Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security

André Schaller*, Wenjie Xiong†, Nikolaos Athanasios Anagnostopoulos*, Muhammad Umair Saleem*,
Sebastian Gabmeyer*, Stefan Katzenbeisser* and Jakub Szefer†
*Technische Universität Darmstadt and CYSEC, Darmstadt, Germany
†Yale University, New Haven, CT, USA

***Abstract*—Physically Unclonable Functions (PUFs) have become an important and promising hardware primitive for device fingerprinting, device identification, or key storage. Intrinsic PUFs leverage components already found in existing devices, unlike extrinsic silicon PUFs, which are based on customized circuits that involve modification of hardware. In this work, we present a new type of a memory-based intrinsic PUF, which leverages the Rowhammer effect in DRAM modules – the *Rowhammer PUF*. Our PUF makes use of bit flips, which occur in DRAM cells due to rapid and repeated access of DRAM rows. Prior research has mainly focused on Rowhammer attacks, where the Rowhammer effect is used to illegitimately alter data stored in memory, e.g., to change page table entries or enable privilege escalation attacks. Meanwhile, this is the first work to use the Rowhammer effect in a positive context – to design a novel PUF. We extensively evaluate the Rowhammer PUF using commercial, off-the-shelf devices, not relying on custom hardware or an FPGA-based setup. The evaluation shows that the Rowhammer PUF holds required properties needed for the envisioned security applications, and could be deployed today.**

I. Introduction

In recent years, attacks that exploit the Rowhammer effect have gained a lot of attention, as they can enable a plethora of security-related risks due to the wide-spread vulnerability imposed by the Rowhammer effect in today's DRAM modules. The phenomenon was first described by Kim et al. [1], who were able to induce so-called disturbance errors in high-density, commodity DRAM modules by repeatedly accessing uncached memory rows. Disturbance errors occur due to the charge coupling between DRAM cells, which accelerates charge leakage in adjacent rows, and eventually results in bits being flipped in so-called victim rows in DRAM, even though said victim rows were not explicitly accessed. The Rowhammer effect allows for breaking many software-based security mechanisms, as well as memory and process isolation, because it allows flipping memory bits, which would otherwise be protected by software-based access control mechanisms. Numerous papers have been published that use the Rowhammer effect in order to improve the identification of vulnerable DRAM cells or to implement various Rowhammer attacks [2], [3], [4], [5].

In contrast to the existing work on the Rowhammer effect, we present a novel approach that uses DRAM disturbance errors, in order to strengthen the security of DRAM-equipped devices, instead of attacking such platforms. We propose to use bit flips, induced by the Rowhammer effect, as basis for a Physically Unclonable Function (PUF) that allows for robust identification of DRAM-equipped devices. We further present a software-only solution that works on commodity hardware and which enables runtime queries to the Rowhammer PUF, not requiring custom hardware or an FPGA setup. Prior work on DRAM PUFs has considered using the decay characteristics of DRAM cells when refresh is disabled, e.g. [6], but the Rowhammer effect as a source of a PUF has not been explored so far. Compared to existing DRAM decay-based PUFs, the Rowhammer PUF takes advantage of disturbance errors to increase the entropy of the PUF response. With our new approach, we enable DRAM-equipped low-cost platforms to use hardware-based fingerprinting, identification, or key storage mechanisms without adding extra logic, e.g., as opposed to extrinsic arbiter PUFs that require new circuits to be added to the computing platform. Since many, if not most, DRAM-equipped platforms are affected by the Rowhammer effect [1], application of Rowhammer PUF goes well beyond just the platforms tested in this work. Additionally, unlike most known intrinsic PUFs, particularly SRAM-based PUFs, which can only be accessed at SRAM boot-up time, the Rowhammer PUF can be queried both at boot-up time and at runtime of a system.

Contributions

This paper extends the field of Physically Unclonable Functions (PUFs) with the following contributions:

- We introduce the Rowhammer PUF, which leverages disturbance errors among DRAM rows that manifest themselves as bit flips, which are used as basis for the new type of Physically Unclonable Function.
- We implement the Rowhammer PUF on commodity, off-the-shelf devices, in a way which is accessible during runtime and which requires no custom hardware or an FPGA setup.
- We provide an extensive evaluation, showing very good metrics for uniqueness, robustness and entropy. We further show the PUF's ability to operate at different ambient temperatures in a stable manner.

II. Background

A. DRAM Data Storage and Access

DRAM stores a "bit" as charge on a capacitor. A single DRAM cell consists of a capacitor for storage and a transistor for access, as shown in Figure 1(a). The gate of the transistor is connected to a *wordline* (WL). Each wordline controls access to the whole row. The capacitor is connected to a *bitline* (BL) through the transistor. Each bitline is also connected to an *equalizer* and a *sense-amplifier* to convert charge on the capacitor to a digital signal. DRAM cells are organized in arrays, as in Figure 1(b). Each array is also called a bank. Usually, a DRAM chip consists of 8 banks.

The charge on the capacitor will leak over time, and data in the cell will be lost. Figure 1(a) shows several charge interaction paths where the charge may leak. The time until a cell loses its data is called the *data retention time*. To keep data for longer time, wordlines of each row must be accessed periodically, so that the sense-amplifiers recharge the capacitors of that row through the bitlines. This process is called "DRAM refresh". To ensure data integrity, every DRAM row needs to be refreshed with a certain frequency, which is usually 32 to 64ms in current DRAMs.

B. The Rowhammer Effect in DRAM

The Rowhammer effect, which is based on disturbance errors in DRAM cells, has been discovered in recent years [1]. It is an unintended side effect in DRAM that occurs when one memory row (the *hammer row*) is rapidly and repeatedly accessed. This causes cells in nearby rows to leak charge more quickly and thereby introduces changes (i.e., "bit flips") to the contents of the affected memory cells. This is due to the interaction between adjacent wordlines as well as between DRAM cells and their neighbouring capacitors and wires, as depicted in Figure 1(a). It has been shown that hammering a row will most likely affect its two adjacent rows. Consequently, *double-sided* Rowhammer, where both adjacent rows of a victim row are hammered, has been proposed to increase the chance of bit flips [2].

Usually, to allow for a sufficiently high DRAM access rate, and thus to trigger disturbance errors, non-cached memory accesses are needed, e.g., by leveraging the `CLFLUSH` instruction. Lately, several works have demonstrated the feasibility to exploit the Rowhammer effect on platforms that do not provide such cache line flush instructions. In order to circumvent CPU caching mechanisms and ensure direct access to DRAM, Gruss et al. [7] and Aweke et al. [8] enforce cache eviction through elaborate memory access patterns. Qiao and Seaborn [9] make use of x86 non-temporal store instructions, which do not use the CPU cache and van der Veen et al. [3] utilize non-cacheable DMA queries to exploit the Rowhammer effect. Other papers have presented techniques to gain understanding of the locations of flipping bits. Razavi et al. [5] presented a technique that allows for targeted bit flips at arbitrary physical memory locations by combining the Rowhammer effect with memory duplication. In order to conduct predictable Rowhammer attacks, van der Veen et al. [3] use a brute-force approach to hammer all DRAM rows and collect information about expectable bit flip locations.

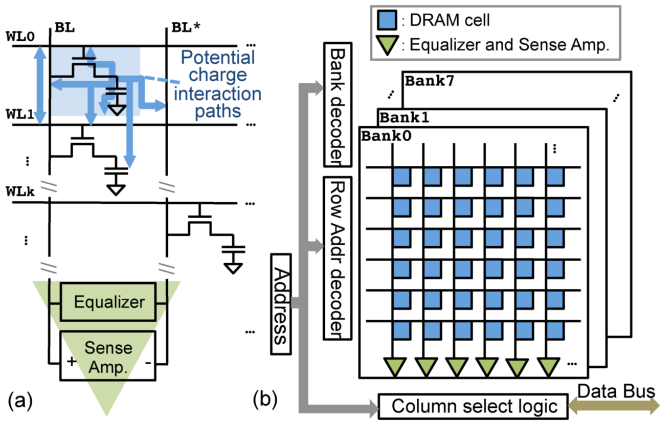

Fig. 1. (a) Schematic of DRAM cells: the blue arrows show potential charge interaction paths. (b) DRAM organization.

Since its discovery, the Rowhammer effect has been used mainly as means of attacking a computer system. In particular, changing the contents of memory cells can result in modification of important data. Seaborn and Dullien [2] as well as van der Veen et al. [3] rely on the Rowhammer effect in order to gain root privileges, by flipping bits in page table entries. Xiao et al. [4] attack Xen's paravirtualized memory isolation by employing the Rowhammer effect from within a malicious virtual machine. Razavi et al. [5] as well as Bhattacharya and Mukhopadhyay [10] successfully attack RSA by creating bit flips in keys stored in DRAM.

Meanwhile, to the best of our knowledge, this is the first work that leverages the Rowhammer effect in a positive way.

C. Memory-Based Intrinsic PUFs

In this work we focus on the class of intrinsic PUFs, which do not require the addition of extra circuits to a device in order to use them. Unlike extrinsic PUFs that necessitate addition of circuitry, e.g., arbiter circuits, intrinsic PUFs only use standard hardware components already present in commodity computer systems, such as SRAM or DRAM memory arrays. SRAM PUFs have been studied extensively and are based on the startup values that SRAM cells take on after powering on a device [11], [12]. DRAM-based PUFs have so far only leveraged DRAM cells' start-up values [13], or DRAM cell-decay effects [6]. In contrast, this work presents the Rowhammer PUF, which is a new type of an intrinsic, DRAM-based PUF that uses the Rowhammer effect.

III. Rowhammer PUF in Commodity DRAM

Previous work has shown that the locations of disturbance errors in DRAM cells are stable [1], [3]. This makes the Rowhammer effect a promising candidate for a PUF. However, the number of bit flips introduced by the Rowhammer effect can be relatively small, and thus may only provide a limited amount of entropy. We thus introduce three techniques to help increase entropy, without changing the physical DRAM properties. First, we disable DRAM refresh for those memory locations where the PUF is located. This prevents the PUF cells from being recharged, as would happen if normal refresh was on, and increases the number of bit flips. Second, multiple

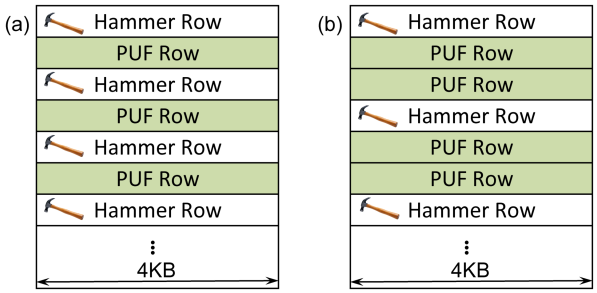

Fig. 2. Rowhammer types: (a) Double-sided Rowhammer (`DSRH`) with `PUF size` = 12KB; (b) Single-sided Rowhammer (`SSRH`) with `PUF size` = 16KB. We assume DRAM with 4KB row size.

DRAM rows are used together to create an instance of the Rowhammer PUF that encompasses larger amount of cells. Third, hammering time and initial values of the DRAM cells are controlled to induce a maximum number of bit flips.

A. Rowhammer PUF Parameters

There are many parameters that can influence the Rowhammer PUF. In Section IV, we present an evaluation upon which the most suitable values among these parameter are selected.

Rowhammer type: As presented in the literature [2], [8], there are two approaches, or Rowhammer types (`RH types`), in order to induce the Rowhammer effect. If for one victim row there is only one adjacent hammer row, used to induce bit flips, we call it single-sided Rowhammer (`SSRH`). In contrast, double-sided Rowhammer (`DSRH`) involves exposing both neighbors of a particular victim row as hammer rows. The patterns of hammer and victim rows (also called PUF rows), used to conduct `SSRH` and `DSRH` are shown in Figure 2.

PUF address and size: The `PUF address` defines the starting address of a PUF in the DRAM. The `PUF size` depends on the the number of rows that are examined after the Rowhammer process finishes (PUF rows in Figure 2). Due to the existence of hammer rows, PUF rows are not consecutive. `PUF size` and `RH type` influence the actual hammering frequency, as smaller PUFs will allow each hammer row to be accessed more frequently. Likewise, `SSRH` has fewer hammer rows, so each can be accessed more often within the same time period.

Hammer row IV (initial value): For the memory range that corresponds to `PUF address` and `PUF size`, corresponding hammer rows will be pre-initialized by writing `Hammer row IV` to it, before conducting the Rowhammer process.

PUF row IV (initial value): Similarly, all PUF rows that are included in this memory range are initialized with `PUF row IV` before the Rowhammer process is started. Both, `Hammer row IV` and `PUF row IV` are important parameters because disturbance errors are caused by interaction of DRAM cell charges. Moreover, DRAM cells represent a particular logic value using different charge states, resulting in so-called true-cells and anti-cells [14]. True-cells represent a logic '1' as charge on the capacitor and '0' as no charge, while anti-cells do the opposite. Consequently, initializing a true-cell with '0' would not lead to a bit flip in that cell. Thus, it is important to evaluate the effect of different values of `PUF row IV` and `Hammer row IV`. As the layout of true- and anti-cells is identical for DRAM modules of the same type, once optimal settings for both initial values have been found, they can be re-used for other instances of the same device type.

Algorithm 1: Process of the Rowhammer PUF query.

```
Data: RH_type, PUF_address, PUF_size,
      Hammer_row_IV, PUF_row_IV, RH_time
Result: PUF measurement m
· reserve memory defined by PUF_address and PUF_size;
· initialize PUF_rows with PUF_row_IV and hammer_rows
  with Hammer_row_IV;
· disable auto-refresh of PUF rows;
while t < RH_time do
    for r_i ∈ hammer_rows do
        · read access to row r_i;
    end
end
· enable auto-refresh;
· read PUF_rows as PUF measurement m;
```

Rowhammer time: The Rowhammer time (`RH time`) defines the total duration of the PUF measurement, including disabling the refresh rate and conducting the hammering process. `RH time`, just as `PUF size` and `RH type`, affects how many times each hammer row will be accessed in total.

B. Rowhammer PUF Access

Given the above parameters, the process of accessing a Rowhammer PUF is depicted in Algorithm 1. Based on `PUF address`, `PUF size`, and `RH type`, the DRAM region for the Rowhammer PUF is defined. First, this DRAM region is reserved, such that no other program accesses the same region. Next, the PUF rows and hammer rows are initialized with `Hammer row IV` and `PUF row IV`, respectively. The PUF query is started by disabling the DRAM auto-refresh in the next step. This is done using the same technique as employed in [6]. Subsequently, the process of hammering respective rows is started. For this purpose, the hammer rows need to be accessed repeatedly for a certain time. This is achieved by a read operation to the first word of each hammer row, which in turn causes the whole DRAM row to be refreshed. Hence, bits in the PUF rows will start to leak charge and will eventually flip. After `RH time`, the process ends and the DRAM auto-refresh is enabled again. Finally, the PUF measurement can be read from the PUF rows.

C. Factors Affecting the Rowhammer PUF

Because the Rowhammer PUF is inherently tied to the underlying physical properties of the DRAM modules, there are three factors that can influence the operation of the PUF.

Temperature: Prior work has shown that Rowhammer victim cells are not strongly affected by temperature [1]. However, the Rowhammer PUF is based on the interaction of the Rowhammer effect and DRAM decay, which was shown to be temperature-sensitive. Thus, we evaluate the temperature effect in Section IV, which confirms that the Rowhammer PUF exhibits increased bit flips but stable noise values at higher operating temperatures.

Voltage: Prior work has also shown that voltage affects the leakage in DRAM cells [15]. In commodity, off-the-shelf devices there is currently no interface to control the voltage of DRAM cells. We assume that for the Rowhammer PUF, the DRAM operates at the factory specified voltage parameters. Voltage factors will be investigated in future work.

Error Correcting Codes (ECC): ECC can be used in DRAM to protect from bit flips. Many computing platforms, such as the PandaBoard used in this work, do not have ECC implemented. Even if ECC is present, the authors of [16] showed that ECC is not enough to mitigate the Rowhammer effect. In order to use the Rowhammer PUF when ECC is used, the PUF size would have to be increased. Further, ECC registers that indicate rows, which observed bit flips, could potentially be exploited for PUF measurements. We will explore this in future work. In this paper we assume that no ECC is used.

D. Software Implementations in Commodity Devices

In this paper, the Rowhammer PUF is implemented and tested on the PandaBoard [17]. The PandaBoard ES Revision B3 used in our experiments houses a TI OMAP 4460 System-on-Chip (SoC) module and 1GB DDR2 memory from ELPIDA in a Package-on-Package (PoP) configuration, which operates at 1.2V. Our PUF implementation is purely in software, leaving hardware configuration unchanged.

The Rowhammer PUF is implemented in the U-Boot boot loader. Since the DRAM is idle during U-Boot runtime, queries to the Rowhammer PUF can be conducted without affecting other functions of the platform. In U-Boot, one can control the DRAM refresh cycle. Further, one can access physical DRAM addresses without caching[1].

The reference manuals provide the physical address mapping of the DRAM. We allocate hammer rows and PUF rows in `Bank0` and make them adjacent, as shown in Figure 2(a–b).

It is further possible to access the Rowhammer PUF from within a kernel module to achieve runtime access. Similar to U-Boot, DRAM refresh can be disabled from kernel space. Moreover, in contrast to U-Boot, where caching can be avoided by accessing the Rowhammer PUF during an early point of DRAM initialization, the kernel module allows for disabling caching by setting respective register values can be disabled if the platform does not support the `CLFLUSH` instruction.

IV. Evaluation

In this section, we will provide details on the various characteristics of the proposed Rowhammer PUF. We will first discuss how different values of the parameters presented in Section III-A affect the number of observed bit flips. We then evaluate the Rowhammer PUF on the basis of a fixed parameter configuration with regards to uniqueness, robustness and entropy. Finally, we discuss varying ambient temperature conditions that could influence the Rowhammer PUF.

[1]PandaBoard implements an ARM processor that does not provide the `CLFLUSH` instruction. Thus, we avoid caching by querying the Rowhammer PUF during an early stage during DRAM initialization, before caching is enabled by the boot loader.

TABLE I. Parameters used for evaluation of the Rowhammer PUF characteristics, and their corresponding set of values.

Parameter	Evaluated Values
`RH type`	single-sided (`SSRH`), double-sided (`DSRH`)
`PUF size`	4KB, 32KB, 128KB
`Hammer row IV`	'0x00', '0x55', '0xAA', '0xFF'
`PUF row IV`	'0x00', '0x55', '0xAA', '0xFF'
`RH time`	60s, 120s

We will follow an explorative approach, which involves assessment of a subset of all potential parameter values. Due to the lack of information about the distribution of true- and anti-cells[2], it is necessary to explore the correlation between parameter values and PUF behavior experimentally, by testing various parameter settings.

In our evaluation, three different memory regions, each located on one individual PandaBoard, are measured[3]. For all of the measurements, the PUF address was fixed. For each parameter combination, 20 measurements were taken.

A. Effects of Rowhammer Parameters

Given the high dimensional parameter space (see Section III-A), we focussed on evaluating such configuration settings that are expected to yield a good PUF. An overview of all evaluated parameter settings is given in Table I. For the sake of brevity, we only present most important results of the extensive evaluation data we obtained.

In order to extract the maximum possible entropy from PUF measurements, we primarily strive to maximize the number of bit flips. For this purpose, we first identify those parameters that have the largest effects on the amount of bit flips. In the following, we will discuss the parameters listed in Table I, in the context of their impact on observable bit flips.

The bit flips we observe in the Rowhammer PUF measurements are due to the hammering process and the DRAM cell decay that emerges after DRAM refresh is disabled. In order to confirm that the Rowhammer effect adds a significant number of extra flips, we measured the number of bit flips, which are solely caused by the decay process, and compared it to the total number of bit flips we observed in the Rowhammer PUF measurements. Compared to the bit flips caused by DRAM decay, the Rowhammer PUF introduces 2.4 times bit flips in 60 seconds and about twice the number of bit flips in 120 seconds. Further, the set of bits that flip (i.e., their locations in the measurements) obtained from the Rowhammer PUF only partially overlaps with the set of bit flips induced by the DRAM decay process, even for longer decay times (i.e., without the influence of the Rowhammer effect). Thus, the Rowhammer PUF induces new bit flips, which are at different locations compared to the DRAM decay process.

Rowhammer type: Initially, we expected the `RH type` parameter to have a strong influence on the number of flipped bits. In Figures 3(a–b), we present the *fractional* number of bit flips as a percentage of the absolute number of bits

[2]Usually this is the case when dealing with commercial, off-the-shelf devices as most vendors treat such implementation details regarding their hardware components as intellectual property and thus will not disclose them.

[3]In the following we denote such a memory region as a *PUF instance*.

TABLE II. OVERVIEW OF THE AVERAGE NUMBER OF OBSERVABLE BIT FLIPS, DEPENDING ON COMBINATIONS OF HAMMER ROW IV AND PUF ROW IV. CONFIGURATION USED: PUF SIZE= 128KB AND RH TIME= 120S (SSRH/DSRH).

	Hammer row IV			
PUF row IV	'0x00'	'0x55'	'0xAA'	'0xFF'
'0x00'	7405 / 8032	17558 / 20358	7391 / 7200	17288 / 20152
'0x55'	0 / 0	0 / 0	0 / 0	0 / 0
'0xAA'	22547 / 24480	**32904** / **37548**	14218 / 14243	24479 / 28268
'0xFF'	15633 / 17798	15402 / 17579	6132 / 6479	6095 / 6416

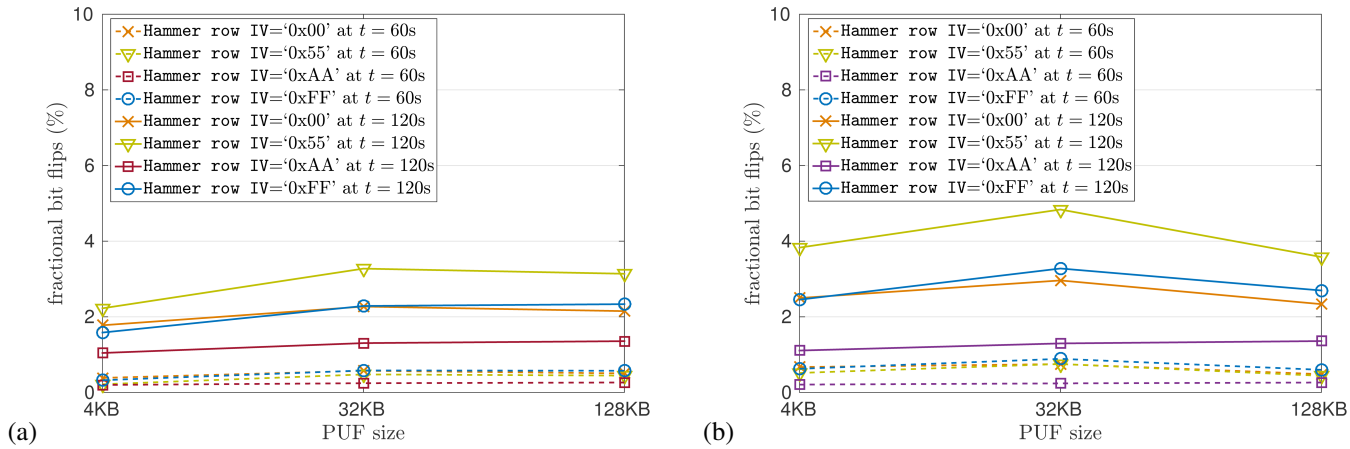

Fig. 3. Fractional number of bit flips, given in percent relative to PUF size, using PUF row IV='0xAA'. Left: number of bit flips using SSRH. Right: number of bit flips using DSRH.

available (PUF size). Contrary to our expectations, applying DSRH (right) instead of SSRH (left), does not lead to a highly increased number of flips, despite hammering both rows adjacent to respective PUF rows. Instead, compared to SSRH, using DSHR only leads to $\approx 9\%$ more bit flips in 60 seconds and to $\approx 15\%$ in 120 seconds on average.

PUF size: The PUF size influences the total time required to execute a single iteration of hammering the DRAM. In our implementation, each hammer row is accessed roughly every $6\mu s$ when hammering 2 rows (4KB PUF) and every $8\mu s$ when hammering 17 rows (128KB PUF) in the SSRH setting. Figures 3(a–b) show the number of bit flips relative to PUF size. The number of bit flips does not change significantly for different values of PUF size, i.e., the fraction of bit flips for different memory ranges stays stable.

Hammer row and PUF row IV: Given that DRAM arrays consist of true-cells and anti-cells, the initial value (IV) of the hammer rows as well as the PUF rows is expected to play an important role regarding the number of observable bit flips. Depending on the type of a cell, a bit flip in a PUF row can be observed only if the cell is initialized with the logic value that corresponds to its charged state. Similarly, due to physical interaction of charged analog elements in the hammer and PUF rows (i.e., wires and capacitors) and the resulting charge interaction paths, different IVs of the hammer rows influence the number of bit flips as well. Thus, the values of both parameters must be chosen carefully, in order to maximize bit flips. As can be seen from Table II, different configurations of Hammer row IV and PUF row IV lead to measurements that exhibit different bit flips. In general, it can be inferred from the experiments, that the number of bit flips on the PandaBoard can be maximized, if PUF rows are pre-initialized in such a way that keeps true-cells and anti-cells in the charged state, whereas cells of the adjacent hammer rows are kept in an uncharged state. In particular, the measurements show that most bit flips can be observed, if PUF rows are initialized with '0xAA', which depicts a bit-wise checkerboard pattern, with a leading '1'. Adjacent hammer rows are set up using the complementary pattern, starting with a '0' bit ('0x55'). In contrast, no bit flips can be observed when initializing PUF rows with '0x55', as in this case, cells of the PUF rows were initialized corresponding to their uncharged states.

Summary: The parameters Hammer row IV and PUF row IV have a predominant influence on the number of bit flips, which we strive to maximize. We therefore first fix their values as follows: Hammer row IV='0x55' and PUF row IV='0xAA'. We further set RH type to SSRH, as the number of introduced bit flips is in the same order of magnitude as for DSRH. Additionally, SSRH requires $\approx 55\%$ less memory and involves less memory accesses compared to DSRH. Although setting RH time= 120s leads to $\approx 400\%$ more bit flips as for 60s, we will provide evaluation results for both parameter values. The resulting PUF readout time is similar to existing runtime accessible decay-based DRAM PUFs.

B. PUF Characteristics

In order to assess the applicability of the set of flipped bits as a PUF, we validated uniqueness, robustness and entropy of the Rowhammer PUF measurements, using the parameter configuration identified above. Instead of using metrics that

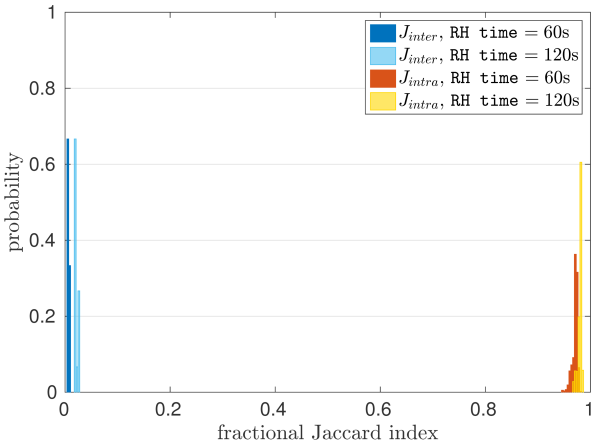

Fig. 4. Histogram of J_{inter} and J_{intra} values for three PUF instances using 20 measurements with `PUF row IV`='0xAA', `Hammer row IV`='0x55', `PUF size` = 128KB and `RH type` set to `SSRH`.

are based on the Hamming distance (i.e., inter- and intra-Hamming distance), we follow the approach of [6] and utilize the *Jaccard index* [18]. This is motivated by the fact that DRAM-based PUFs show different characteristics compared to classic PUFs (such as SRAM-based PUFs). In particular, measurements from DRAM modules draw their PUF characteristics from the location of the flipped bits. This fact (uniqueness of indices) is not properly reflected in Hamming distance-based measures. Let s_x denote the set of indices of flipped bits in the corresponding PUF measurement m_x. The Jaccard index between two measurements is calculated as $J(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$, depicting their similarity.

Uniqueness and Robustness: Uniqueness is measured by means of the J_{inter} metric. This metric compares the indices of bit flips between pairs of measurements obtained from different PUF instances. Ideally, for maximal PUF uniqueness, the two sets should have no common elements (i.e., no overlaps of bit flip locations), resulting in J_{inter} values close to '0'. In contrast, the J_{intra} metric measures the PUF's robustness, in particular the influence of noise, present in subsequent PUF measurements obtained from a fixed PUF instance. Here, the elements of two sets should be identical in the best case, resulting in J_{intra} close to '1'.

Figure 4 shows histograms of values obtained for both metrics. Clearly, both histograms separate, indicating that Rowhammer PUF instances can be robustly and uniquely identified. With a minimum J_{intra} value of 0.9454, Rowhammer PUF measurements exhibit a maximum noise of $\approx 5\%$, which can be easily corrected by standard Fuzzy Extractor constructions [19].

Entropy: PUF measurements should exhibit sufficient entropy in order to derive a cryptographic key. We estimated the entropy of the PUF measurements, as proposed in [6]. Denoting the total number of bits contained in a PUF measurement (i.e., `PUF size`) as N, k as the cardinality of s_x and assuming that the locations of flipped bits are distributed uniformly, the entropy can be calculated as follows: $H = log_2\binom{N}{k}$.

Using the minimum number of bit flips ($k = 30994$) observed in the measurements, based on the optimized parameter setting identified above ($N = 1048576 \hat{=} 128$KB), the lower bound for the fractional entropy (i.e., the entropy per cell), is 0.192. Given the vast amount of available cells, the PUF measurements show sufficient entropy to derive cryptographic keys. For example, the derivation of a 128bit key, given entropy of 0.192 requires approximately 85Bytes (excluding entropy required to compensate leakage due to helper data), while the PUF is already 128KB.

TABLE III. COMPARISON OF THE AVERAGE NUMBER OF BIT FLIPS AND MINIMUM J_{intra} VALUES OBTAINED AT OPERATING TEMPERATURES OF 40°C, 50°C AND 60°C USING THE OPTIMAL PARAMETER CONFIGURATION FOR `PUF SIZE` = 128KB.

Metric	Operational Temperature		
	40°C	50°C	60°C
avg. bit flips	32904	65431	132450
min. J_{intra}	0.9662	0.9810	0.9847

Temperature Dependency: The behavior of DRAM bit flips can be influenced by the operating temperature. In order to validate usage of the Rowhammer PUF in several operating temperatures, we evaluated the PUF at temperatures of 40°C (working temperature of DRAM on Pandaboard), 50°C and 60°C. We computed the number of bit flips and J_{intra} values for measurements taken at these respective temperatures. Table III shows the average number of bit flips as well as the minimum J_{intra} (i.e., maximum noise). The temperature evaluation shows that, while bit flips increase at higher temperatures, the noise level stays constant for each temperature. Thus, the Rowhammer PUF exhibits sufficient stability to be used at higher temperatures.

V. CONCLUSION

This paper presented the *Rowhammer PUF*, a new type of an intrinsic, memory-based PUF. Unlike the majority of work that has used the Rowhammer effect to trigger a security exploit, this is the first paper that uses the Rowhammer effect in a positive context. We extensively evaluated the Rowhammer PUF using commercial, off-the-shelf devices, not requiring custom hardware or an FPGA-based setup. The evaluation showed that the Rowhammer PUF holds required properties needed for device authentication or cryptographic key storage.

As with any new hardware security primitive, further work is required to expand the understanding of the Rowhammer PUF. Especially, we expect to perform studies investigating how voltage affects the Rowhammer PUF. Moreover, aging experiments with use of a thermal chamber need to be conducted to understand the long-term stability of this new PUF under various conditions. Finally, we will investigate how to improve the PUF readout time. The code of our design will be made available as Open Source at http://www.seceng.de/schaller/rowhammer-puf/.

ACKNOWLEDGEMENT

This work has been partly funded by the German Research Foundation (DFG) as part of project P3 within the CRC 1119 CROSSING and the German Academic Exchange Service (DAAD).

REFERENCES

[1] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *ACM SIGARCH Computer Architecture News*, 2014, pp. 361–372.

[2] M. Seaborn and T. Dullien, "Exploiting the DRAM rowhammer bug to gain kernel privileges," *Black Hat*, 2015.

[3] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *ACM Conference on Computer and Communications Security*, 2016.

[4] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation," in *USENIX Security Symposium*, 2016.

[5] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip feng shui: Hammering a needle in the software stack," in *USENIX Security Symposium*, 2016, pp. 1–18.

[6] W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Run-Time Accessible DRAM PUFs in Commodity Devices," in *International Conference on Cryptographic Hardware and Embedded Systems*, 2016, pp. 432–453.

[7] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer. js: A remote software-induced fault attack in javascript," *arXiv preprint arXiv:1507.06955*, 2015.

[8] Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, "ANVIL: Software-based protection against next-generation rowhammer attacks," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2016, pp. 743–755.

[9] R. Qiao and M. Seaborn, "A new approach for rowhammer attacks," in *International Symposium on Hardware Oriented Security and Trust*, May 2016, pp. 161–166.

[10] S. Bhattacharya and D. Mukhopadhyay, "Curious case of Rowhammer: Flipping Secret Exponent Bits using Timing Analysis," in *International Conference on Cryptographic Hardware and Embedded Systems*, 2016, pp. 602–624.

[11] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, *FPGA intrinsic PUFs and their use for IP protection*. Springer, 2007.

[12] G.-J. Schrijen and V. van der Leest, "Comparative analysis of SRAM memories used as PUF primitives," in *Conference on Design, Automation and Test in Europe*, 2012, pp. 1319–1324.

[13] F. Tehranipoor, N. Karimina, K. Xiao, and J. Chandy, "DRAM based Intrinsic Physical Unclonable Functions for System Level Security," in *Great Lakes Symposium on VLSI*, 2015, pp. 15–20.

[14] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," in *ACM SIGARCH Computer Architecture News*, 2013, pp. 60–71.

[15] T. Hamamoto, S. Sugiura, and S. Sawada, "On the retention time distribution of dynamic random access memory (DRAM)," *IEEE Transactions on Electron Devices*, pp. 1300–1309, 1998.

[16] B. Aichinger, "DDR memory errors caused by Row Hammer," in *High Performance Extreme Computing Conference*, 2015, pp. 1–5.

[17] "PandaBoard," http://www.pandaboard.org accessed Nov. 2016.

[18] P. Jaccard, *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901.

[19] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 2004, pp. 523–540.