an average exponential growth factor of 0.23 per 10°C. This results in approximately a 10x decrease in the retention failure rate with every 10°C decrease in temperature and is consistent with prior work's findings with older DRAM chips [78, 101, 120]. Due to the sensitivity of DRAM retention PUFs to temperature, a *stable temperature* is required to generate a *repeatable* PUF response.

To find the evaluation time of DRAM retention PUFs, we use a similar methodology to prior works on DRAM retention PUFs, which disable DRAM refresh and wait for at least 512 retention failures to accumulate across a memory segment [53, 120]. Figure 2 shows the results of DRAM retention PUF evaluation times for three different memory segment sizes (8KiB, 64KiB, 64MiB) across our testable DRAM temperature range (i.e., 55°C-70°C). Results are shown for the average across all tested chips from each manufacturer in order to isolate manufacturer-specific variation [54, 78, 79, 101]. Figure 2 also shows, for comparison, the DRAM latency PUF evaluation time, which is experimentally determined to be 88.2ms on average for *any* DRAM device at *all* operating temperatures (see Section 6.2.1).

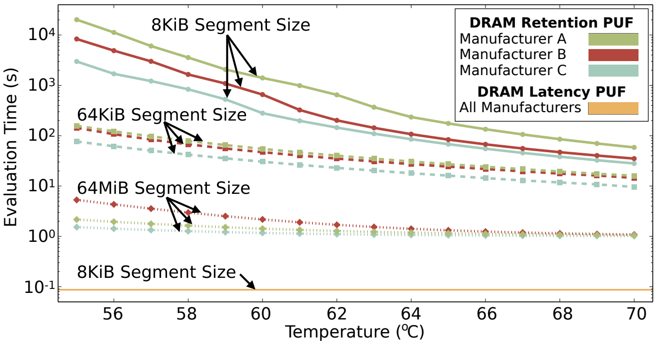


Figure 2: Average DRAM retention PUF evaluation time vs. temperature shown for three selected memory segment sizes for each manufacturer. Average DRAM latency PUF evaluation time (Section 6.2.1) is shown as a comparison point.

We find that at our maximum testing temperature of 70°C, the average DRAM retention PUF across all manufacturers can be evaluated on average (minimum, maximum) in 40.6s (28.1s, 58.6s) using an 8KiB segment size. By increasing the memory segment size from 8KiB to 64KiB, we can evaluate a DRAM retention PUF in 13.4s (9.6s, 16.0s), and at 64MiB, in 1.05s (1.01s, 1.09s). However, at our lowest testable temperature (i.e., 55°C), DRAM retention PUF evaluation time increases to 2.9 *hours* (49.7 minutes, 5.6 *hours*) using an 8KiB segment, 125.8s (76.6s, 157.3s) using a 64KiB segment, and 3.0s (1.5s, 5.3s) using a 64MiB segment.[1]

A DRAM retention PUF evaluation time on the order of even seconds or minutes is *prohibitively high* for at least three reasons: 1) such high latency leads to very long application stall times and very high system interference, 2) since DRAM refresh intervals can be modified only at a rank/bank granularity, the memory controller must continuously issue *extra accesses*, during PUF evaluation, to each row inside the rank/bank but outside of the PUF memory segment, which causes significant bandwidth performance and energy overhead, and 3) such a long evaluation time allows ample opportunity for temperature to fluctuate, which would result in a PUF response with low similarity to the golden key, and thus, an unreliable PUF.

In general, DRAM retention PUF evaluation time increases with *decreasing* temperature. This is due to the temperature dependence of charge leakage in DRAM cell capacitors, and is a *fundamental limitation* of using DRAM retention failures as a PUF mechanism. Therefore, any devices operating at common-case operating temperatures (35°C-55°C) [26, 67, 81] or below will have great difficulty adopting DRAM retention PUFs for runtime accessibility. In Sections 6.1 and 7.2, we describe the DRAM latency PUF in detail and show how it 1) provides a much lower evaluation time than the DRAM retention PUF, and 2) enables a reliably short evaluation time across *all* operating temperatures.

5.3. Optimizing Retention PUFs

We explore if it is possible to make DRAM retention PUFs runtime-accessible (i.e., significantly faster) at common-case operating temperatures by increasing the rate at which retention failures are induced. Given that ambient (i.e., environmental) temperature is fixed, we can increase the rate of induced retention failures in two ways: 1) using a larger PUF memory segment in DRAM, or 2) accelerating the rate of charge leakage using means other than increasing ambient temperature.

Larger PUF memory segments. Using a larger PUF memory segment results in additional DRAM capacity overhead that does *not* scale favorably with decreasing temperatures. As shown in Section 5.2, the number of retention failures drops exponentially with temperature, so the PUF memory segment size required to compensate for the decreasing retention failure rate *increases exponentially.* Our experimental analysis in Figure 2 shows that at 55°C, even using a PUF memory segment size on the order of tens of megabytes, a DRAM retention PUF *cannot* be evaluated in under 1 second. Assuming the exponential growth factor of 0.23 for DRAM BER as a function of temperature (found in Section 5.2), a corresponding PUF evaluation time of ~1s at 20°C would require a PUF memory segment over a thousand times larger (i.e., hundreds of gigabytes). Thus, it is not cost-effective (i.e., scalable) to naïvely increase the PUF memory segment size.

Accelerating charge leakage. Accelerating charge leakage given a fixed temperature can be done by either 1) making hardware modifications or 2) exploiting factors other than temperature that affect charge leakage. Unfortunately, as we discuss in this section, there is no easy way to achieve these using commodity off-the-shelf (COTS) systems.

In-DRAM hardware modifications proposed in prior work can be leveraged to increase the number of retention failures observed at a fixed ambient temperature. For example, partial

[1] These evaluation times are consistent with prior work on DRAM retention PUFs [53, 120, 135], which find that evaluation times on the order of minutes or longer are required to induce enough retention failures in a 128KiB memory segment to generate a PUF response at 20°C.

restoration of DRAM cells [67, 144] can be used to prepare the PUF memory segment with reduced charge levels in order to exacerbate the number of retention failures observed with a given refresh interval. Similarly, other mechanisms in prior work (e.g., [39, 114]) can be used to decrease DRAM retention PUF evaluation time at common-case temperatures where DRAM retention PUFs are otherwise infeasible. However, these approaches require modifications in DRAM or the memory controller, and thus, cannot be used in COTS DRAM.

System-level hardware modifications, such as adding a heating source local to the DRAM chip [30], could be used to exacerbate the occurrence of retention failures at low ambient temperatures. However, these approaches require custom system architectures, which contradicts our goal of designing a PUF for COTS systems. They may also open up system security and reliability concerns.

Experimental studies on DRAM have shown that charge leakage rates are dependent on factors such as supply voltage [16], data pattern effects [54, 55, 56, 57, 73, 78, 101], and random charge fluctuations known as *variable retention time* (*VRT*) [54, 78, 101, 102, 104, 137]. Analogously to temperature control, any of these quantities could be intelligently manipulated to exacerbate the number of retention failures observed. Unfortunately, these effects are either *relatively weak* to significantly increase the number of observed retention failures (e.g., data pattern dependence), require *system modifications* to implement (e.g., voltage control [16, 24]), or are inherently *difficult to control* (e.g., VRT effects).

In order to reduce the number of extra row refresh operations necessary to prevent data loss throughout retention PUF evaluation (Section 5.1), DRAM refresh optimizations proposed in prior work [7,10,21,76,78,79,95,101,102,130,131] can be used to increase the granularity of the refresh operation. While this approach could potentially eliminate the extra refresh operations altogether, these mechanisms come with their own hardware and runtime overheads that may diminish the benefits of *not* having to issue the extra refresh commands during PUF evaluation. Many such mechanisms also require hardware modifications to either DRAM chips or memory controllers or both.

We conclude that there is no good known way to optimize DRAM retention PUF evaluation time for COTS DRAM devices today. While many approaches to improve evaluation time exist, they are all impractical in COTS systems due to 1) lack of applicability and scalability to common-case temperatures, 2) need for DRAM modification, or 3) inherent difficulties in control. This motivates the need for a runtime-accessible PUF that is suitable across all temperature conditions and can be implemented on COTS DRAM devices today.

6. DRAM Latency PUFs

Our goal is to develop a DRAM PUF that can be evaluated 1) with low latency and low system interference across all operating temperatures, and 2) without any modification to DRAM chips. To this end, we present the DRAM latency PUF, a new class of DRAM PUFs with these characteristics. In particular, a DRAM latency PUF provides low evaluation time at a wide range of operating temperatures (0°C-70°C), which includes common-case temperatures (35°C-55°C) [26, 67, 81].

Key Idea. The key idea of the DRAM latency PUF is to provide unique device signatures using the error pattern resulting from accessing DRAM with *reduced* timing parameters. These *latency failures* are inherently related to chip-specific random process variation introduced during manufacturing (Section 2.3), which allows us to use the failures as unique identifiers for each DRAM chip. To evaluate a DRAM latency PUF, we write known data into a fixed-size *memory segment* (e.g., 4 DRAM rows $\approx$ 8KiB in our LPDDR4 DRAM chips) and read it back with reduced timing parameters. The resulting failures form a pattern of bits unique to the tested device.

Probabilistic Nature. Inducing latency failures is a stochastic process in which the probability of cell failure is based on random variations in both the cell itself and any peripheral circuitry used to access the cell. This is due to the probabilistic behavior of circuit elements when timing requirements are violated. To find a repeatable set of latency-failure-prone DRAM cells, each cell should be accessed *multiple* times with reduced timing parameters. In the case of reduced t_{RCD}, we require multiple *iterations* of reading each cell to accumulate a reliable set of latency failures. Fortunately, as we show in Section 6.2.1, finding a reliable set of latency failures is a relatively fast process (i.e., it takes 88.2ms on average).

Key Variables. We identify three key variables to optimize for when designing the DRAM latency PUF. These variables define the tradeoffs between the DRAM latency PUF's evaluation time and its effectiveness.

1) *Memory segment ID.* DRAM PUFs can be evaluated using memory segments from different parts of DRAM. Each segment results in unique error patterns and can therefore be used for *different* challenge-response pairs. In Section 7.3, we discuss how variation in process manufacturing causes some chips to have fewer memory segments that are viable for DRAM latency PUF evaluation than others.

2) *Memory segment size.* Larger memory segments allow more devices to be uniquely identified at the cost of higher PUF evaluation time because more memory accesses are required to induce latency failures across the memory segment. With an experimental analysis of memory segment size based on data from 223 real DRAM chips (Section 6.1), we find that a memory segment size of 8KiB is sufficient to find enough latency failures for an effective DRAM latency PUF.

3) *DRAM timing parameters.* Both using different timing parameters and changing the amount of reduction in the chosen timing parameter result in *different* error patterns (Section 2.3.1). This is because 1) different timing parameters guard against different underlying error mechanisms [16, 67, 69], and 2) different amounts of latency reduction exercise different failure-prone bits [67]. These two dimensions of control add more degrees of freedom to the DRAM latency PUF, further increasing its diffuseness (Section 6.1.1).

Throughout the rest of this section, we first demonstrate that the DRAM latency PUF satisfies all requirements for 1) a reliable PUF (Section 3.1) and 2) runtime-accessible PUF evaluation (Section 3.2) across all temperatures. We focus on t_{RCD}-induced DRAM read errors in this work, but DRAM latency PUFs also work with any other timing parameter whose timing violation results in failures (e.g., t_{RP}, t_{RAS}, t_{WR}), thereby enabling a potentially larger challenge-response space than obtained by using a single timing parameter alone.

6.1. PUF Characteristics: Experimental Analysis

This section shows, with experimental results from 223 state-of-the-art LPDDR4 DRAM chips, that the DRAM latency PUF satisfies each of the five characteristics of a desirable PUF discussed in Section 3.1.

6.1.1. Diffuseness. Different memory segments within the same device result in different error patterns [13, 16, 67, 69]. Given the large address space provided by modern DRAM, different memory segments provide different challenge-response pairs. For example, our selected segment size of 8KiB (Section 7.3) in a 2GiB DRAM, offers up to 256K ($\frac{2GiB}{8KiB}$) different challenge-response pairs, which is on the same order of magnitude as prior DRAM PUFs [120, 135].

6.1.2. Uniqueness and Uniform Randomness. To show the uniqueness and uniform randomness of DRAM latency PUFs evaluated across different memory segments, we study a large number of different memory segments from each of our 223 LPDDR4 DRAM chips (as specified in Table 1).

	#Chips	#Tested Memory Segments
A	91	17,408
B	65	12,544
C	67	10,580

Table 1: The number of tested PUF memory segments across the tested chips from each of the three manufacturers.

For each memory segment, we evaluate the PUF 50 times at 70°C. To measure the uniqueness of a PUF, we use the notion of a *Jaccard index* [46], as suggested by prior work [5, 109, 135]. We use the Jaccard index to measure the similarity of two PUF responses. The Jaccard index is determined by taking the two sets of latency failures (s_1, s_2) from two PUF responses and computing the ratio of the size of the shared set of failures over the total number of unique errors in the two sets $\frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$. A Jaccard index value closer to 1 indicates a high similarity between the two PUF responses, and a value closer to 0 indicates uniqueness of the two. Thus, a unique PUF should have Jaccard index values close to 0 across all pairs of *distinct* memory segments.

We choose to employ the Jaccard index instead of the *Hamming distance* [35] as our metric for evaluating the similarity between PUF responses because the Jaccard index places a heavier emphasis on the differences between two large bitfields. This is especially true in the case of devices that exhibit inherently lower failure rates. In the case of Hamming distance, calculating similarity between two PUF responses depends heavily on the number of failures found, and we find this to be an unfair comparison due to the large variance in the number of failures across distinct memory segments. For example, consider the case where two memory segments each generate PUF responses consisting of a single failure in different locations of a bitfield comprised of 100 cells. The Hamming distance between these PUF responses would be 1, which could be mistaken for a match, but the Jaccard index would be calculated as a 0, which would guarantee a mismatch. Because we are more interested in the locations *with* failures than without, we use the Jaccard index, which discounts locations without failures. Throughout the rest of this paper, we use the terms 1) *Intra-Jaccard* [109, 135] to refer to the Jaccard index of two PUF responses from the *same* memory segment and 2) *Inter-Jaccard* [109, 135] to refer to the Jaccard index of two PUF responses from *different* memory segments.

A PUF must exhibit uniqueness and uniform randomness across any memory segment from any device from any manufacturer. To show that these characteristics hold for the DRAM latency PUF, we ensure that the distribution of Inter-Jaccard indices are distributed near 0. This demonstrates that 1) the error patterns are unique such that no two distinct memory segments would generate PUF responses with high similarity, and 2) the error patterns are distributed uniform randomly across the DRAM chip(s) such that the likelihood of two chips (or two memory segments) generating the same error pattern is exceedingly low.

Figure 3 plots, in blue, the distribution of Inter-Jaccard indices calculated between *all possible pairs* of PUF responses generated at the same operating temperature (70°C) from all tested memory segments across all chips from three manufacturers. The distribution of the Intra-Jaccard indices are also shown in red (discussed later in this section). The x-axis shows the Jaccard indices and the y-axis marks the probability of any pair of memory segments (either within the same device or across two different devices) resulting in the Jaccard index indicated by the x-axis. We observe that the distribution of the Inter-Jaccard indices is multimodal, but the Inter-Jaccard index *always* remains below 0.25 for *any pair* of distinct memory segments. This means that PUFs from different memory segments have low similarity. Thus, we conclude that latency-related error patterns approximate the behavior of a desirable PUF with regard to both uniqueness and uniform randomness.

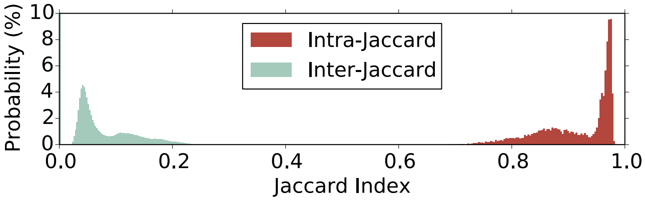


Figure 3: Distributions of Jaccard indices calculated across every possible pair of PUF responses across all tested PUF memory segments from each of 223 LPDDR4 DRAM chips.

To understand manufacturer-related effects, Figure 4 separately plots the Intra- and Inter-Jaccard distributions of PUF responses from chips of a *single* manufacturer in subplots. Each subplot indicates the manufacturer encoding in the top left corner (A, B, C). From these per-manufacturer distributions, we make three major observations: 1) Inter-Jaccard values are quite low, per-manufacturer, which shows uniqueness and uniform randomness, 2) there is variation across manufacturers, as expected, and 3) Figure 3's multimodal behavior for Inter- and Intra-Jaccard index distributions can be explained by the mixture of per-manufacturer distributions. We also find that the distribution of Inter-Jaccard indices calculated between two PUF responses from chips of distinct manufacturers are tightly distributed close to 0 (not shown).

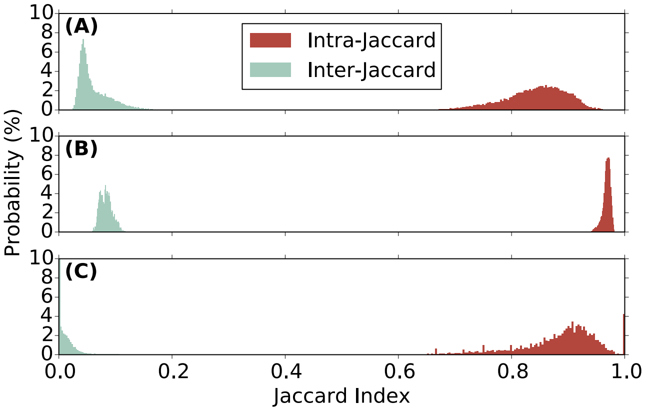


Figure 4: Distributions of Jaccard indices calculated between PUF responses of DRAM chips from a single manufacturer.

6.1.3. Unclonability. We attribute the probabilistic behavior of latency failures to physical variation inherent to the chip (discussed in Section 2.3.2). Chips of the same design contain physical differences due to manufacturing process variation which occurs as a result of imperfections in manufacturing [12, 13, 16, 59, 67, 68, 69]. The exact physical variations are inherent to each individual chip, as shown by previous work [12, 13, 16, 59, 67, 68, 69] and confirmed by our experiments (not shown), and the pattern of variations is very difficult to replicate as it is created entirely unintentionally.

6.1.4. Repeatability. To demonstrate that the DRAM latency PUF exhibits repeatability, we show how well a PUF memory segment can result in the *same* PUF response 1) at different times or 2) under different operating temperatures. For each of many different memory segments, we evaluate a PUF multiple times and calculate all possible *Intra-Jaccard* indices (i.e., Jaccard indices between two PUF responses generated from the *same* exact memory segment). Because a highly-repeatable PUF generates very similar PUF responses during each evaluation, we expect the Intra-Jaccard indices between PUF responses of a highly-repeatable PUF to be tightly distributed near a value of 1. Figure 3 plots the distribution of Intra-Jaccard indices across every PUF memory segment we tested in red. We observe that while the distribution is multimodal, the Intra-Jaccard indices are clustered very close to 1.0 and *never* drop below 0.65.

Similarly to the Inter-Jaccard index distributions (discussed in Section 6.1.2), we find that the different modes of the Intra-Jaccard index distribution shown in Figure 3 arise from combining the Intra-Jaccard index distributions from all three manufacturers. We plot the Intra-Jaccard index distributions for each manufacturer alone in Figure 4 as indicated by (A),(B), and (C). We observe from the higher distribution mean of Intra-Jaccard indices in Figure 4 for manufacturer B that DRAM latency PUFs evaluated on chips from manufacturer B exhibit higher repeatability than those from manufacturers A or C. We conclude from the high Intra-Jaccard indices in Figures 3 and 4, that DRAM latency PUFs exhibit high repeatability.

Long-term Repeatability. We next study the repeatability of DRAM latency PUFs on a subset of chips over a 30-day period to show that the repeatability property holds for longer periods of time (i.e., a memory segment generates a PUF response similar to its previously-enrolled golden key irrespective of the time since its enrollment). We examine a total of more than a million 8KiB memory segments *across* many chips from each of the three manufacturers as shown in Table 2. The right column indicates the number of memory segments across *n* devices, where *n* is indicated in the left column, and the rows indicate the different manufacturers of the chips containing the memory segments.

	#Chips	#Total Memory Segments
A	19	589,824
B	12	442,879
C	14	437,990

Table 2: Number of PUF memory segments tested for 30 days.

In order to demonstrate the repeatability of evaluating a DRAM latency PUF over long periods of time, we continuously evaluate our DRAM latency PUF across a 30-day period using each of our chosen memory segments. For each memory segment, we calculate the Intra-Jaccard index between the first PUF response and each subsequent PUF response. We find the *Intra-Jaccard index range*, or the range of values (*max_value* − *min_value*) found across the Jaccard indices calculated for every pair of PUF responses from a memory segment. If a memory segment exhibits a low Intra-Jaccard index range, the memory segment generates highly-similar PUF responses during each evaluation over our testing period. Thus, memory segments that exhibit low Intra-Jaccard index ranges demonstrate high repeatability.

Figure 5 shows the distribution of *Intra-Jaccard index ranges* across our memory segments with box-and-whisker plots[2] for each of the three manufacturers. We observe that the Intra-Jaccard index ranges are quite low, i.e., less than 0.1 on average for all manufacturers. Thus, we conclude that the vast majority of memory segments across all manufacturers exhibit very high repeatability over long periods of time.

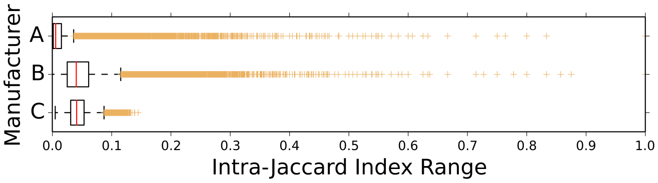


Figure 5: Distribution of the Intra-Jaccard index range values calculated between many PUF responses that a PUF memory segment generates over a 30-day period.

In order to show that every chip has a significant proportion of memory segments that exhibit high reliability over time, we analyze per-chip Intra-Jaccard index range properties. Table 3 shows the *median [minimum, maximum]* of the fraction of memory segments per chip that are observed to have Intra-Jaccard index ranges below 0.1 and 0.2. Over 90% of all segments *in each chip* are suitable for PUF evaluation for Intra-Jaccard index ranges below 0.1, and over 97% for Intra-Jaccard index ranges below 0.2. This means that each chip has a significant number of memory segments that are viable for DRAM latency PUF evaluation. Furthermore, the distributions are very narrow, which indicates that different chips show similar behavior. We conclude that every chip has a significant number of PUF memory segments that exhibit high repeatability across time. We show in Section 7.5 how we can use a simple characterization step to identify these viable memory segments quickly and reliably.

	%Memory Segments per Chip	
	Intra-Jaccard index range <0.1	Intra-Jaccard index range <0.2
A	100.00 [99.08, 100.00]	100.00 [100.00, 100.00]
B	90.39 [82.13, 99.96]	96.34 [95.37, 100.00]
C	95.74 [89.20, 100.00]	96.65 [95.48, 100.00]

Table 3: Percentage of PUF memory segments per chip with Intra-Jaccard index ranges <0.1 or 0.2 over a 30-day period. Median [minimum, maximum] values are shown.

Temperature Effects. To demonstrate how changes in temperature affect PUF evaluation, we evaluate the DRAM latency PUF 10 times for each of the memory segments in Table 2 at each 5°C increment throughout our testable temperature range (55°C-70°C). Figure 6 shows the distributions of Intra-Jaccard indices calculated between every possible pair of PUF responses generated by the *same* memory segment. The deltas between the operating temperatures at the time of PUF evaluation are denoted in the x-axis (*temperature delta*). Since we test at four evenly-spaced temperatures, we have four distinct temperature deltas. The y-axis marks the Jaccard indices calculated between the PUF responses. The distribution of Intra-Jaccard indices found for a given temperature delta is shown using a box-and-whisker plot.

Figure 6 subdivides the distributions for each of the three manufacturers as indicated by A, B, and C. Two observations are in order. 1) Across all three manufacturers, the distribution of Intra-Jaccard indices strictly shifts towards zero as the temperature delta increases. 2) The Intra-Jaccard distribution of PUF responses from chips of manufacturer C are the most sensitive to changes in temperature as reflected in the large distribution shift in Figure 6(C). Both observations show that evaluating a PUF at a temperature different from the temperature during enrollment affects the quality of the PUF response and reduces repeatability. However, 1) for small temperature

[2] The box is bounded by the first quartile (i.e., the median of the first half of the ordered set of Intra-Jaccard index ranges) and third quartile (i.e., the median of the second half of the ordered set of Intra-Jaccard index ranges). The median is marked by a red line within the bounding box. The *inter-quartile range* (IQR) is defined as the difference between the third and first quartiles. The whiskers are drawn out to extend an additional 1.5 × *IQR* above the third quartile and 1.5 × *IQR* below the first quartile. Outliers are shown as orange crosses indicating data points outside of the range of whiskers.

deltas (e.g., 5°), PUF repeatability is not significantly affected, and 2) we discuss in Section 7.5 how we can ameliorate this effect during device enrollment.

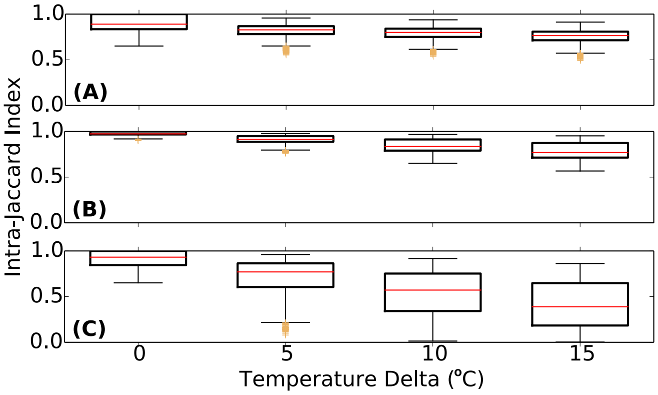


Figure 6: DRAM latency PUF repeatability vs. temperature.

6.2. Runtime-Accessible PUF Metrics Evaluation

Throughout the remainder of this section, we show 1) how the DRAM latency PUF satisfies the characteristics of a *runtime-accessible* PUF (i.e., low latency and low system interference) discussed in Section 3.2, and 2) that the DRAM latency PUF significantly outperforms the DRAM retention PUF in terms of both evaluation time and system interference.

6.2.1. Low Latency. The DRAM latency PUF consists of two key phases: 1) inducing latency failures, and 2) filtering the PUF segment, which improves PUF repeatability (to be discussed in Section 7.1). During Phase 1, we induce latency failures multiple times (i.e., for multiple *iterations*) over the PUF memory segment and count the failures in a separate buffer for additional bookkeeping (we discuss this in further detail in Section 7.2). The execution time of this phase depends directly on three factors:

1. The value of the t_{RCD} timing parameter. A smaller t_{RCD} value causes each read to have a shorter latency.
2. The size of the PUF memory segment. A larger memory segment requires more DRAM read requests per iteration. In our devices, we observe that latency failures are induced at a granularity of 32 bytes with each read request, so we can find the total number of required DRAM reads by dividing the size of the memory segment by 32 bytes.
3. The number of iterations used to induce latency failures. More iterations lead to a longer evaluation time.

Increasing any one of these factors independently of the others directly results in an increase in PUF evaluation time. We experimentally find that a single low-t_{RCD} access to DRAM, along with its associated bookkeeping and memory barrier, takes 3.4μs. Because the value of t_{RCD} is on the scale of tens of nanoseconds [50], changing its value negligibly affects the time for each low-t_{RCD} access. Thus, we use a constant 3.4μs for each read regardless of the t_{RCD} value to find a good estimate of the PUF evaluation time in Equation 1. We experimentally show that Phase 2 has negligible runtime (< 0.1% of total DRAM latency PUF evaluation time) compared with Phase 1, so we omit Phase 2 in our PUF evaluation time estimation.We express PUF evaluation time estimation as:

$$T_{PUF\_eval} = (N_{iters}) \times [(size_{mem\_seg})/(32 \text{ bytes})] \times 3.4\mu s \quad (1)$$

where N_{iters} is the number of times we induce latency failures on each 32 byte block of the memory segment, and $size_{mem\_seg}$ is the size of the memory segment used to evaluate the PUF. For our final chosen configuration (discussed in detail in Section 7), we use the parameters $size_{mem\_seg}$ = 8KiB (Section 7.3), t_{RCD} = 9.8ns (Section 7.4), and N_{iters} = 100 (Section 7.1). Using Equation 1, we expect this configuration to result in an evaluation time of approximately 87ms.

In order to experimentally verify Equation 1, we measure the evaluation time of the DRAM latency PUF for 10000 evaluations across chips from all three manufacturers at 55°C. We find that evaluation times are normally distributed per-manufacturer according to $\mathcal{N}_A(\mu = 89.1\text{ms}, \sigma = 0.0132\text{ms})$, $\mathcal{N}_B(\mu = 88.2\text{ms}, \sigma = 0.0135\text{ms})$, and $\mathcal{N}_C(\mu = 87.2\text{ms}, \sigma = 0.0102\text{ms})$. These distribution parameters show that evaluation times have very similar means and are extremely tightly distributed (i.e., < 0.0002 relative standard deviation). This is expected because, for any particular configuration, DRAM latency PUF evaluation essentially requires a *constant* number of DRAM accesses. Therefore, any variation in PUF evaluation time comes from variations in code execution (e.g., multitasking, interrupts, DRAM refresh, etc.) rather than any characteristics of the PUF itself. In order to compare these runtime distributions with the result of Equation 1, we take take the mean of the mixture distribution of the three per-manufacturer distributions (i.e., $\mathcal{N}_{ABC}(\mu = 88.2\text{ms}, \sigma = 0.716\text{ms})$) and find that the 87ms estimate from Equation 1 results in only 1.4% error.

Figure 2 provides a comparison of DRAM latency PUF evaluation time with retention PUF evaluation time across our testable temperature range (i.e., 55°C-70°C). We find that the DRAM latency PUF significantly outperforms the DRAM retention PUF for an equivalent DRAM capacity overhead of 64KiB (i.e., 8KiB latency PUF memory segment + 56KiB counter buffer), providing an average (minimum, maximum) speedup of 152x (109x, 181x) at 70°C and 1426x (868x, 1783x) at 55°C. By increasing the memory segment size from 64KiB to 64MiB, we can evaluate a DRAM retention PUF in 1.05s (1.01s, 1.09s) at 70°C (Section 5.3). However, the DRAM latency PUF still outperforms this configuration *without* an increase in DRAM capacity overhead (i.e., still with an 8KiB memory segment), providing a speedup of 12.1x (11.6x, 12.5x).

Similarly to prior work on DRAM latency reduction [13, 67], we experimentally find that inducing latency failures is minimally affected by changes in temperature. Importantly, since our method of inducing latency failures does *not* change with temperature (Section 7.2), DRAM latency PUF evaluation time remains reliably short across *all* operating temperatures. We conclude that the DRAM latency PUF 1) can be evaluated at speeds that are orders of magnitude faster than the DRAM retention PUF, and 2) overcomes the temperature dependence of the DRAM retention PUF and maintains a low evaluation time across all temperatures.

6.2.2. Low System Interference. The DRAM latency PUF exhibits two major sources of system interference: 1) requiring exclusive DRAM rank/bank access throughout PUF evaluation, and 2) using a region in a separate DRAM rank to count latency failures (Section 7.2).

First, because DRAM timing parameters can only be manipulated for the coarse granularity of a DRAM rank, any other access to the same rank containing the PUF memory segment must be blocked during PUF evaluation. Such blocking prevents other accesses from obeying the same reduced timing parameters and corrupting the data. For this reason, DRAM latency PUF evaluation requires exclusive access to a full DRAM rank for the entire duration of PUF evaluation. Fortunately, the DRAM latency PUF's quick evaluation time (i.e., 88.2ms on average) guarantees that the DRAM rank will be unavailable only for a short period of time. This is in stark contrast with the DRAM retention PUF, which 1) blocks rank/bank access for much longer periods of time (e.g., on the order of *minutes* or seconds), and 2) requires the memory controller to issue a large number of refresh operations to rows in the rank/bank outside of the PUF memory segment for the same period of time [135].

Second, the DRAM latency PUF algorithm (described in detail in Section 7.2) requires a small *counter buffer* (e.g., a 56KiB buffer for an 8KiB PUF memory segment) which stores counters for each bit of the PUF memory segment. This comes at the cost of both DRAM capacity overhead and additional memory traffic penalty. However, given that the DRAM capacity overhead is small (e.g., <0.003% for a 2GB DRAM using an 8KiB memory segment) and the additional bandwidth consumed is extremely low (e.g., on the order of 100MB/s using an 8KiB memory segment) in the context of total DRAM bandwidth (e.g., 8GB/s), we conclude that the additional system interference induced by the counter buffer is insignificant. In practice, we expect system caches to (fully) hold the counter buffer, further reducing the required DRAM bandwidth.

7. Design Considerations

As we experimentally showed in Section 6, utilizing DRAM latency failures is a viable method for evaluating runtime-accessible PUFs in commodity, unmodified DRAM chips. However, due to variation across DRAM cells and chips, there are various important design considerations that must be made in the implementation of the DRAM latency PUF. In this section, we discuss these considerations for implementing the DRAM latency PUF.

7.1. Repeatability of Cell Latency Failures

Due to many underlying factors (e.g., process variation, temperature), each DRAM cell fails with a different probability when read with a timing parameter reduced beyond the manufacturer specification [12, 13, 67, 68, 69]. We define a *latency-weak cell* as a cell that has a significant probability of failure when read with a reduced timing parameter. Our DRAM latency PUFs are comprised of the locations of latency-weak cells because such cells can be repeatably found. In order to repeatably find the set of latency-weak cells, we employ *many* iterations (e.g., on the order of 100) of inducing latency failures at the PUF memory segment. This improves the chances of a PUF evaluation to find a significant proportion of the latency-weak cells. Because we assume that any given cell has a static probability (p) to fail when accessed with reduced latency, we can model the number of times that a cell must be accessed before observing a latency failure as a *geometric random variable* with a success probability of p. The geometric distribution with parameter p has a mean value of $\frac{1}{p}$. By sampling cells over x iterations during DRAM latency PUF evaluation, we expect to find all cells that fail with a probability greater than or equal to $\frac{1}{x}$. There is a chance that a cell with a failure probability below the threshold fails during an instance of the PUF evaluation, reducing the similarity of the PUF responses across evaluations and thus the repeatability of the PUF. To mitigate this issue, we apply a *filter* (see Section 7.2) that removes cells that we observe to fail in only a small proportion of the x iterations. We empirically find that removing cells that fail in less than 10% of the iterations results in the highest Intra-Jaccard indices across PUF responses.

In order to determine how many iterations to induce latency failures for during latency PUF evaluation, we generate PUF responses across our devices using a varying number of iterations between 1 and 1024. For each set of PUF responses generated with a given number of iterations, we calculate the box-and-whisker plots for both Inter- and Intra-Jaccard distributions (not shown). We find that for PUF responses from chips across all manufacturers, the Inter-Jaccard *and* Intra-Jaccard distributions have strictly the same or increasing medians, first and third quartiles, and whiskers, for an increasing number of iterations.

Higher Intra-Jaccard index distribution values represent a more repeatable PUF since the distribution directly reflects the similarities of PUF responses from the same memory segment. We find that the Intra-Jaccard index distribution's median and bottom whiskers increase by 0.0025 and 0.0054, respectively, for every doubling of the number of iterations. On the other hand, higher Inter-Jaccard index distribution values represent higher similarity across distinct memory segments. Such higher values would limit the PUF's ability to identify many unique devices. We find that the Inter-Jaccard index distribution's median and top whiskers increase by 0.0012 and 0.0011, respectively, for every doubling of the number of iterations. Based on our experimental analyses of these tradeoffs, we choose to induce latency failures for 100 iterations during each DRAM latency PUF evaluation. We next discuss in detail our algorithm for evaluating DRAM latency PUFs with high repeatability.

7.2. DRAM Latency PUF Evaluation Algorithm

We provide an implementable algorithm for evaluating a repeatable DRAM latency PUF at a given memory segment. While we focus on evaluating DRAM latency PUFs with t_{RCD}-induced failures, Algorithm 2 works with any other timing parameter capable of inducing failures. We first initialize the PUF memory segment indicated by $Segment[seg\_id]$ by setting every bit in the memory segment to "1" (line 2). We then attempt to find the reliable set of failures as fast as possible in the memory segment (lines 4-11). Because DRAM RD commands require a t_{RCD} delay only after the activation of a previously closed DRAM row, t_{RCD} failures can only be observed when issuing a read request to a *closed* DRAM row. The key idea is to iterate over each row sequentially such that each read request goes to a different row (i.e., perform column order accesses through the memory segment of interest) as shown in lines 7-9. Before inducing failures across the PUF memory segment, we must first obtain exclusive access to the rank containing the PUF memory segment (line 4), due to the rank-level granularity of changing DRAM timing parameters (Section 2.3). We then reduce the value of t_{RCD} for the entire rank containing the PUF memory segment (line 5). During the iterations of inducing *tRCD* failures (lines 6-11), we issue a memory barrier (line 10) after each read. This ensures that 1) *only one* memory instruction is in flight at a given time and, thus, improves repeatability by simplifying the logic required by the memory controller when issuing memory accesses, and 2) read requests do *not* get reordered by the memory controller to exploit row buffer locality [60, 61, 92, 93, 105, 106, 118, 126]. Instead, each access activates a new row, while obeying the t_{RCD} timing parameter. We find that the instruction order indicated by lines 6-11 is the fastest method for finding a reliable set of latency failures in a memory segment. For every read, the t_{RCD} failure locations are determined and their failures are counted in a separate rank for bookkeeping (line 11). After all iterations of inducing t_{RCD} failures, we must reset the t_{RCD} value to the default (line 12), filter the PUF segment (line 13; see *Filtering Mechanism*), release exclusive access to the rank containing the PUF memory segment (line 14), and finally return the PUF response, i.e., the resulting *error pattern* from the PUF evaluation at the PUF memory segment (line 15).

Filtering Mechanism. In order to improve the repeatability of the DRAM latency PUF, we employ a *filtering mechanism* which removes the cells with low failure probability from the PUF response (as shown on line 13 in Algorithm 2). The key idea is to count, for each bit location in the PUF memory segment, the number of iterations in which the location fails and then use that count to determine whether the bit

Algorithm 2: Evaluate DRAM latency PUF

```
evaluate_DRAM_latency_PUF(seg_id):
  write known data (all 1's) to Segment[seg_id]
  rank_id ← DRAM rank containing seg_id
  obtain exclusive access to Rank[rank_id]
  set low $t_{RCD}$ for Rank[rank_id]
  for i = 1 to num_iterations :
    for col in Segment[seg_id]
      for row in Segment[seg_id]:           // column-order reads
        read()                              // induce read failures
        memory_barrier()                    // one access at a time
        count_failures()                    // record in another rank
  set default $t_{RCD}$ for Rank[rank_id]
  filter the PUF memory segment             // See Filtering Mechanism
  release exclusive access to Rank[rank_id]
  return error pattern at Segment[seg_id]
```

location should be *set* ("1") or *cleared* ("0") in the final PUF response. Every bit in the DRAM PUF memory segment has a corresponding counter that we store in the *counter buffer*, a data structure we allocate in a DRAM rank separate from the one containing the PUF memory segment. This is to ensure that read requests to the counter buffer follow manufacturer-specified timing parameters and do not induce latency failures.

After each reduced-latency read request in the PUF memory segment, we find all bit locations in the read data that resulted in a latency failure, and increment their corresponding counters in the counter buffer. After all iterations of inducing latency failures are completed, we compare every counter of each bit location in the PUF memory segment against a threshold. If a counter holds a value greater than the threshold (i.e., the counter's corresponding bit location failed more than n times, where n is the threshold), we set the corresponding bit location. Otherwise, we clear it.

Memory Footprint. Equation 2 provides the memory footprint required by PUF evaluation:

$$mem_{total} = (size_{mem\_seg}) + (size_{counter\_buffer}) \qquad (2)$$

where $size_{mem\_seg}$ is the size of the PUF memory segment and $size_{counter\_buffer}$ is the size of the counter buffer. The size of the counter buffer can be calculated using Equation 3:

$$size_{counter\_buffer} = (size_{mem\_seg}) \times \lceil \log_2 N_{iters} \rceil \qquad (3)$$

where $size_{mem\_seg}$ is the size of the PUF memory segment and N_{iters} is the number of iterations that we want to induce latency failures for. Since we require one counter per bit in the memory segment, we must multiply this quantity by the size of each counter. Since the counter must be able to store up to the value of N_{iters} (e.g., in the case of a cell that fails every iteration), each counter must be $\lceil \log_2 N_{iters} \rceil$ bits wide. For a memory segment size of 8KiB, we find that the DRAM latency PUF's total memory footprint is 64KiB. From this, we conclude that DRAM latency PUFs have insignificant DRAM capacity overhead.

7.3. Variation Among PUF Memory Segments

We observe a variation in latency failure rates across different memory segments, which make some DRAM memory segments more desirable to evaluate DRAM latency PUFs with than others. Because we want to find 512 bits that fail per PUF memory segment (Section 5.2), we consider only those memory segments that have at least 512 failing bits as *good* memory segments. In order to determine the best size of the memory segment to evaluate the DRAM latency PUF on, we study the effect of varying memory segment size on 1) DRAM capacity overhead, 2) PUF evaluation time, and 3) fraction of good memory segments per device. As the memory segment size increases, both the DRAM capacity overhead and the PUF evaluation time increase linearly. The number of possible PUF memory segments for a DRAM device with a DRAM latency PUF is obtained by counting the number of contiguous PUF memory segments across all of DRAM (i.e., dividing the DRAM size by the PUF memory segment size). Thus, larger PUF memory segments result in fewer possible PUF memory segments for a DRAM device. From an experimental analysis of the associated tradeoffs of varying the PUF memory segment size (not shown), we choose a PUF memory segment size of 8KiB.[3]

In Table 4, we represent the distribution of the percentage of good memory segments per chip with a *median [minimum, maximum]* across each of the three manufacturers. The left column shows the number of chips tested, the right column shows the representation of the distribution, and the rows indicate the different manufacturers of the chips. We see that an overwhelming majority of memory segments from manufacturers A and B are good for PUF evaluation. Memory segments from chips of manufacturer C were observed to exhibit less latency failures, but across each of our chips we could find at least 19.4% of the memory segments to be good for PUF evaluation. Of the total number of PUF memory segments tested (shown in Table 2), we experimentally find that 100%, 64.06%, and 19.37% of memory segments are *good* (i.e., contain enough failures to be considered for PUF evaluation) in the worst-case chips from manufacturers A, B, and C. We conclude that there are plenty of PUF memory segments that are good enough for DRAM latency PUF evaluation.

	#Chips	Good Memory Segments per Chip (%)
A	19	100.00 [100.00, 100.00]
B	12	100.00 [64.06, 100.00]
C	14	30.86 [19.37, 95.31]

Table 4: Percentage of *good* memory segments per chip across manufacturers. Median [min, max] values are shown.

7.4. Support for Changing Timing Parameters

In order to induce latency failures, the manufacturer-specified DRAM timing parameters must be changed. Some existing processors [1, 3, 67] enable software to directly manipulate DRAM timing parameters. These processors can trivially implement and evaluate a DRAM latency PUF with *minimal* changes to the software and no changes to hardware. However, for other processors that cannot directly manipulate DRAM timing parameters, we would need to simply enable software to programmatically modify memory controller registers which indicate the DRAM timing parameters that a memory access must observe.

We find that we can reliably induce latency failures when we reduce the value of t_{RCD} from a default value of 18ns to between 6ns and 13ns. Given this wide range of failure-inducing t_{RCD} values, most memory controllers should be able to issue read requests with a t_{RCD} value within this range.

7.5. Device Enrollment

Device enrollment is a one-time process consisting of evaluating all possible PUFs from across the entire challenge-response space and securely storing the evaluated PUFs in a trusted database such that they can be later queried for authentication [52, 120, 135]. Since the goal of PUF authentication is to ensure that a challenge-response is difficult to replicate without access to the original device, enrollment must be done securely so that the full set of all possible challenge-response

[3] We will provide details in a technical report/extended version for all other results that we cannot provide detail for in the submission.

pairs is known only to the trusted database and can be created *only* by the device owner.

Similar to prior works' approach to DRAM PUFs [120, 123, 135], we assume that a trusted third party (e.g., the DRAM manufacturer) performs device enrollment prior to making the system available to the end consumer. This ensures that the complete set of all possible challenge responses are known only by the trusted third party. After the device is in the field, even the trusted third party *cannot* regenerate the enrollment data. Thus, allowing the trusted third party to both characterize and enroll the responses makes it extremely difficult for a malicious attacker to obtain the full set of possible response pairs without first compromising the trusted third party.

Because DRAM latency PUF responses vary depending on the temperature of DRAM during evaluation time (see Section 6.1.4), we must enroll multiple golden keys at varying temperature intervals. This enables a PUF response to match at least one golden key during authentication regardless of the temperature during evaluation time. We find that some chips generate PUF responses with less variation across a range of temperatures than other chips. Chips with less variation can enroll golden keys for temperatures at larger intervals than chips with more variation.

7.6. In-DRAM Error Correcting Codes

Some new DRAM chips utilize in-DRAM *error-correcting codes* (ECC), which perform single-bit error correction per *word* (i.e., typically 64 data bits) invisibly to the system [51, 96, 97], to overcome the reliability challenges of DRAM technology scaling [59, 79, 89, 90, 91, 94, 110]. When such chips are used for DRAM latency PUFs, ECC words with only one error appear to be error-free to the memory controller, leaving fewer total errors available for PUF response. We note that error correction *deterministically* transforms DRAM error patterns. Thus, a DRAM PUF (on a chip with in-DRAM ECC) that repeatably induces the same error pattern prior to ECC correction, would repeatably result in a different but *consistent error pattern* after ECC correction.

In order to support PUF evaluation in a system using DRAM chips with in-DRAM ECC, we would need to evaluate PUFs with a higher *raw bit error rate* (i.e., the error rate before ECC is performed) relative to non-ECC DRAMs. A higher raw bit error rate would produce enough observable failures (after ECC is performed) for a PUF. The DRAM latency PUF can achieve this higher raw bit error rate by simply reducing the latency parameter value further. Such reduction would also ideally reduce PUF evaluation time and system interference. Therefore, we expect the DRAM latency PUF to be evaluated even faster in chips with built-in ECC.[4] As stronger ECC mechanisms are used, i.e., ECC can correct DRAM words containing more than 1 error, we expect even lower evaluation latencies and lower system interference with the DRAM latency PUF.

7.7. Effect of High-Temperature

Our evaluation of the DRAM latency and retention PUFs is limited to the 70°C maximum DRAM temperature. We clearly show in Section 5.2 that DRAM latency PUFs are much faster than DRAM retention PUFs at 70°C and lower (Figure 2). However, at higher temperatures (e.g., $>$ 85°C), DRAM retention PUFs could become faster than DRAM latency PUFs.[5]

If a DRAM retention PUF is faster than the DRAM latency PUF at a very high temperature, it is easy to envision a mechanism that dynamically switches between DRAM latency PUFs and DRAM retention PUFs based on the device operating temperature at the time of evaluation. This mechanism could exploit the strengths of each type of DRAM PUF in order to allow the fastest possible PUF evaluation time. By exploiting in-DRAM temperature sensors that already exist in modern DRAM chips [47, 48, 50, 67], this mechanism could potentially be implemented with no additional hardware overhead beyond what is already required for DRAM retention PUFs and DRAM latency PUFs individually.

Such a mechanism would require challenge-response pairs from both the DRAM retention PUF and DRAM latency PUF to be enrolled. Furthermore, since retention failure rates vary significantly across different chips [54, 78, 101], each chip will have a different temperature at which the mechanism switches between the two DRAM PUFs. This could considerably impact enrollment time and complicate the device authentication process. Ultimately, it is up to the system architect to decide whether such a mechanism is worth the evaluation runtime benefits at very high temperatures (which is likely to be on the order of tens of milliseconds). We leave a full exploration of this hybrid DRAM latency-retention PUF mechanism to future work.

8. Related Work

To our knowledge, this is the first work to: 1) introduce the idea of violating DRAM read latency parameters to create a fast, runtime-accessible DRAM PUF without modifying commodity DRAM devices, 2) introduce an effective DRAM PUF that is runtime-accessible at all operating temperatures, 3) demonstrate a wide variety of tradeoffs in DRAM PUFs, based on extensive new experimental data from 223 state-of-the-art LPDDR4 DRAM chips, 4) demonstrate the prohibitively slow evaluation times of the DRAM retention PUF, the previously fastest DRAM PUF suitable for commodity devices.

In this section, we discuss prior works that propose DRAM PUFs and PUFs based on other substrates. The proliferation of recent works on DRAM PUFs reflects the growing importance of DRAM PUFs given DRAM's near ubiquity in modern systems and large address space.

DRAM Retention PUFs. We have already described the basics of DRAM retention PUFs in Section 2.2 and extensively evaluated them in Section 5. We briefly explain the differences between prior proposals. Keller et al. [53] is the first to propose using DRAM retention failures as unique identifiers, shortly followed by Xiong et al. [135] and D-PUF [120], both of which enable runtime-accessible DRAM retention PUFs. Other works propose further optimizations for improving the quality of DRAM retention PUFs [103, 122, 124]. As our experimental evaluations across 223 LPDDR4 DRAM chips show, DRAM retention PUFs take very long to evaluate at common-case operating temperatures; they are orders of magnitude slower than our proposal (see Section 5).

Other DRAM PUFs. Hashemian et al. [37] propose adding a delay generator to the DRAM write-circuitry to induce failures. However, this requires additional hardware and cannot be applied to existing DRAM designs. Tehranipoor et al. [123] suggest using DRAM start-up values for PUFs, but this precludes runtime evaluation by requiring a DRAM power cycle for every authentication.

PUFs Based on Other Substrates. Many PUFs have been proposed for various other substrates, including other memory technologies and customized hardware designs. We categorize them into delay-based and memory-based PUFs.

Delay-based PUFs include 1) arbiter PUFs [36, 72, 75, 88, 98, 99, 108, 138], which rely on process variation to extract the

[4] In contrast, DRAM retention PUF *must* be evaluated with a longer refresh interval to increase the raw bit error rate in a DRAM chip with in-DRAM ECC. This leads to a significantly longer DRAM retention PUF evaluation time when in-DRAM ECC is used.

[5] Note that the DDR protocol specifies that every cell must be refreshed at least every 32ms for LPDDR3/4 or 64ms for DDR3/4 below 85°C and at even higher rates at higher temperatures.

unique behavior of two identical competing circuit paths, 2) ring oscillator PUFs [28, 29, 77, 119], which rely on frequencies of oscillating signals from chained inverters, and 3) Current Mirror Array (CMA) PUFs [133], which rely on the manufacturing process variation in a customized circuit typically used for machine learning tasks. These works rely on customized hardware *not* present in commodity systems. FPGA-based PUFs [31, 32, 33, 65, 86, 87] overcome the need for hardware changes. However, they are not as commonly found as DRAM in computer systems.

Memory-based PUFs include SRAM PUFs, which rely on SRAM start up values [9, 20, 32, 41, 42, 121, 134, 145] and voltage reduction induced failures [6]; butterfly PUFs, which mimic the behavior of SRAM cells with cross-coupled data latches [65]; latch PUFs, which cross-couple two NOR-gates [116]; flip-flop PUFs, which exploit the power up behavior of regular flip-flops [84, 127]; and PUFs for emerging memory technologies [8, 17, 18, 23, 45, 64, 80, 100, 107, 128, 129, 140, 141]. These prior works either require additional customized hardware or usage of SRAM, which has a small address space compared to DRAM, and thus cannot accommodate a large number of challenge-response pairs.

9. Conclusion

We introduce and analyze the DRAM latency PUF, a new DRAM PUF suitable for runtime authentication. The DRAM latency PUF intentionally violates manufacturer-specified DRAM timing parameters in order to provide many highly repeatable, unique, and unclonable PUF responses with low latency. Through experimental evaluation using 223 state-of-the-art LPDDR4 DRAM devices, we show that the DRAM latency PUF reliably generates PUF responses at runtime-accessible speeds (i.e., 88.2ms on average) at all operating temperatures. We show that the DRAM latency PUF achieves an average speedup of 152x/1426x at 70°C/55°C when compared with a DRAM retention PUF of the same DRAM capacity overhead, and it achieves even greater speedups at lower temperatures. We conclude that the DRAM latency PUF enables a fast and effective substrate for runtime device authentication across all operating temperatures, and we hope that the advent of runtime-accessible PUFs like the DRAM latency PUF and the detailed experimental characterization data we provide on modern DRAM devices will enable security architects to develop even more secure systems for future devices.

Acknowledgments

We thank Ivan Puddu for useful comments and anonymous reviewers and SAFARI group members for feedback.

References

[1] "AMD Opteron 4300 Series Processors," http://www.amd.com/en-us/products/server/4000/4300.

[2] "RDMA Protocol Specification," http://www.rdmaconsortium.org/.

[3] AMD, "BKDG for AMD Family 16h Models 00h-0Fh Processors," 2013.

[4] R. Ausavarungnirun *et al.*, "Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems," in *ISCA*, 2012.

[5] A. Aysu *et al.*, "A New Maskless Debiasing Method for Lightweight Physical Unclonable Functions," in *HOST*, 2017.

[6] A. Bacha *et al.*, "Authenticache: Harnessing Cache ECC for System Authentication," in *MICRO*, 2015.

[7] S. Baek *et al.*, "Refresh Now and Then," in *TC*, 2014.

[8] K. Beckmann *et al.*, "Performance Enhancement of a Time-Delay PUF Design by Utilizing Integrated Nanoscale ReRAM Devices," in *TETC*, 2017.

[9] M. Bhargava *et al.*, "Reliability Enhancement of Bi-Stable PUFs in 65nm Bulk CMOS," in *HOST*, 2012.

[10] I. Bhati *et al.*, "DRAM Refresh Mechanisms, Penalties, and Trade-offs," in *TC*, 2016.

[11] K. Chandrasekar *et al.*, "Exploiting Expendable Process-Margins in DRAMs for Run-Time Performance Optimization," in *DATE*, 2014.

[12] K. K. Chang, "Understanding and Improving Latency of DRAM-Based Memory Systems," Ph.D. dissertation, Carnegie Mellon University, 2017.

[13] K. K. Chang *et al.*, "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization," in *SIGMETRICS*, 2016.

[14] K. K. Chang *et al.*, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.

[15] K. K. Chang *et al.*, "Low-cost Inter-linked Subarrays (LISA): Enabling Fast Inter-subarray Data Movement in DRAM," in *HPCA*, 2016.

[16] K. K. Chang *et al.*, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *SIGMETRICS*, 2017.

[17] W. Che *et al.*, "A Non-Volatile Memory Based Physically Unclonable Function without Helper Data," in *ICCAD*, 2014.

[18] P.-Y. Chen *et al.*, "Exploiting Resistive Cross-Point Array for Compact Design of Physical Unclonable Function," in *HOST*, 2015.

[19] J. Choi *et al.*, "Modeling and Managing Thermal Profiles of Rack-mounted Servers with Thermostat," in *HPCA*, 2007.

[20] M. Cortez *et al.*, "Adapting Voltage Ramp-up Time for Temperature Noise Reduction on Memory-based PUFs," in *HOST*, 2013.

[21] Z. Cui *et al.*, "DTail: A Flexible Approach to DRAM Refresh Management," in *SC*, 2014.

[22] J. L. Danger *et al.*, "PUFs: Standardization and Evaluation," in *MST*, 2016.

[23] J. Das *et al.*, "MRAM PUF: A Novel Geometry Based Magnetic PUF with Integrated CMOS," in *TNANO*, 2015.

[24] H. David *et al.*, "Memory Power Management via Dynamic Voltage/Frequency Scaling," in *ICAC*, 2011.

[25] S. Desai, "Process Variation Aware DRAM (Dynamic Random Access Memory) Design Using Block-based Adaptive Body Biasing Algorithm," Ph.D. dissertation, Utah State University, 2012.

[26] N. El-Sayed *et al.*, "Temperature Management in Data Centers: Why Some (Might) Like it Hot," in *SIGMETRICS*, 2012.

[27] Y. Gao *et al.*, "Memristive Crypto Primitive for Building Highly Secure Physical Unclonable Functions," in *Scientific Reports*, 2015.

[28] B. Gassend *et al.*, "Silicon Physical Random Functions," in *CCS*, 2002.

[29] B. L. Gassend, "Physical Random Functions," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[30] S. Govindavajhala and A. W. Appel, "Using Memory Errors to Attack a Virtual Machine," in *SP*, 2003.

[31] C. Gu and M. O'Neill, "Ultra-Compact and Robust FPGA-Based PUF Identification Generator," in *ISCAS*, 2015.

[32] J. Guajardo *et al.*, "FPGA Intrinsic PUFs and Their Use for IP Protection," in *CHES*, 2007.

[33] J. Guajardo *et al.*, "Physical Unclonable Functions and Public-key Crypto for FPGA IP Protection," in *FPL*, 2007.

[34] T. Hamamoto *et al.*, "On the Retention Time Distribution of Dynamic Random Access Memory (DRAM)," in *ED*, 1998.

[35] R. W. Hamming, "Error Detecting and Error Correcting Codes," in *Bell Labs Technical Journal*, 1950.

[36] G. Hammouri *et al.*, "Unclonable Lightweight Authentication Scheme," in *ICICS*, 2008.

[37] M. S. Hashemian *et al.*, "A Robust Authentication Methodology Using Physically Unclonable Functions in DRAM Arrays," in *DATE*, 2015.

[38] H. Hassan *et al.*, "ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality," in *HPCA*, 2016.

[39] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.

[40] R. Helinski *et al.*, "A Physical Unclonable Function Defined Using Power Distribution System Equivalent Resistance Variations," in *DAC*, 2009.

[41] D. E. Holcomb *et al.*, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," in *TC*, 2009.

[42] D. E. Holcomb *et al.*, "Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags," in *RFID*, 2007.

[43] Y. Hori *et al.*, "Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs," in *ReConFig*, 2010.

[44] E. Ipek *et al.*, "Self-Optimizing Memory Controllers: A Reinforcement Learning Approach," in *ISCA*, 2008.

[45] A. Iyengar *et al.*, "DWM-PUF: A Low-Overhead, Memory-Based Security Primitive," in *HOST*, 2014.

[46] P. Jaccard, "Étude Comparative de la Distribution Florale dans une Portion des Alpes et des Jura," in *Bull Soc Vaudoise Sci Nat*, 1901.

[47] JEDEC, "Double Data Rate 4 (DDR4) SDRAM Standard," 2012.

[48] JEDEC, "Low Power Double Data Rate 3 (LPDDR3)," 2012.

[49] JEDEC, "Low Power Double Data Rate 4 (LPDDR4) SDRAM Specification," *JEDEC Solid State Technology Association*, 2014.

[50] JEDEC, "LPDDR4," *JEDEC Standard JESD209–4A*, 2014.

[51] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.

[52] S. Katzenbeisser *et al.*, "PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon," in *CHES*, 2012.

[53] C. Keller *et al.*, "Dynamic Memory-based Physically Unclonable Function for the Generation of Unique Identifiers and True Random Numbers," in *ISCAS*, 2014.

[54] S. Khan *et al.*, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS*, 2014.

[55] S. Khan *et al.*, "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM," in *DSN*, 2016.

[56] S. Khan *et al.*, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," in *CAL*, 2016.

[57] S. Khan *et al.*, "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content," in *MICRO*, 2017.

[58] K. Kim and J. Lee, "A New Investigation of Data Retention Time in Truly Nanoscaled DRAMs," in *EDL*, 2009.

[59] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.

[60] Y. Kim *et al.*, "ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers," in *HPCA*, 2010.

[61] Y. Kim *et al.*, "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior," in *MICRO*, 2010.

[62] Y. Kim *et al.*, "A Case for Exploiting Subarray-level Parallelism (SALP) in DRAM," in *ISCA*, 2012.

[63] Y. Kim *et al.*, "Ramulator: A Fast and Extensible DRAM Simulator," in *CAL*, 2016.

[64] P. Koeberl *et al.*, "Memristor PUFs: A New Generation of Memory-based Physically Unclonable Functions," in *DATE*, 2013.

[65] S. S. Kumar *et al.*, "The Butterfly PUF Protecting IP on Every FPGA," in *HOST*, 2008.

[66] C. J. Lee *et al.*, "DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems," *HPS Technical Report*, 2010.

[67] D. Lee *et al.*, "Adaptive-latency DRAM: Optimizing DRAM Timing for the Common-case," in *HPCA*, 2015.

[68] D. Lee, "Reducing DRAM Latency at Low Cost by Exploiting Heterogeneity," Ph.D. dissertation, Carnegie Mellon University, 2016.

[69] D. Lee *et al.*, "Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms," in *SIGMETRICS*, 2017.

[70] D. Lee *et al.*, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," in *HPCA*, 2013.

[71] D. Lee *et al.*, "Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM," in *PACT*, 2015.

[72] J. W. Lee *et al.*, "A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications," in *Symposium on VLSIC*, 2004.

[73] M. J. Lee and K. W. Park, "A Mechanism for Dependence of Refresh Time on Data Pattern in DRAM," in *EDL*, 2010.

[74] Y. Li *et al.*, "DRAM Yield Analysis and Optimization by a Statistical Design Approach," in *CSI*, 2011.

[75] D. Lim *et al.*, "Extracting Secret Keys from Integrated Circuits," *VLSI*, 2005.

[76] C. H. Lin *et al.*, "SECRET: Selective Error Correction for Refresh Energy Reduction in DRAMs," in *ICCD*, 2012.

[77] C. Q. Liu *et al.*, "ACRO-PUF: A Low-power, Reliable and Aging-Resilient Current Starved Inverter-Based Ring Oscillator Physical Unclonable Function," in *TCS*, 2017.

[78] J. Liu *et al.*, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," in *ISCA*, 2013.

[79] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.

[80] R. Liu *et al.*, "Experimental Characterization of Physical Unclonable Function Based on 1 KB Resistive Random Access Memory Arrays," in *EDL*, 2015.

[81] S. Liu *et al.*, "Hardware/Software Techniques for DRAM Thermal Management," in *HPCA*, 2011.

[82] W. Liu *et al.*, "A Trustworthy Key Generation Prototype Based on DDR3 PUF for Wireless Sensor Networks," in *Sensors*, 2014.

[83] K. Lofstrom *et al.*, "IC Identification Circuit Using Device Mismatch," in *ISSCC*, 2000.

[84] R. Maes *et al.*, "Intrinsic PUFs from Flip-flops on Reconfigurable Devices," in *WISSec*, 2008.

[85] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions," in *Towards Hardware-Intrinsic Security*, 2010.

[86] A. Maiti *et al.*, "The Impact of Aging on an Fpga-Based Physical Unclonable Function," in *FPL*, 2011.

[87] M. Majzoobi *et al.*, "FPGA PUF Using Programmable Delay Lines," in *WIFS*, 2010.

[88] M. Majzoobi *et al.*, "Testing Techniques for Hardware Security," in *ITC*, 2008.

[89] J. Meza *et al.*, "Revisiting Memory Errors in Large-scale Production Data Centers: Analysis and Modeling of New Trends from the Field," in *DSN*, 2015.

[90] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.

[91] O. Mutlu, "The RowHammer Problem and Other Issues we may Face as Memory Becomes Denser," in *DATE*, 2017.

[92] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," in *MICRO*, 2007.

[93] O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enabling High-performance And Fair Shared Memory Controllers," in *ISCA*, 2008.

[94] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," in *SUPERFRI*, 2014.

[95] P. J. Nair *et al.*, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates," in *ISCA*, 2013.

[96] P. J. Nair *et al.*, "XED: Exposing On-Die Error Detection Information for Strong Memory Reliability," in *ISCA*, 2016.

[97] T.-Y. Oh *et al.*, "A 3.2Gbps/pin 8Gb 1.0V LPDDR4 SDRAM with Integrated ECC Engine for sub-1V DRAM Core Operation," 2014.

[98] E. Öztürk *et al.*, "Physical Unclonable Function with Tristate Buffers," in *ISCAS*, 2008.

[99] E. Öztürk *et al.*, "Towards Robust Low Cost Authentication for Pervasive Devices," in *PerCom*, 2008.

[100] Y. Pang *et al.*, "Optimization of RRAM-based Physical Unclonable Function with a Novel Differential Read-Out Method," in *EDL*, 2017.

[101] M. Patel *et al.*, "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," in *ISCA*, 2017.

[102] M. K. Qureshi *et al.*, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems," in *DSN*, 2015.

[103] A. Rahmati *et al.*, "Probable Cause: The Deanonymizing Effects of Approximate DRAM," in *ISCA*, 2016.

[104] P. J. Restle *et al.*, "DRAM Variable Retention Time," in *IEDM*, 1992.

[105] S. Rixner, "Memory Controller Optimizations For Web Servers," in *MICRO*, 2004.

[106] S. Rixner *et al.*, "Memory Access Scheduling," in *ISCA*, 2000.

[107] G. S. Rose *et al.*, "Foundations of Memristor Based PUF Architectures," in *NANOARCH*, 2013.

[108] U. Rührmair *et al.*, "On the Foundations of Physical Unclonable Functions," in *IACR Cryptology Archive*, 2009.

[109] A. Schaller *et al.*, "Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security," in *HOST*, 2017.

[110] B. Schroeder *et al.*, "DRAM Errors in the Wild: a Large-scale Field Study," in *SIGMETRICS*, 2009.

[111] V. Seshadri, "Simple DRAM and Virtual Memory Abstractions to Enable Highly Efficient Memory Systems," Ph.D. dissertation, Carnegie Mellon University, 2016.

[112] V. Seshadri *et al.*, "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization," in *MICRO*, 2013.

[113] V. Seshadri *et al.*, "Ambit: In-memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," in *MICRO*, 2017.

[114] W. Shin *et al.*, "NUAT: A Non-Uniform Access Time Memory Controller," in *HPCA*, 2014.

[115] C. G. Shirley and W. R. Daasch, "Copula Models of Correlation: A DRAM Case Study," in *TC*, 2014.

[116] Y. Su *et al.*, "A 1.6 pJ/bit 96% Stable Chip-ID Generating Circuit Using Process Variations," in *ISSCC*, 2007.

[117] L. Subramanian *et al.*, "The Blacklisting Memory Scheduler: Achieving High Performance And Fairness At Low Cost," in *ICCD*, 2014.

[118] L. Subramanian *et al.*, "BLISS: Balancing Performance, Fairness and Complexity in Memory Access Scheduling," in *TPDS*, 2016.

[119] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *DAC*, 2007.

[120] S. Sutar *et al.*, "D-PUF: An Intrinsically Reconfigurable DRAM PUF for Device Authentication in Embedded Systems," in *CASES*, 2016.

[121] S. Sutar *et al.*, "Memory-based Combination PUFs for Device Authentication in Embedded Systems," in *arXiv*, 2017.

[122] Q. Tang *et al.*, "A DRAM Based Physical Unclonable Function Capable of Generating $>10^{32}$ Challenge Response Pairs per 1Kbit Array for Secure Chip Authentication," in *CICC*, 2017.

[123] F. Tehranipoor *et al.*, "DRAM Based Intrinsic Physical Unclonable Functions for System Level Security," in *GLVLSI*, 2015.

[124] F. Tehranipoor *et al.*, "Investigation of DRAM PUFs Reliability Under Device Accelerated Aging Effects," in *ISCAS*, 2017.

[125] P. Tuyls *et al.*, "Read-proof Hardware From Protective Coatings," in *CHES*, 2006.

[126] H. Usui *et al.*, "DASH: Deadline-Aware High-performance Memory Scheduler for Heterogeneous Systems fith Hardware Accelerators," in *TACO*, 2016.

[127] V. Van der Leest *et al.*, "Hardware Intrinsic Security from D Flip-flops," in *ACM STC*, 2010.

[128] E. I. Vatajelu *et al.*, "STT MRAM-Based PUFs," in *DATE*, 2015.

[129] E. I. Vatajelu *et al.*, "STT-MRAM-Based PUF Architecture Exploiting Magnetic Tunnel Junction Fabrication-Induced Variability," in *JETC*, 2016.

[130] J. Wang *et al.*, "ProactiveDRAM: A DRAM-initiated Retention Management Scheme," in *ICCD*, 2014.

[131] Y. Wang *et al.*, "RADAR: A Case for Retention-aware DRAM Assembly and Repair in Future FGR DRAM Memory," in *DAC*, 2015.

[132] Y. Wang *et al.*, "Flash Memory for Ubiquitous Hardware Security Functions: True Random Number Generation and Device Fingerprints," in *SP*, 2012.

[133] Z. Wang *et al.*, "Current Mirror Array: A Novel Circuit Topology for Combining Physical Unclonable Function and Machine Learning," in *TCS*, 2017.

[134] K. Xiao *et al.*, "Bit Selection Algorithm Suitable for High-volume Production of SRAM-PUF," in *HOST*, 2014.

[135] W. Xiong *et al.*, "Run-time Accessible DRAM PUFs in Commodity Devices," in *CHES*, 2016.

[136] W. Yan *et al.*, "A Novel Way to Authenticate Untrusted Integrated Circuits," in *ICCAD*, 2015.

[137] D. S. Yaney *et al.*, "A Meta-stable Leakage Phenomenon in DRAM Charge Storage-Variable Hold Time," in *IEDM*, 1987.

[138] J. Ye *et al.*, "OPUF: Obfuscation Logic Based Physical Unclonable Function," in *IOLTS*, 2015.

[139] C. E. Yin *et al.*, "Design and Implementation of a Group-Based RO PUF," in *DATE*, 2013.

[140] L. Zhang *et al.*, "Highly Reliable Spin-Transfer Torque Magnetic RAM-Based Physical Unclonable Function with Multi-Response-Bits per Cell," in *TIFS*, 2015.

[141] L. Zhang *et al.*, "Optimizing Emerging Nonvolatile Memories for Dual-Mode Applications: Data Storage and Key Generator," in *TCAD*, 2015.

[142] T. Zhang *et al.*, "Half-DRAM: A High-bandwidth and Low-power DRAM Architecture from the Rethinking of Fine-grained Activation," in *ISCA*, 2014.

[143] X. Zhang *et al.*, "Exploiting DRAM Restore Time Variations In Deep Sub-micron Scaling," in *DATE*, 2015.

[144] X. Zhang *et al.*, "Restore Truncation for Performance Improvement in Future DRAM Systems," in *HPCA*, 2016.

[145] Y. Zheng *et al.*, "RESP: A Robust Physical Unclonable Function Retrofitted into Embedded SRAM Array," in *DAC*, 2013.