PUF the Magic DRAmGON: Persistent Storage in Volatile Memory using Rowhammer PUF

Emiliia Gelóczi*,[1], Martin Heckel*,[2], Stefan Katzenbeisser[1], and Florian Adamsky[2]

[1] University of Passau, Passau, Germany
[2] Hof University of Applied Sciences, Hof, Germany
*These authors contributed equally to this work.

Abstract. Physical Unclonable Functions (PUFs) exploit inherent manufacturing variations in electronic devices to create unique hardware fingerprints, which are typically used as a source of randomness for generating device-specific cryptographic keys. In this work, however, we explore the possibility of repurposing PUFs as a secure storage mechanism for pre-existing keys. First, we evaluate existing Rowhammer-based PUFs (RH-PUFs), introduce five novel RH-PUF constructions, and compare all of them with respect to their suitability for cryptographic key generation. Second, we propose DRAmGON, a novel, unconventional PUF-based approach that enables the direct encoding of cryptographic keys into Rowhammer-susceptible DRAM, thereby allowing persistent data storage in volatile memory. Our experiments using FlippyRAM show that 81 % of the tested systems exhibit Rowhammer susceptibility, establishing RH-PUFs as a viable hardware-based security primitive.

Keywords: Rowhammer · Rowhammer PUF · FlippyRAM · DRAmGON

1 Introduction

Physical Unclonable Functions (PUFs) are a promising solution for ensuring key security, as they enable the generation of highly random, unique-per-device keys derived from variations in the physical characteristics of device components introduced during manufacturing [1]. Moreover, such keys can often be generated on demand, thereby eliminating the need for persistent key storage. PUF-derived keys are applicable in a wide range of security scenarios [2]; however, in some cases it is necessary to securely store an existing cryptographic key, e. g., one that was previously shared among various parties.

To address this requirement, modern computers use dedicated security hardware, such as Trusted Platform Modules (TPMs). TPMs, however, exhibit several limitations: they require additional external hardware integration; they are not yet universally deployed [3]; and they may enforce vendor-imposed restrictions that can potentially violate user privacy [4]. In this context, PUFs also emerge as a promising alternative. While traditionally used as entropy sources for device-specific key generation, we argue that PUFs can be repurposed to enable the secure storage of pre-generated cryptographic keys.

In this paper, we use Rowhammer-based PUFs (RH-PUFs), repurposing an originally malicious technique for security applications [5]. The Rowhammer effect is a read-disturbance phenomenon in Dynamic Random-Access Memory (DRAM) that can induce bit flips in memory rows physically adjacent to those repeatedly accessed by an adversary [6]. During a Rowhammer attack, the occurrence of these bit flips is non-deterministic and device-specific, properties that make them suitable for constructing PUFs [5]. However, existing research on RH-PUFs provides limited insight. Early studies relied on outdated memory technologies such as DDR2 and DDR3 [5, 7], while more recent works evaluate only a small number of samples [8–10], rendering the general validity of their results slightly uncertain. We therefore hypothesize that the potential of RH-PUFs has been underestimated.

To explore the potential of PUFs not only as a source of entropy but also as a source of stability for persistent storage, we conduct a broad study of RH-PUFs to deepen our understanding of their behavior and address limitations in existing research. In summary, we provide the following *contibutions*:

- We identify 44 of 54 devices as susceptible to Rowhammer, representing 81 % of the tested systems, thereby demonstrating that RH-PUFs are a viable hardware-based security primitive (see Section 3.3). These results also confirm and extend the results presented by Heckel et al. [11].
- We introduce five novel RH-PUF constructions and show that they yield better results than existing ones using key PUF metrics (see Section 4).
- We propose DRAmGON[1], a novel and unconventional approach that leverages PUFs not as a source of randomness but as a medium for key storage. It enables secure, persistent storage of secret keys in volatile DRAM memory, which, to the best of our knowledge, has not been demonstrated before (see Section 5).

Outline. Section 2 provides background information relevant to our research. Section 3 describes our experimental setup and methodology including our results for the fully automated Rowhammer tests. Section 4 introduces and evaluates the novel RH-PUF constructions. Section 5 presents and discusses a key encoding approach DRAmGON. Section 6 outlines works related to our research. Section 7 concludes the paper and outlines future research directions.

2 Background

In this section, we provide a brief overview of concepts related to our work.

[1] Source code available at `https://github.com/iisys-sns/PUF_the_Magic_DRAmGON_evaluation`

2.1 Rowhammer

DRAM is the main memory used in current computer systems. DRAM cells store one bit as an electrical charge on a capacitor, are arranged in rows and columns, and are organized into banks with a row buffer. DRAM banks typically span multiple DRAM *chips*, each chip containing parts of several banks. DRAM chips are grouped into ranks, and one or more ranks form a Dual In-line Memory Module (DIMM), which is connected to the CPU.

When a DRAM row is accessed, its content is loaded into the buffer, destroying the original data. Before another row in the same bank can be served, the buffered data must be written back, causing a *row conflict*. Repeatedly triggering such conflicts can induce bit flips in neighboring rows, enabling modification of memory content without direct interaction [6]. This technique is known as a *Rowhammer* attack: the repeatedly activated rows are *aggressors*, while the rows in which bit flips occur are *victims*.

Rowhammer attacks have been demonstrated across a wide range of scenarios [6, 9, 12–31], spanning different platforms, targeted assets, access patterns, and DRAM generations. In response, a variety of mitigation techniques have been proposed that reduce the number of bit flips [32–36]; however, they cannot eliminate them entirely [27, 37, 38] and can often be bypassed [12, 14, 17, 19–21].

2.2 Rowhammer Physical Unclonable Functions

Manufacturing variations cause even identically produced devices to exhibit unique physical characteristics that can be exploited to derive a hardware fingerprint known as a PUF [1], which maps a challenge C to a device-specific response R, typically represented as a bit string.

Because PUF responses are derived from physical effects, they are inherently noisy and not always reliably reproducible. Thus, the raw response is typically processed using a fuzzy extractor [39], which applies error-correcting code (ECC) to eliminate noise and randomness extractor (e.g., hashing) to transforom the original R in a uniform and cryptographically secure key. Although a fuzzy extractor can mitigate moderate noise or bias, a reasonably "good" raw response is required to avoid unnecessary resource overhead and to ensure stronger security guarantees. Therefore, the response R corresponding to a given challenge C should be reliably reproducible on the intended device (*reliability*), yet be significantly different on another device (*uniqueness*). Ideally, the response should also be *uniform*, i. e., not be biased toward either 1 or 0, and exhibit *high entropy* [40].

PUFs can be derived from different components. In research, delay-based PUFs based on-chip elements (e.g., arbiters) are widely studied, whereas in practice most deployed PUFs are memory-based, as memory modules are present in virtually all devices and allow simple PUF extraction. In this work, we focus on RH-PUF that exploits Rowhammer-induced bit flips to generate device fingerprints [5]. In this context, the initial PUF address, response length, data pattern, measurement duration, and Rowhammer type typically form the challenge. The response can be defined in different ways, for example, by the addresses of the observed bit flips [8], or the interpretation of bit flip patterns [41].

3 System Model, Experimental Setup and Methodology

This section describes the system model, experimental setup, and methodology used to construct and evaluate RH-PUFs (see Section 4) and the DRAmGON encoding approach (see Section 5), as well as to validate selected results from the large-scale Rowhammer study by Heckel et al. [11] (see Section 3.3).

3.1 System Model & Experimental Setup

We consider a set of 54 commodity desktop systems, of which 44 exhibit observable Rowhammer-induced bit flips. All systems are deployed in institutional computer rooms, representing a realistic operating environment, and equipped with Intel CPUs and DDR4 DRAM *hardware* (see Table 1), as existing Rowhammer tools have been shown to operate reliably in this setting [11]. We further assume that system configurations and components remain unchanged, since even transferring a DIMM between machines can alter Rowhammer behavior [42].

Table 1. Hardware specification of the systems involved in the experiments.

ID	CPU	Memory Configuration	Affected	Total
S_{01}	i9-10900K	1x 1Rx16 (8 GiB) DDR4 2400	1	1
S_{02} – S_{27}	i7-8700	2x 1Rx16 (8 GiB) DDR4 2666	26	30
S_{28} – S_{44}	i7-8700	2x 2Rx16 (16 GiB) DDR4 2666	17	23

All experiments are conducted in a Linux-based environment booted from an external USB drive. We use an adapted version of the FlippyRAM Rowhammer *software* [11], which includes AMDRE [43], TRRespass RE [14], and Dare [20] for the DRAM address-mapping reverse-engineering. All Rowhammer tools are disabled except Blacksmith [12], which demonstrates strong performance on DDR4 DRAM [11]. It is executed for 300 s to verify correct operation without introducing additional overhead. Blacksmith induces bit flips in memory by repeatedly accessing rows in non-uniform, complex patterns. It first identifies effective row access patterns and then hammers these rows to create electrical interference in adjacent rows, thereby triggering bit flips.

3.2 Methodology

For each system, we first determine suitable Rowhammer access patterns and corresponding DRAM address mappings using the reverse-engineering tools described in Section 3.1. We then run Blacksmith in fuzzing mode and select the first pattern that produces at least 5 000[2] bit flips.

Next, we repeatedly execute Blacksmith in sweeping mode with the selected pattern and record all observed bit-flip metadata for analysis. This includes:

[2] Based on a pretest showing that approximately 10 % of bit flips from a single sweep remain stable across 20 runs, consistent with prior work [5, 7].

addr, the virtual address[3]; *bitmask*, indicating which bits flipped within the byte; *data*, the value read after the Rowhammer execution; *dram_addr*, the corresponding DRAM location (*bank*, *row*, and *column*); and *observed_at*, a timestamp of the bit flip occurrence.

Due to operational constraints, all experiments were conducted during weekends, yielding an execution window of approximately 52 h. Because systems vary in their susceptibility to Rowhammer, we performed between 5 to 151 sweeping runs per system, averaging 67 measurements per device.

3.3 Large-Scale Rowhammer Evaluation

Heckel et al. [11] performed a large-scale study on Rowhammer susceptibility using 1 006 datasets from 822 unique systems. They report 27.8 % of the datasets being susceptible, considering only those for which addressing function reverse-engineering worked. They hypothesized that the actual susceptibility is higher and show that longer testing leads to more accurate detection of affected systems. Their [11] results reveal that Blacksmith [12] is the tool with the best results on systems with DDR4 DRAM and is 10 times more effective at finding susceptible systems than the second-best tool, Rowpress [15]. They also show that out of 126 datasets affected in total, 123 were Intel CPUs. So, Blacksmith yields good results on systems with Intel CPUs and DDR4 DRAM.

Using our experimental setup, we verify their hypothesis on a smaller scale, involving 54 systems from two computer rooms at our institution. In one room, there are 30 systems, and in the other room, there are 23 systems as described in Section 3. All systems have Intel CPUs and DDR4 DRAM, so we decided to use Blacksmith as it yields good results on this kind of hardware [12] We run Blacksmith over a whole weekend (a significantly longer time frame than in the large-scale study of Heckel et al. [11], which ran 7 Rowhammer tools and split the total runtime equally among them). We show that the susceptibility is significantly higher as the 27.8 % identified in the large-scale study: We identify 44 of 54 tested systems (81 %) to be susceptible to Rowhammer using the FlippyRAM framework with some minor adjustments described above.

4 Rowhammer-based PUF Constructions

This section introduces several existing and novel (marked below with *) RH-PUF constructions, and then evaluates their properties. For each system, we split collected Blacksmith sweeping runs (measurements) into a *training* set (20 %), from which a reference response is derived using majority voting, and a *testing* set (80 %), which is used for response reproduction. To avoid trivial all-zero responses caused by flip sparsity, we select for each system a continuous 2 KiB *Best Memory Region (BMR)* that exhibits the highest number of bit flips in the training set. Finally, the challenges for all RH-PUFs in this section include the Rowhammer pattern used and the corresponding most effective address mapping.

[3] The 1 GiB hugepage is mapped to the virtual address `0x2000000000`, so the offset can be calculated by substracting that address from the value of the *addr* field.

4.1 RH-PUF Constructions

All RH-PUFs described in this section are illustrated in Appendix A (Figure 5).

(I) Bitmask Snapshot is based on the work of Schaller et al. [5]. In this RH-PUF, the challenge corresponds to a memory region, specifically, the BMR in our case. The response is constructed from the bitmasks associated with each address within this region. However, due to the sparsity of Rowhammer-induced bit flips, most bitmasks are `0x00`, which biases the response toward zeros.

(II) Affected Addresses uses as a challenge the set of addresses that exhibit bit flips [8]. The response encodes each bit-flip location by its bank (bb), row ($rrrr$), and column ($cccc$), yielding a 40-bit response segment per address. However, because Blacksmith hammers only a single bank, the bank encoding repeats across the response, which significantly reduces its entropy.

(III) Affected Bitmasks.* To improve the characteristics of the Bitmask Snapshot approach [5], we propose a variant in which the challenge includes only addresses that exhibit stable bit flips. Accordingly, the response contains only non-empty bitmasks. Although scanning the full memory can yield thousands of candidate flips, we restrict the response size to 2 KiB by using the first 2 048 affected bitmasks. This approach improves bit balance and entropy compared to the original method; however, bit flips remain sparse, as typically only a single bit flip per byte, leaving most response bits at zero.

(IV) Affected Addresses Short* is a variation of the *Affected Addresses* method [8]. To mitigate the limitation caused by the repeated bank encoding in the original approach response, we encode each bit-flip location using only the row and column components ($rrrr\ cccc$). As a result, each challenge address corresponds to a 32-bit response segment. This modification removes the explicit bank pattern and improves both entropy and bit balance.

(V) Flip Existence.* We propose an approach based on binary information indicating whether a bit flip occurred at a specific address. The challenge is a memory region, specifically BMR. The response consists of bits denoting the presence (1) or absence (0) of a flip in each byte of the region. Compared to the *Bitmask Snapshot* approach, this compression improves entropy and bit balance but yields shorter responses.

(VI) Flip Combo (Existence and Direction).* Another novel approach considers not only the presence of bit flips but also their direction. The challenge corresponds to a memory region, specifically the BMR. The response encodes each address in this region using a 2-bit code: `00` if no bit flip occurred, `10` if a flip from 1 to 0 occurred, and `11` if a flip from 0 to 1 occurred. Although Blacksmith does not explicitly report the flip direction, it can be derived by computing the $bitmask \oplus data$.

(VII) Flip Direction* relies solely on the direction of bit flips. The challenge is the list of addresses at which bit flips occur stably, and the response is the list of flip directions at each address: 0 for a flip from 1 to 0, and 1 for a flip from 0 to 1, resulting in one response bit per byte. As both flip directions occur with similar frequency [6], responses are well balanced, but their length varies across systems and may be limited when few stable flips are available.

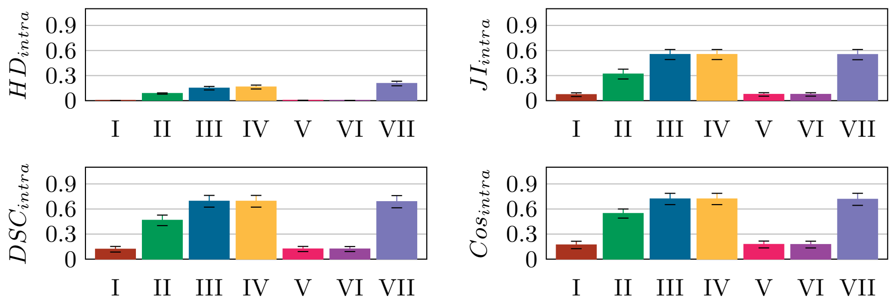

Fig. 1. Average results of reliability metrics for responses derived from different RH-PUF constructions: (I) Bitmask Snapshot, (II) Affected Addresses, (III*) Affected Bitmasks, (IV*) Affected Addresses Short, Flip (V*) Existence/ (VI*) Combo / (VII*) Direction across evaluated systems. Error bars indicate 99 % confidence intervals.

4.2 Evaluation of RH-PUFs

Using the collected experimental data, we implemented each RH-PUF type[1] on all available systems that we described in Section 3.1 and evaluated the following properties: reliability, uniqueness, uniformity and entropy.

Reliability measures how consistently a system reproduces the same response for an identical challenge [40]. We assess reliability by comparing reproduced responses to a reference using four metrics: *fractional intra-device Hamming Distance* (HD_{intra}, ideal value: 0), which represents the fraction of differing bits; *intra-device Jaccard Index* (JI_{intra}, ideal value: 1), which captures the consistency of flipped addresses; *intra-device Dice-Sørensen Coefficient* (DSC_{intra}, ideal value: 1), which places greater weight on matches and is less sensitive to sparsity [44]; and *intra-device Cosine Similarity* (Cos_{intra}, ideal value: 1), which measures angular similarity rather than direct set overlap [45]. Figure 1 summarizes the average results across all evaluated systems. In general, HD_{intra} remains low and close to 0, especially for zero-dominated responses ((I) Bitmask Snapshot , Flip (V*) Existence/(VI*) Combo). Although other constructions exhibit slightly higher HD_{intra} values, these remain correctable with appropriate ECC schemes [46], albeit at the cost of increased overhead as instability grows. In contrast, JI_{intra} indicates high instability of the observed bit flips in sparse-response constructions, while the remaining approaches exhibit only moderate stability (0.3 to 0.6), confirming previous findings that RH-PUFs can be inherently unstable. DSC_{intra}, being less sensitive to sparsity, highlights this contrast more clearly. Cos_{intra} follows the same trend but yields slightly higher values, as it captures proportional overlap rather than absolute bit matches.

Uniqueness measures how well responses from different systems can be distinguished under the same challenge [40]. We evaluate uniqueness by applying each system's reference challenge to all other systems and comparing the resulting

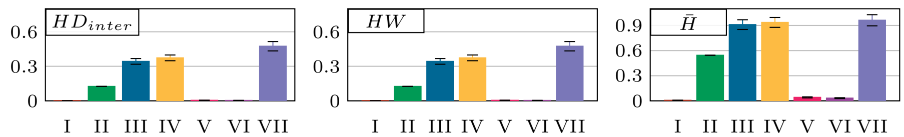

Fig. 2. Average uniqueness, uniformity and entropy of responses derived from different RH-PUF constructions: (I) Bitmask Snapshot, (II) Affected Addresses, (III*) Affected Bitmasks, (IV*) Affected Addresses Short, Flip (V*) Existence/ (VI*) Combo / (VII*) Direction across evaluated systems. The error bars depict 99 % confidence intervals.

responses using the same metrics as for reliability, but in an inter-device setting. The *inter-device Hamming Distance* (HD_{inter}) ideally should approach 0.5. However, as shown in Figure 2, (I) Bitmask Snapshot and Flip (V*) Existence/(VI*) Combo exhibit $HD_{inter} \approx 0$, indicating very poor uniqueness, as their responses are dominated by zero bits and are therefore nearly identical across systems. Affected (II) Addresses/(III*) Bitmasks/(IV*) Addresses Short perform slightly better but remain insufficient for reliable fingerprinting, while (VII*) Flip Direction shows the highest uniqueness with $HD_{inter} = 0.47$, producing well-distinguishable responses across systems. DSC_{inter}, JI_{inter}, and Cos_{inter} are omitted because they emphasize overlap of set bits. In sparse RH-PUF responses, where ones are rare, these metrics are dominated by 1-bit intersections and do not capture full-response uniqueness. In contrast, HD_{inter} accounts for mismathces across all bit positions.

Uniformity & Entropy. Uniformity measures how balanced a response is, i. e., the ratio of zeros to ones, which ideally approaches 0.5 [40]. For RH-PUFs, however, uniformity primarily reflects the sparsity of Rowhammer-induced bit flips. Naturally occurring responses, such as full memory snapshots, are heavily biased toward zero and exhibit uniformity close to zero. We therefore report uniformity for completeness and complement it with an entropy metric that quantifies bit unpredictability [46]. Figure 2 summarizes the results. We evaluate uniformity using *fractional Hamming Weight* (HW) and entropy *Shannon entropy* ($\bar{H}$). (I) Bitmask Snapshot and Flip (V*) Existence/(VI*) Combo show near-zero uniformity that is thus indistinguishable in the plot. (III*) Affected Bitmasks improves uniformity to 0.12 on average but remains far from the ideal. (II) Affected Addresses (0.34) performs better and is slightly outperformed by (IV*) Affected Addresses Short (0.37), while (VII*) Flip Direction achieves the highest uniformity, reaching $HW = 0.47$. Entropy follows a similar trend: the sparsity-dominated approaches (I,V*,VI*) exhibit near-zero entropy, whereas (III*) Affected Bitmasks reaches 0.54, and the remaining approaches exceed 0.9.

Summary. Our evaluation of the implemented RH-PUFs shows that some constructions, e. g., Affected (II) Addresses/(III*) Bitmasks, and (VII*) Flip Direction, achieve near-ideal uniformity, high entropy, and response lengths aligned with cryptographic keys security recommendations [47]. Most constructions also achieve reasonable uniqueness, although some ((I) Bitmask Snapshot, Flip (V*) Existence/ (VI*) Combo) consistently underperform, while (VII*) Flip Direction shows the strongest overall results.

RH-PUFs exhibit an inherent trade-off between security and reliability. Constructions based on only flipped addresses improve reproducibility but may leak structural information, such as the approximate flip density, which increases the likelihood of partially guessing responses. In contrast, approaches that derive responses from memory regions provide higher entropy at the cost of increased instability. This instability can be mitigated using appropriate ECC schemes (e.g., typical for PUFs Bose-Chaudhuri-Hocquenghem (BCH) codes [48, 49]), albeit with additional resource overhead. Furthermore, the use of Blacksmith introduces potential information leakage due to predictable access patterns, which allows an adversary to infer pre-Rowhammer memory values and estimate likely flip directions (e.g., `0x01` has one possible $1 \rightarrow 0$ flip but seven possible $0 \rightarrow 1$ flips) [6], thereby reducing response entropy.

Overall, the choice of RH-PUF design should balance reliability and security requirements and be tailored to the target system and application.

5 DRAmGON

Traditionally, PUFs are used as a source of deterministic randomness to derive device-specific fingerprints or keys that are stable per device yet unpredictable. In contrast, we explore the potential of using PUFs not to generate keys, but to *store* already existing ones. We introduce DRAmGON, an unconventional approach that employs RH-PUFs not as a randomness source but as a persistent storage for arbitrarily chosen keys, enabling permanent storage in volatile DRAM. The key itself can be generated using any suitable mechanism, such as a cryptographically secure pseudorandom number generator. We construct challenges that reproduce the desired key as the concatenation of responses on a specific device while yielding different responses on other devices. Furthermore, our approach supports encoding multiple key–challenge pairs on a single device, allowing different keys to be stored for distinct services or providers.

In DRAmGON, the intrinsic quality metrics of raw PUF responses are of secondary importance, since the key–challenge mapping is constructed explicitly. Instead, the critical factors are the achievable response length and the stability of the underlying bit flips, which determine whether a key of the desired length can be encoded and reliably reconstructed. For this reason, DRAmGON employs the *(V*) Flip Existence* RH-PUF, as it yields the shortest responses while still exhibiting sufficiently good HD_{inter}. This makes it well-suited for demonstrating the feasibility of our approach, even under challenging edge-case conditions.

5.1 DRAmGON Encoding based on RH-PUF

Our approach is based on byte offsets within the 1 GiB hugepage mapped by Blacksmith [12]. These offsets are treated as addresses, since Blacksmith maps the hugepage to the same virtual address on each run. We use 20 % of the sweeping runs to identify the most *stable* bit flips. The *stability* of a specific address is defined as the number of runs in which a bit flip occurred at that address across all runs used for construction. Next, we encode the key by selecting offsets (addresses) based on the bits of the key to be stored:

- **Key bit is "1"**: The offset with the highest possible stability.
- **Key bit is "0"**: A random offset at which no bit flip occurred, ensuring it follows the distribution of addresses encoding a one.

The selection of addresses must prevent an adversary from inferring whether a bit flip occurred at a given address. Systems use linear DRAM addressing functions that are similar across devices (i. e., there is a limited set of possible functions). Since we use only the most effective mapping for the sweeping runs, Blacksmith effectively hammers only a single DRAM bank, so all bit flips are on that bank. Therefore, the addresses encoding zeroes have to be chosen in a way that is on the same DRAM bank that was hammered and, therefore, on the same bank that contains the addresses encoding ones.

To avoid peaks in the distribution of addresses, which may occur when addresses of stable bit flips are close together, we use a Monte Carlo simulation to generate random addresses for encoding zeroes. First, we select all addresses used for encoding ones and sort them by address. Then, we iterate over the list of addresses and calculate the ranges between the last flipped offset and the current offset. Additionally, we define two ranges: one between the minimum address in the area and the address of the first bit flip, and another between the address of the last bit flip and the maximum address in the area. For each of these ranges, we calculate its width and assign a probability by dividing the width of the smallest range by that of the current range, so that smaller ranges receive higher probabilities. We then generate two random numbers: one is an address between the maximum and minimum addresses of the range (constrained to remain within the same bank), and one representing a probability value between 0 and 1. We identify the range in which the random address lies and compare the generated probability value with the assigned one for that range. An address is added only if the generated probability is less than or equal to the range's assigned probability. On average, the Monte-Carlo approach for address selection takes 1.81 s. The total construction of the PUF challenge takes 1.84 s in average.

5.2 Experimental Evaluation of DRAmGON

For all results presented in this section, we encode a random 256 bit key using 20 % of the measurements for challenge construction. We were unable to generate the required offsets on systems S_{30}, S_{31}, and S_{41} due to insufficient bit flips during sweeping. Although configured identically to other systems, their Rowhammer susceptibility differs, so they were excluded from the evaluation.

To prevent information leakage via the challenge addresses, we use a Monte Carlo simulation to ensure that the distributions of addresses encoding 0 and 1

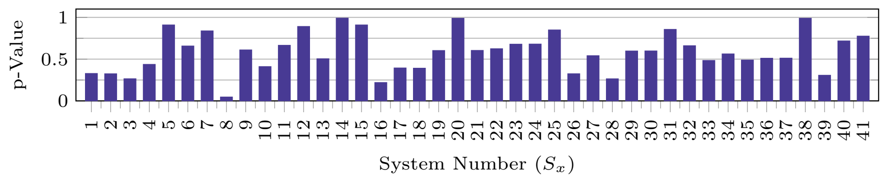

Fig. 3. p-Values from the Mann-Whitney U test ran on the considered systems.

are similar. We then apply a Mann–Whitney U test to assess whether these two distributions are statistically indistinguishable across all systems (see Figure 3).

We specify an alpha value of 0.05; thus, the value of 0.038 observed on S_{08} falls below this threshold, while all other values exceed it. To address the issue of multiple comparisons, we apply a Holm-Bonferroni correction, with which all p-values are greater than 0.05. Therefore, the distributions of addresses encoding a 1 and those encoding a 0 are sufficiently similar, preventing an adversary from inferring the stored value at a given address based on the address itself.

Next, we verify that addresses at specific offsets do not exhibit a bias towards encoding 1 or 0. First, we sort all addresses used in a challenge by their value. Then, we generate 10 challenges per system, yielding a total of 410 responses, which we compile into a single list. For each offset, we count the number of ones and compute the fraction, which should ideally be approximately 50 %. The results presented in Figure 4 show that a total of 9 out of 256 bits (3.52 %) are below 45 % or above 55 %, so 94.48 % are within 50 % ± 5 %. The first and last offsets tend to encode a 0 in 63 % or 62 % of the cases, which happens due to the distribution of zeroes generated by the Monte Carlo simulation. While it might be possible to specify additional regions with specific probabilities to achieve a more uniform distribution of ones and zeroes at the first and last offsets, such parameters may not generalize across systems, e. g., other systems may still exhibit bias. Therefore, we recommend generating responses with $n+2$ bits and removing the smallest and largest addresses from the challenge.

We also analyze the *uniqueness* of DRAmGON by computing the average inter-device fractional Hamming Distance HD_{inter} between the key and responses generated on other systems across all measurements (see Figure 4). The observed average HD_{inter} is 0.5, matching the ideal value and indicating that applying the same challenge to different systems yields statistically independent responses. Consequently, the encoded key cannot be reproduced on other devices, even when they share the same hardware configuration.

To evaluate the *reliability* of DRAmGON, we compute the average intra-device fractional Hamming Distance (HD_{intra}) between the encoded key and the reproduced responses for each system across all measurements (see Figure 4). According to Shannon's noisy-channel coding theorem, an average HD_{intra} of 0.2 is correctable with suitable ECCs. Across the 41 evaluated systems, 15 (37 %) exceed this threshold. When relaxing the bound to 0.25, only 4 systems (10 %) remain above it. These results indicate the need for pre-processing of RH-PUF responses prior to applying DRAmGON, particularly through ECC.

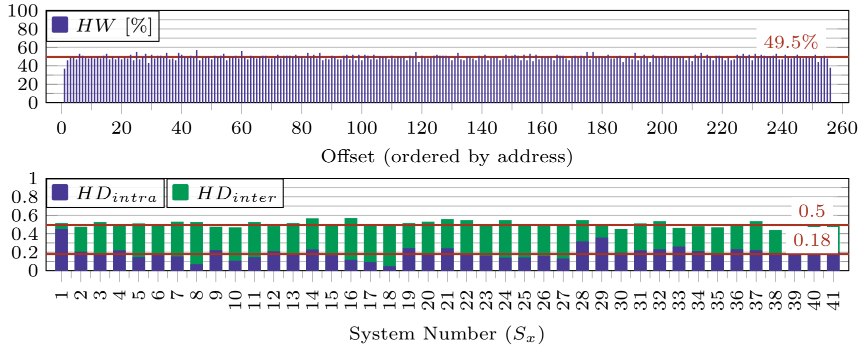

Fig. 4. Fractional Hamming Weight for each offset over 10 generation runs and average fractional intra-/inter-device Hamming Distances between the encoded key and the responses of all measurements on a specific system.

Here, we face a trade-off between security and reliability. The used RH-PUF construction is highly secure by design, but exhibits relatively low reliability, with error rates of up to 18 % on average. Thus, it is required to use ECC before applying DRAmGON[3]. Although the observed error rate is correctable, it demands more resources compared to lower error rates (e.g., below 10 %). Therefore, the choice of error correction scheme may vary depending on the system, ranging from simpler to more complex constructions.

For our scenario, suitable options include BCH, Low-Density Parity-Check, or concatenated codes. On average, coorecting a 256-bit key requires three redundancy bits per input bit, resulting in an initial response of around 1 Kbit. This requirement is easily met, as all our RH-PUFs produce at least 5000 bit flips. Additionally, ECC introduces helper data of similar magnitude $\approx$ 1 Kbit, which is public and used for key reconstruction. This is roughly twice the amount required for 10 % error rates; however, given the use of desktop systems, the resulting overhead remains negligible.

5.3 Discussion

This section discusses the security, performans and limitations of DRAmGON.

Adversary Model. We consider an adversary whose objective is to predict the cryptographic key encoded using DRAmGON. The adversary is assumed to know all public parameters, including the encoding scheme, challenge and helper data for used ECC. While unprivileged code execution on the system is possible, the adversary cannot access the hugepage memory region used by Blacksmith. Invasive physical attacks and denial-of-service attacks are out of scope; consequently, the adversary has no physical access to the target system, cannot modify hardware or firmware, and cannot replace DRAM modules.

[3] Unlike conventional PUF-based systems, DRAmGON does not require a full fuzzy extractor, as only error correction is needed.

Security. Under the assumed model, an adversary may execute unprivileged code on the system and, in principle, invoke Blacksmith. However, successful key reconstruction requires access to the specific 1 GiB hugepage initially allocated by Blacksmith. Identifying or accessing this hugepage without root privileges is infeasible within a limited time window unless the adversary is present from system initialization. However, such a scenario would invalidate most security guarantees and is therefore explicitly excluded from our model. Additionally, available access to the helper data used for ECC does not benefit the adversary, as it is designed not to reveal any sensitive information about the key.

DRAmGON is a key-storage mechanism rather than a security protocol. In a naive protocol design, an adversary guesses key bits by modifying challenge addresses and observing unchanged responses, which would likely indicate encoded zeros. Such inference attacks can be prevented by embedding DRAmGON in a secure protocol, e. g., using the reconstructed key with authenticated encryption (AES-GCM) over structured data and sequence numbers. Any decryption or freshness failure results in no response, preventing leakage of key information.

Performance. Time Overhead. We use an unmodified version of Blacksmith [12] that does not require root privileges beyond the initial 1 GiB hugepage allocation. After fuzzing, we perform sweeping runs with the most effective pattern; a full sweep takes 38.54 min on average (2.3 s per pattern replay over 1 024 locations). Restricting execution to only challenge locations reduces the time for generating 256 encoded bits with DRAmGON to ≈4.8 min. Further speedups would require low-level system access or DRAM modifications (e.g., disabling refresh), which are impractical and may compromise system security. As this work presents a proof of concept, we focus on feasibility rather than runtime optimization; performance improvements are left for future work.

Memory Overhead. DRAmGON requires approximately 125.7 MB per system on average, including 1.2 KiB for challenge (256-bit key), 1 Kbit for ECC helper data, and Blacksmith-related primitives (e.g., Rowhammer patterns and configuration data). This overhead is negligible for modern desktop systems.

Limitations. The main concern is the *application scale* of DRAmGON, as not all systems are sufficiently susceptible to Rowhammer and some exhibit reliability issues. However, susceptibility may improve with deeper system access (e. g., firmware level, vendor support), and reliability can be mitigated using ECC. It is important to emphasize that DRAmGON is a proof of concept for persistent key storage in volatile memory and should be viewed as an initial feasibility demonstration rather than a fully deployable solution. The concept can also be extended to more stable PUFs, such as SRAM PUFs.

The *number of keys* that can be encoded with DRAmGON is inherently limited. To prevent key cross-inference between services, challenges must not reuse memory addresses, which bounds the total number of keys due to the finite number of reliable Rowhammer-induced bit flips. More generally, DRAmGON is applicable only to systems with sufficient Rowhammer susceptibility to provide enough stable bit flips for the desired key lengths.

Long-term stability of DRAmGON may raise concerns. Accelerated aging studies indicate that DRAM remains stable for at least 1–2 years, while its susceptibility to Rowhammer increases, allowing bit flips to be induced faster [50, 51]. At the same time, bit-flip patterns remain relatively stable over time [5, 7, 41]. This suggests that DRAmGON remains stable over similar periods, although reconfiguration may be required over longer lifetimes if ECC limits are exceeded.

Finally, *temperature variations* can affect DRAmGON. Prior works [5, 7] show that variations within $\approx 10°C$ have negligible impact on bit-flip behavior; thus, for indoor deployments, DRAmGON can be considered stable. In contrast, extreme variations (e.g., $\pm 50°C$), such as in aviation, significantly impact stability. Nevertheless, DRAmGON remains applicable with adaptations, e.g., using different ECC helper data for different temperatures (30–40°C, 40–50°C, etc.). The overhead is minimal, as each helper data typically is less then 1 Kbit.

6 Related Work

The first use of Rowhammer for security was presented by Schaller et al. [5], who leveraged Rowhammer-induced bit flips to generate a unique device fingerprint, i.e., PUF. In their design, the initial PUF address and size, data pattern, measurement time, and Rowhammer type form a challenge, while the selected rows serve as a response. Implementation on DDR2 demonstrated good uniqueness, robustness, and entropy, with flip rates increasing at higher temperatures while remaining stable under fixed conditions. Anagnostopoulos et al. [7] later confirmed these findings across a wider temperature range.

Li et al. [8] proposed *FPHammer*, a DRAM-based device identification framework. They use the PUF address, rowhammer, and data patterns, and measurement time as a challenge and derive the response from the locations of bit flips encoded by bank, row, and column indices. Using DDR4 systems and a many-sided access pattern [14], they show good uniqueness and reliability, with the most stable fingerprints obtained after approximately 257 s of Rowhammering.

Gerlach et al. [9] evaluated RH-PUFs using Blacksmith on various combinations of four machines, and 10 DIMMs, showing that bit flip patterns vary across runs and hardware, and require ECC to achieve sufficient reliability to be used for PUF. Thereby, they confirm the conclusions of previous studies.

Mechelinck et al. [10] proposed *GlueZilla*, which uses Rowhammer-induced bit flips to bind software execution to specific hardware. By placing control-flow "junction" instructions at known vulnerable memory locations, the program behaves correctly only on the target device; on non-target hardware, where the bit flips are not reliably reproduced, execution follows an alternative, semantically benign path. GlueZilla increases runtime by up to 16 % and code size by 185 %.

Venugopalan et al. [41] propose *FP-Rowhammer*, a DRAM-based device fingerprinting approach.[4] The challenge contains the Rowhammer pattern and

[4] Although the authors do not explicitly frame the FP-Rowhammer method as a PUF, it can be interpreted in PUF terminology, as it conceptually serves the same purpose. This alignment enables consistent comparison with other works in this study.

hammered memory chunks, while the response is the probability distribution of bit flips across these chunks. FP-Rowhammer shows that each memory chunk exhibits stable, high-entropy flip distributions on DDR4, enabling reliable fingerprinting. The authors also investigate the DIMM re-seating scenario, observing reduced reliability but no impact from temperature variations.

Fischer et al. [42] propose a RH-PUF in which hammering patterns and targeted memory regions form the challenge, while statistical summaries of the resulting bitflip patterns serve as the response. Evaluation on DDR4 modules shows high uniqueness and reproducibility, with responses differing even across systems using identical DIMMs, indicating that Rowhammer behavior depends on the interaction between DRAM and the memory controller. This demonstrates that the RH-PUFs are resistant to cloning via DIMM replacement. The authors further employ Mahalanobis distance to account for noise, achieving higher identification accuracy than with the Jaccard index.

Table 2 shows the *summary of the reviewed literature*. Most studies, including ours, evaluate RH-PUFs on desktop systems () with DDR3 [10] or DDR4 [8–10, 41, 42] (), while some research used PandaBoards () with DDR2 [5, 7].

Number of evaluated DIMMs () includes only those affected by Rowhammer. Most studies used fewer than 10 items [5, 7–10]. Fischer et al. [42] analyzed 22 systems, while Venugopalan et al. [41] evaluated up to 98 DIMMs (36 for entropy, 98 for uniqueness and reliability, and 10 for stability). In contrast, our study reports results for 44 systems, exceeding the scale of most prior work.

Regarding the Rowhammer pattern used (), roughly half of the studies employed n-sided hammering [5, 7, 8, 10], whereas the other half, including our work, used multi-sided [41, 42] or non-uniform [9, 41, 42] patterns.

Time characteristics () are often omitted [9, 10, 42] or treated differently: early works used fixed durations [5, 7], while later studies optimize the trade-off between speed and stability [8] or minimize response extraction time [41]. In this work, we focus on security properties and DRAmGON feasibility, and therefore do not optimize runtime, but briefly discuss it in Section 5.3.

The number of measurements per device () varies widely across studies, with most prior work using fewer than 20. In contrast, we collected between 5 and 151 measurements per device, averaging 67.

Most works evaluate PUF properties such uniqueness () and reliability () using Jaccard Indexes and occasionally Mahalanobis distance [42]; some also consider temperature sensitivity () [5, 7, 41] or response entropy () [5, 7, 41]. In contrast, we additionally use Hamming Distance, Dice–Sørensen, and Cosine Similarity, and assess uniformity via Hamming Weight and Shannon entropy, enabling a more comprehensive evaluation of RH-PUF designs.

In *summary*, our work presents a comprehensive experimental study covering a wide range of evaluations, from large-scale system measurements to a multifaceted analysis of PUF properties. In addition, we introduce several new RH-PUF variants and the DRAmGON approach, which together clearly distinguish our work from existing ones.

Table 2. Overview of existing works on Rowhammer-based PUFs.

Authors	**Setup**						**Evaluation**			
			#			#	°C			
Schaller et al. [5] (2017)		2	3	1/2	60/120s	20	40-60	✓	✓	✓
Anagnostopoulus et al. [7] (2018)		2	4	1/2	60/120s	20	0-70	✓	✓	✓
Li et al. [8] (2023)		4	2	22	257s	10	-	✓	✓	-
Gerlach et al. [9] (2023)		4	10	nu	-	3	-	✓	✓	-
Mechelinck et al. [10] (2024)		3/4	$\sum 4$	2	-	2	-	-	-	-
Venugopalan et al. [41] (2025)		4	$\sum 98$	ms/nu	5s	10	15-40	✓	✓	✓
Fischer et al. [42] (2025)		4	22	ms/nu	-	10	-	✓	✓	-
This paper		4	44	nu	-	67	-	✓	✓	✓

Setup: : Type of the platform (- PandaBoard, - PC); : Memory module (DDR(**2**/**3**/**4**)); #: Number of the evaluated modules ($\sum$ - the total number across evaluations with varying sample sizes); : Rowhammering pattern (**n** - n-sided, **ms** - multi-sided, **nu** - non-uniform); : Response exctraction time; #: Number of measurements; *Evaluation:* : Temperature; : Uniqueness; : Reliability; : Entropy. The property is evaluated (✓)/is not evaluated (−).

7 Conclusion and Future Work

In this paper, we present a deep-dive study of Rowhammer-based PUFs (RH-PUFs). First, using the FlippyRAM framework [11], we found that 44 out of 54 systems (81 %) are Rowhammer-susceptible. This confirms that longer run-times increase observed susceptibility and supports prior lower-bound estimates of 27.8 % (see Section 3.3). By focusing on Intel CPUs with DDR4 DRAM, where Blacksmith [12] is effective, we observe substantially higher Rowhammer prevalence than previously reported.

Then, we introduced five new RH-PUF constructions and evaluating them alongside two existing approaches [5, 8] on 44 Rowhammer-susceptible systems (see Section 4). Despite the sparse and unstable bit flips, our results show that carefully designed RH-PUFs can achieve sufficient reliability, uniformity, and uniqueness for device fingerprinting and key generation.

As main contribution, we propose DRAmGON, a novel approach that repurposes Rowhammer-induced bit-flip locations not as an entropy source, but as a medium for persistent key encoding in a PUF-like challenge–response manner (see Section 5), technically enabling persistent data storage in volatile memory. We evaluate DRAmGON on 41 systems and show successful operation on 90 % of them. Furthermore, the results demonstrate that an encoded key can be reconstructed only on the system for which the challenge was generated.

Future work includes improving RH-PUF stability, exploring new constructions, mitigating current limitations of DRAmGON (e.g., applicability, performance, key length, and number of keys), and extending the concept to other PUF types, such as SRAM PUFs.

Acknowledgments. This work was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant number 503876675, the European Union under grant number ROF-SG20-3066-3-2-2, and by the Bavarian State Ministry of Science and the Arts (BayStMWK) under the Bavarian Research Association project "FORDaySec." We also want to thank Jörg Scheidt for the discussions on the statistical verification of DRAmGON.

This preprint has not undergone any post-submission improvements or corrections. The version of Record of this contribution will be published in the proceedings Computer Security – ESORICS 2026

References

[1] Stefan Katzenbeisser and André Schaller. "Physical Unclonable Functions: Sicherheitseigenschaften und Anwendungen". In: *DuD* (2012).

[2] Charles Herder, Meng-Day Yu, Farinaz Koushanfar, and Srinivas Devadas. "Physical Unclonable Functions and Applications: A Tutorial". In: *Proceedings of the IEEE* (2014).

[3] Kristian Monsen and Arnar Birgisson. *Fighting Cookie Theft Using Device-Bound Keys*. `https://blog.chromium.org/2024/04/fighting-cookie-theft-using-device.html`. 2024.

[4] Emiliia Geloczi, Nico Mexis, and Stefan Katzenbeisser. "PUSH for Security: A PUF-Based Protocol to Prevent Session Hijacking". In: *ESORICS*. 2025.

[5] André Schaller, Wenjie Xiong, Nikolaos Athanasios Anagnostopoulos, Muhammad Umair Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. "Intrinsic rowhammer PUFs: Leveraging the rowhammer effect for improved security". In: *HOST*. 2017.

[6] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors". In: *ISCA*. 2014.

[7] Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Yufan Fan, Christian Hatzfeld, André Schaller, Wenjie Xiong, Manishkumar Jain, Muhammad Umair Saleem, Jan Lotichius, Sebastian Gabmeyer, Jakub Szefer, and Stefan Katzenbeisser. "Intrinsic Run-Time Row Hammer PUFs: Leveraging the Row Hammer Effect for Run-Time Cryptography and Improved Security". In: *Cryptography* (2018).

[8] Dawei Li, Di Liu, Yangkun Ren, Ziyi Wang, Yu Sun, Zhenyu Guan, Qianhong Wu, and Jianwei Liu. "FPHammer: A Device Identification Framework based on DRAM Fingerprinting". In: *TrustCom*. 2023.

[9] Lukas Gerlach, Fabian Thomas, Robert Pietsch, and Michael Schwarz. "A Rowhammer Reproduction Study Using the Blacksmith Fuzzer". In: *ESORICS*. 2023.

[10] Ruben Mechelinck, Daniel Dorfmeister, Bernhard Fischer, Stijn Volckaert, and Stefan Brunthaler. "GlueZilla: Efficient and Scalable Software to Hardware Binding using Rowhammer". In: *DIMVA*. 2024.

[11] Martin Heckel, Nima Sayadi, Jonas Juffinger, Carina Fiedler, Daniel Gruss, and Florian Adamsky. "FlippyR.AM: A Large-Scale Study of Rowhammer Prevalence". In: *NDSS*. 2026.

[12] Patrick Jattke, Victor van der Veen, Pietro Frigo, Stijn Gunter, and Kaveh Razavi. "BLACKSMITH: Rowhammering in the Frequency Domain". In: *S&P*. 2021.
[13] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms". In: *CCS*. 2016.
[14] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. "TRRespass: Exploiting the Many Sides of Target Row Refresh". In: *S&P*. 2020.
[15] Haocong Luo, Ataberk Olgun, Abdullah Giray Yağlıkçı, Yahya Can Tuğrul, Steve Rhyner, Meryem Banu Cavlak, Joël Lindegger, Mohammad Sadrosadati, and Onur Mutlu. "RowPress: Amplifying Read Disturbance in Modern DRAM Chips". In: *ISCA*. 2023.
[16] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript". In: *DIMVA*. 2016.
[17] Finn de Ridder, Pietro Frigo, Emanuele Vannacci, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. "SMASH: Synchronized Many-sided Rowhammer Attacks From JavaScript". In: *USENIX Security*. 2021.
[18] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. "Another Flip in the Wall of Rowhammer Defenses". In: *S&P*. 2018.
[19] Andreas Kogler, Jonas Juffinger, Salman Qazi, Yoongu Kim, Moritz Lipp, Nicolas Boichat, Eric Shiu, Mattias Nissler, and Daniel Gruss. "Half-Double: Hammering From the Next Row Over". In: *USENIX Security*. 2022.
[20] Patrick Jattke, Max Wipfli, Flavien Solt, Michele Marazzi, Matej Bölcskei, and Kaveh Razavi. "ZenHammer: Rowhammer Attacks on AMD Zen-based Platforms". In: *USENIX Security*. 2024.
[21] Diego Meyer, Patrick Jattke, Michele Marazzi, Salman Qazi, Daniel Moghimi, and Kaveh Razavi. "Phoenix: Rowhammer Attacks on DDR5 with Self-Correcting Synchronization". In: *S&P*. 2026.
[22] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation". In: *USENIX Security*. 2016.
[23] Andrei Tatar, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. "Defeating Software Mitigations Against Rowhammer: A Surgical Precision Hammer". In: *RAID*. 2018.
[24] Sangwoo Ji, Youngjoo Ko, Saeyoung Oh, and Jong Kim. "Pinpoint Rowhammer: Suppressing Unwanted Bit Flips on Rowhammer Attacks". In: *AsiaCCS*. 2019.
[25] Andrew Kwong, Daniel Genkin, Daniel Gruss, and Yuval Yarom. "RAMBleed: Reading Bits in Memory Without Accessing Them". In: *S&P*. 2020.

[26] Ingab Kang, Walter Wang, Jason Kim, Stephan van Schaik, Youssef Tobah, Daniel Genkin, Andrew Kwong, and Yuval Yarom. "SledgeHammer: Amplifying Rowhammer via Bank-level Parallelism". In: *USENIX Security*. 2024.
[27] Martin Heckel and Florian Adamsky. "Flipper: Rowhammer on Steroids". In: *uASC*. 2025.
[28] Zhenkai Zhang, Zihao Zhan, Daniel Balasubramanian, Xenofon Koutsoukos, and Gabor Karsai. "Triggering Rowhammer Hardware Faults on ARM: A Revisit". In: *ASHES Workshop*. 2018.
[29] Misiker Tadesse Aga, Zelalem Birhanu Aweke, and Todd Austin. "When good protections go bad: Exploiting anti-DoS measures to accelerate Rowhammer attacks". In: *HOST*. 2017.
[30] Jonas Juffinger, Sudheendra Raghav Neela, Martin Heckel, Lukas Schwarz, Florian Adamsky, and Daniel Gruss. "Presshammer: Rowhammer and Rowpress without Physical Address Information". In: *DIMVA*. 2024.
[31] Michele Marazzi and Kaveh Razavi. "RISC-H: Rowhammer Attacks on RISC-V". In: *DRAMSec Workshop*. 2024.
[32] Ferdinand Brasser, Lucas Davi, David Gens, Christopher Liebchen, and Ahmad-Reza Sadeghi. "CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory". In: *USENIX Security*. 2017.
[33] Radhesh Krishnan Konoth, Marco Oliverio, Andrei Tatar, Dennis Andriesse, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. "ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks". In: *USENIX OSDI*. 2018.
[34] Ataberk Olgun, Yahya Can Tugrul, Nisa Bostanci, Ismail Emir Yuksel, Haocong Luo, Steve Rhyner, Abdullah Giray Yaglikci, Geraldo F Oliveira, and Onur Mutlu. "ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation". In: *USENIX Security*. 2024.
[35] Zelalem Birhanu Aweke, Salessawi Ferede Yitbarek, Rui Qiao, Reetuparna Das, Matthew Hicks, Yossi Oren, and Todd Austin. "ANVIL: Software-based protection against next-generation Rowhammer attacks". In: (2016).
[36] Jonas Juffinger, Lukas Lamster, Andreas Kogler, Maria Eichlseder, Moritz Lipp, and Daniel Gruss. "CSI: Rowhammer - Cryptographic Security and Integrity against Rowhammer". In: *S&P*. 2023.
[37] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. "Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks". In: *S&P*. 2019.
[38] Nureddin Kamadan, Walter Wang, Stephan van Schaik, Christina Garman, Daniel Genkin, and Yuval Yarom. "ECC.Fail: Mounting Rowhammer Attacks on DDR4 Servers with ECC Memory". In: *USENIX Security*. 2025.
[39] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data". In: *SIAM journal on computing* (2008).

[40] Roel Maes. “Physically Unclonable Functions: Properties”. In: *Physically Unclonable Functions: Constructions, Properties and Applications*. 2013.

[41] Hari Venugopalan, Kaustav Goswami, Zainul Abi Din, Jason Lowe-Power, Samuel T. King, and Zubair Shafiq. “FP-Rowhammer: DRAM-Based Device Fingerprinting”. In: *AsiaCCS*. 2025.

[42] Bernhard Fischer, Daniel Dorfmeister, Harald Lampesberger, and Eckehard Hermann. “Leveraging Rowhammer for Physically Unique and Nontamperable Device Identification”. In: *International Conference on Industry 4.0 and Smart Manufacturing*. 2025.

[43] Martin Heckel and Florian Adamsky. “Reverse-Engineering Bank Addressing Functions on AMD CPUs”. In: *DRAMSec*. 2023.

[44] Lee R Dice. “Measures of the amount of ecologic association between species”. In: *Ecology* (1945).

[45] Alan H Cheetham and Joseph E Hazel. “Binary (presence-absence) similarity coefficients”. In: *Journal of Paleontology* (1969).

[46] Claude. E. Shannon. “A mathematical theory of communication”. In: *The Bell System Technical Journal* (1948).

[47] Elaine B. Barker and Allen L. Roginsky. *Transitioning the Use of Cryptographic Algorithms and Key Lengths*. Tech. rep. NIST SP 800-131A Rev. 3. National Institute of Standards and Technology (NIST), 2024.

[48] Alexis Hocquenghem. “Codes correcteurs d’erreurs”. In: *Chiffers* (1959).

[49] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. “On a class of error correcting binary group codes”. In: *Information and control* (1960).

[50] Fatemeh Tehranipoor, Nima Karimian, Wei Yan, and John A Chandy. “Investigation of DRAM PUFs reliability under device accelerated aging effects”. In: *ISCAS*. 2017.

[51] Md Sadik Awal and Md Tauhidur Rahman. “Exploring the Correlation Between DRAM Latencies and Rowhammer Attacks”. In: *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2024.

A Schematical Illustration of RH-PUF Constructions

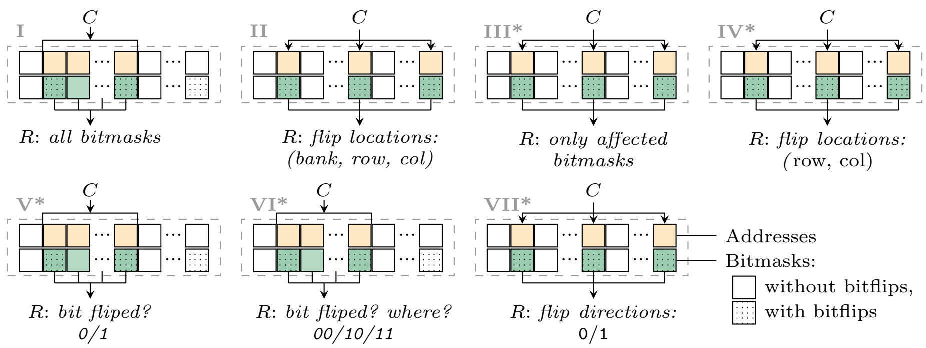

Fig. 5. RH-PUF constructions: (I) Bitmask Snapshot, Affected (II) Addresses/(III*) Bitmasks/(IV*) Addresses Short, Flip (V*) Existence/(VI*) Combo /(VII*) Direction.