Centauri: Fingerprinting the Rowhammered Sky

Anonymous Submission

I. Introduction

Modern browsers and operating systems restrict access to stateful identifiers [42], [18], [3] to conform to privacy regulations [54], [14]. In response to these restrictions, trackers look for other alternatives to track users [44] across websites/apps, with device fingerprinting [50] being rapidly adopted as a viable alternative. With device fingerprinting, trackers run scripts on user devices with the same inputs and use differences in the outputs to identify and track devices.

In order to function as stateful identifiers, a good fingerprint should have high entropy (or discrimination power) and high stability. High entropy ensures that different devices are not identified as the same device and high stability ensures that the same device can be repeatedly identified over long periods of time. Traditional fingerprinting techniques rely on the heterogeneity in hardware and software configurations to extract a fingerprint. They build fingerprints by aggregating attributes such as the browser's User Agent, screen resolution etc [13], [56] or by making inferences based on the differences in the implementation of certain algorithms [41], [8] on different hardware and software platforms.

Normalizing attributes like the User Agent [57], [43], [9] and APIs like HTML5 Canvas to return the same value on all devices [6], [40] is a commonly employed defense against fingerprinting. However, in most cases, such normalization also leads to some loss in utility. Using identical hardware and software on all devices (such as those in universities or companies) as opposed to normalizing certain attributes or APIs is a potential alternative to combat fingerprinting without compromising utility. Indeed, we observe that FingerprintJS [15], a widely deployed fingerprinting library, computes the same fingerprint hash on all 9 machines in our lab that have the same configuration (same browser, same operating system, same CPU architecture etc). We thus ask the question if fingerprints with high entropy and high stability can be obtained with this extreme form of normalization where all hardware and software configurations are identical.

One way to extract fingerprints even with identical hardware and software configurations would be to infer differences that arise as a result of process variation in the manufacture of hardware. Bit flips triggered by the Rowhammer vulnerability [27] present a potential vector to infer these differences. Figure 1 visualizes the distribution of bit flips produced by triggering the Rowhammer vulnerability at the same locations on two identical DRAM modules (also called dual in-line memory modules or DIMMs) at different points in time. From the visualization, we see that the distribution of bit flips in the same DIMM is stable while different across the 2 DIMMs.

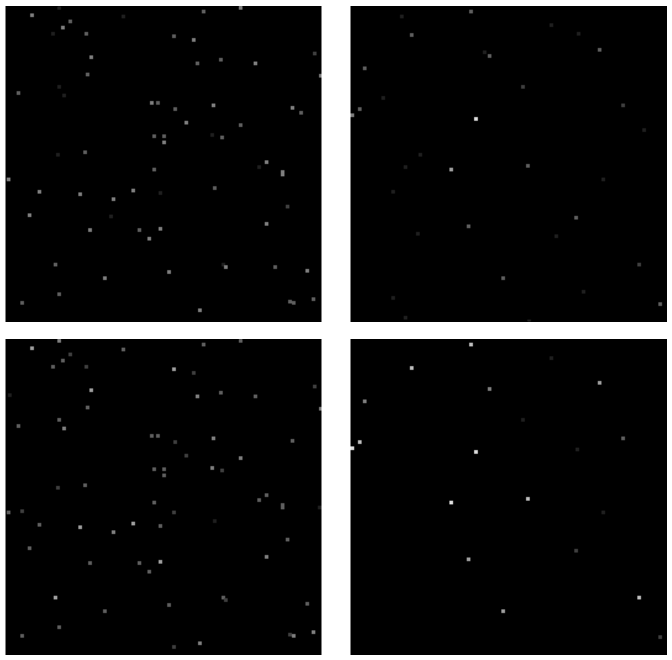
Fig. 1: Visualization of the distribution of bit flips triggered by the Rowhammer [27] vulnerability, with a brighter spot representing a higher probability for a bit to flip. The top row shows the distribution when triggering the vulnerability at the same locations on two identical DRAM modules. The bottom row shows the distribution at the same locations on the same modules when triggering the vulnerability at a later point in time. The visualizations make it clear that the distribution is stable on the same module, while remaining different across different modules.

In this paper, we present Centauri, a memory (DRAM) based fingerprinting technique with high stability and entropy even among homogeneous devices. Prior research exploring the fingerprinting capabilities of DRAM using Rowhammer [51], [32] do not practically demonstrate how a tracker running an unprivileged program on a victim's device can extract a fingerprint on DDR4 DIMMs, which are curretnly the most commonly deployed type of memory.

Using Rowhammer to extract a fingerprint without requiring administrative privileges is challenging for three main reasons. First, a tracker running an unprivileged program on a victim's device has to depend on the operating system to access memory. The abstractions provided by the operating system to do so hide information about where memory was physically allocated on the DRAM. Thus, trackers cannot merely track the locations of bit flips to extract a fingerprint. Second, DDR4 DIMMs implement Target Row Refresh or TRR [33] to mitigate Rowhammer. While existing research has shown ways to bypass TRR by discovering many-sided [16] or non-uniform [23] hammering patterns, they do not focus on the portability of these patterns. Studying the portability of hammering patterns is important for a tracker since they cannot execute their programs for long durations on victim devices to discover such patterns. Third, we find that that while non-uniform hammering patterns are effective at producing bit flips, the resultant bit flips themselves are non-deterministic, i.e., hammering with the same non-uniform parameters at the same physical locations don't always lead to bit flips in the same locations. Trackers would have to account for this non-determinism while extracting a fingerprint.

To overcome these challenges, we first perform a measurement study of the distribution of bit flips across different DDR4 DIMMs. In our study, we observe that non-uniform hammering patterns that produce bit flips on a particular module also tend to produce bit flips on other modules from the same manufacturer. We also observe that the bit flips within contiguous regions of memory (2 MB regions) on each DIMM can be characterized by unique probability distributions. We leverage these observations to devise a sampling based strategy to extract a fingerprint without requiring administrative privileges. On a test set of XX DIMMs, across 7 sets of identical DIMMs Centauri has 100% fingerprint accuracy. The fingerprints extracted by Centauri remain stable over XX months and with changes to the CPU frequency.

Our contributions include:

- A measurement study of the distribution of bit flips produced by non-uniform hammering patterns on a wide range of DRAM DIMMs.
- A sampling based strategy to fingerprint DIMMs based on differences in bit flip distributions.
- Stability analysis of bit flips over time and with variations to CPU frequency.
- A practical unprivileged implementation of our proposed strategy.

II. Background

In this section, we introduce the basics of a DRAM device and its susceptibility to the infamous rowhammer (RH) vulnerability. We follow it up with a general discussion on fingerprinting before concluding the potential of RH as a fingerprint.

A. DRAM Organization

DRAM technology has spawned several generations including DDRx [35], [36], [38], [24], GDDRx [21], [39], LPDDRx [34], DDRxL [37] etc. Each physical DRAM Dual Inline Memory Module or DRAM DIMM is installed on a DRAM *channel* on the motherboard. Channels permit issuing concurrent memory requests to multiple DIMMs. DIMM contain multiple logical structures called *banks* on one or both of their *ranks*. A bank is a two-dimensional array of cells organized into *rows* and *columns*. Each cell contains a capacitor and an access transistor, with the capacitor's charged state representing a single bit. Organizing cells into banks enables faster memory access since memory addresses that map to different banks can be accessed in parallel. Independent dies are packaged into multiple physical entities called *chips*. The banks on each rank are uniformly distributed across these chips as shown in Figure 2. Suppose we have a `x8` width DIMM, which has 16 banks. Each memory read or write request fetches 64 $\times$ 16 bytes of data from all the chips, and then using an appropriate mask based on the bank, the correct 64 bytes are selected. In this case, there are 2^{10} columns, where each column is 8 bits long. Each bit is represented via a capacitor. In total, for a `x8` DIMM row of a single bank, there are 65,536 capacitors.

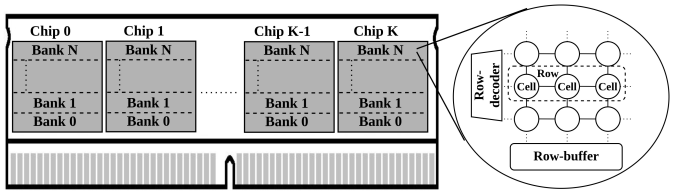

Fig. 2: Figure shows a single rank of a DRAM DIMM. Each rank contains multiple logical structures called banks that are interspersed across multiple physical structures called chips.

B. Basic DRAM Commands

The memory controller receives either memory requests corresponding to an address or a command. The memory controller is responsible for handling the following commands:

- `ACTIVATE (ACT)`: Opens or activates the specified row by loading it into the row-buffer. This has to be done before reading or writing into any row.
- `READ/WRITE (RD/WR)`: Accesses the specified cell within the row-buffer to perform a RD/WR operation.
- `PRECHARGE (PRE)`: Closes or deactivates a row and prepares the row-buffer to load another row by restoring the previously stored values.
- `REFRESH (REFI)`: Since DRAM technology is based on capacitors, therefore it drains charge over time. Hence, the DRAM has to be refreshed periodically, which is achieved via REFI commands. Every DRAM capacitor gets refreshed at least once every 32 ms or 64 ms (`tREFW`) depending upon the operating temperature[1].

C. The Rowhammer Vulnerability

Scaling of DRAM devices over the generations has resulted in the deviation of its nominal parameters from its theoretical counterpart due to variations [60], [5], [?], [31]. Modern DIMMs are susceptible to memory corruption as a result of

[1]`tREFW` usually changes to 32 ms at higher temperatures. However, it has also been reported `tREFW` changes to 32 ms as a rowhammer prevention technique [16], [20].

electrical interference among the cells []. One such technique to leverage bit flips without accessing the row is the rowhammer (RH) exploit [4], [27]. It causes memory corruption by repeatedly accessing two addresses that map to two different rows in a single bank[2]. This leads to the repeated activation and deactivation of the two rows back-and-forth between the row-buffer and the DRAM bank array. The resulting electromagnetic interference between the accessed rows (referred to as *aggressor rows* henceforth) and their neighboring rows (referred to as *victim rows* henceforth) accelerates the rate at which the capacitors in the victim rows lose their charge. Once capacitors have lost a sufficient amount of charge, refreshing the DRAM does not restore their value, resulting in memory corruption in the form of bit flips.

There has been works done in the past to detect and mitigate RH attacks [27], [53], [52], [30], [46], [58]. However, DRAM manufacturers have chosen to implement *target row refresh* (TRR) added the standard mitigation mechanism in DDR4 devices as it is the most practical RH defence mechanism. [49], [38]. TRR has proven to be ineffective, as prior research have bypassed TRR [16], [20], [23]. The idea of TRR is to sample aggressors and send targeted refresh commands to the victim rows when the DRAM device is locked for a normal refresh operation (`REFI`).

D. Digital Fingerprinting

In computing terms, fingerprinting is a technique which maps arbitrarily large data items to a shorter bit string that can uniquely identify the original data item or source [7]. The most common examples of fingerprinting function are high-performance hash functions [17]. In recent times, browser fingerprinting techniques [56], [41], [1], [13] have gained popularity as it allows to track users without their consent [56]. One of the recurring challenge of fingerprinting is to handle changes in the user's configuration [56]. In such scenario, two different fingerprint bit strings map to the original data item. This is known as the *linking* problem. Several works have been done in the past to improve linking [56]. This contributes largely to the pool of errors as the number of false negatives and false positives increases.

Fingerprinting can be broadly classified into two primary categories: software-based and hardware-based fingerprinting. Software-based fingerprinting is fairly common in the browser space [41], [13]. The reason is the ease of implementation and deployment of such fingerprinting method. However, the downside of such a fingerprint is that the software vulnerability that it exploits can be either patched or obfuscated easily []. Hardware-based fingerprinting techniques [29], [10], [11], [59], [50] on the other hand has a higher stability[3] []. Hardware exploits cannot be patched easily. Often there is a hefty cost associated [28], [38], [49]. The *meta* exploit here is the answer to the question: *How often do you upgrade your phone or your PC?* This makes hardware-based fingerprinting stable, reliable and much more dangerous.

E. RH as a Fingerprinting Technique

Variations on DRAM devices have a huge potential for fingerprinting. The rate at which a capacitor loses its charge depends on its physical properties [47], [22]. DRAM capacitors on different DIMMs would have different values for these parameters as a result of process variation in how they were manufactured. Thus, the DRAM capacitors that lose their charge (equivalently, the bits that flip) would also depend on the DIMM being subjected to the RH exploit. When running the exploit with identical parameters (accessing the same addresses at the same rate for the same duration) under comparable environmental conditions on different DIMMs, differences in the bit flip behavior have to be attributed to differences in the physical properties of the DIMMs. In this paper, we explore the potential of using bit flip patterns as fingerprints to uniquely identify DIMMs.

[2]Or a single row if the row-buffer follows a closed-page policy, where after each `ACT`, the row is immediately closed.

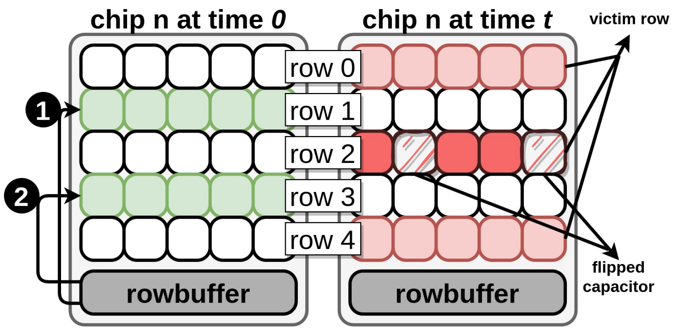

Fig. 3: Figure shows a single chip, where an attacker is performing Actions ❶ and ❷ which activates rows 1 and 3 repeatedly. This makes rows 0, 2 and 4 victims, with 2 being the most probable. At some time t, the aggressor rows inflict bitflips on row 2, shown by shaded red blocks.

III. Measuring the Entropy of Rowhammer Bit Flips

In this section, we first calculate a theoretical upper bound on the entropy that can be obtained from the bits that flip across multiple chunks of contiguous 2 MB of memory. We then measure the actual entropy on a set of $2,406$ such chunks across 15 DIMMs. We focus on 2 MB since this is the largest contiguous chunk of memory that we can reliably obtain from the OS without requiring administrative privileges (using Transparent Huge Pages or THP [25]). From our measurement, we observe that the bits that flip across such chunks are highly unique and use this observation to drive the design of our fingerprint methodology.

A. Theoretical analysis

If we consider that there are approximately 10 billion DIMMs on the planet, with each DIMM having approximately

[3]*Stability* in the context of fingerprinting means the time period until which one can deterministically extract a fingerprint from a device.

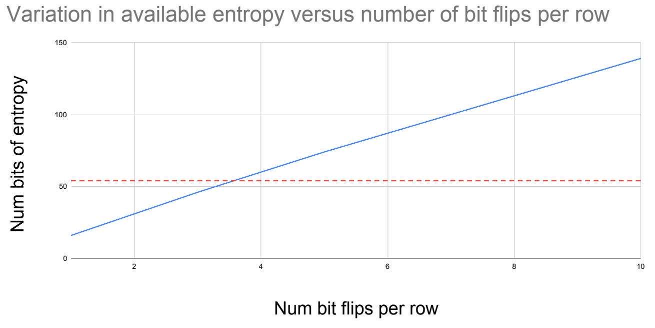

Fig. 4: This plot shows that if exactly 4 or more bits flip per row, we can use the index of the flipped bit within the row (among $65,536$ possibilities) to obtain an entropy of 54 bits, which is sufficient to represent every row on every DIMM in existence.

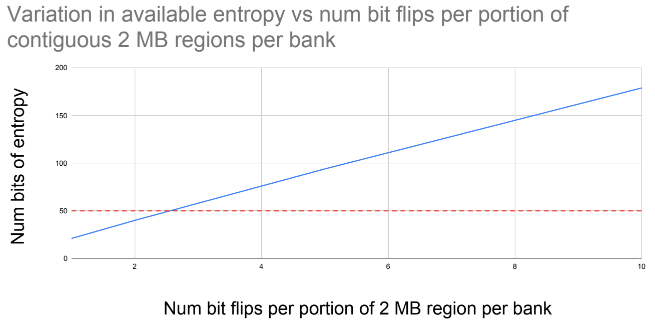

Fig. 5: This plot shows that if exactly 3 or more bits flip within each 2 MB chunk of contiguous memory contained within a bank, we can use the index of the flipped bit within the chunk (among $1,048,576$ possibilities) to obtain an entropy of 50 bits, which is sufficient to represent all such chunks on all DIMMs in existence.

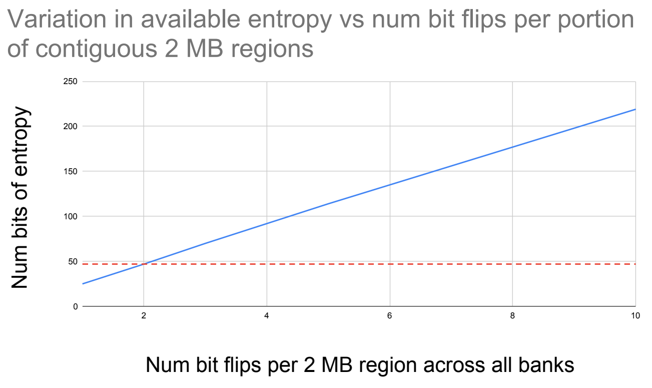

Fig. 6: This plot shows that if exactly 2 or more bits flip within each 2 MB chunk of contiguous memory, we can use the index of the flipped bit within the chunk (among $16,777,216$ possibe indices) to obtain an entropy of 47 bits, which is sufficient to represent all such chunks on all DIMMs in existence.

10 banks, and each bank having approximately $100,000$ rows, we will have a total of 10^{16} possible rows. We will need approximately 54 bits of entropy to represent all these rows (since $2^{54} > 10^{16}$). Since rows within DRAM DIMMs are a finer granularity than the number of DIMMs (and correspondingly the number of devices), obtaining an entropy of 54 bits would be sufficient to represent all devices on the planet.

As discussed in § II, every row of a DRAM chip contains $65,536$ capacitors. If process variation in the manufacture of chips results in exactly one bit flip per row (i.e. only one capacitor loses its charge) when subjected to Rowhammer, we can use the index of the flipped bit within each row to at most identify $65,536$ rows, equivalent of 16 bits of entropy. Thus, if exactly one bit flips per row, using the index of the flipped bit within the row does not have enough entropy to represent all possible rows across all DIMMs. Figure 4, shows the amount of entropy available to represent all rows with with varying number of bit flips per row, with the dashed line showing the required entropy. From the figure, we see that if 4 bits flip per row, we can represent all possible rows since we get 60 bits of entropy ($log_2\binom{65,536}{4}$).

The analysis present so far limit us to only observe the bit flips within a single row. However, with a contiguous chunk of 2 MB of memory, we can access multiple consecutive rows from each bank. For example, in case of single rank DIMMs having a width of 8 bits, 2 MB of contiguous memory corresponds to 16 consecutive rows (or $1,048,576$ capacitors) per bank. In this case, we will need an entropy of 50 bits to represent 10^{15} total such regions across all possible DIMMs. In this case, if 3 bits flip per region, we get 58 bits of entropy ($log_2\binom{1,048,576}{3}$) to represent all such regions. If we take the capacitors across all banks in a 2 MB contiguous chunk, we see that 2 flips per chunk is sufficient to obtain an entropy of 47 bits ($log_2\binom{16,777,216}{2}$), which is sufficient to represent 10^{14} such regions across all DIMMs. We show similar plots showing the variation in entropy with varying number of bit flips produced in these configurations in Figure 5 and Figure 6.

In summary, our analysis indicates that the distribution of bit flips triggered by Rowhammer in individual 2 MB contiguous chunks of memory is potentially unique as long as the chunks can produce at least 4 bit flips. The theoretical analysis assumed that all regions produce bit flips and that the distribution of bit flips was uniform. We now perform experiments to measure the actual entropy across such regions across multiple DIMMs.

B. Measurement study

Existing research on Rowhammer has primarily focused on developing techniques to trigger bit flips [27], [19], [55], [16], [12], [23] in memory, but not on analyzing the distribution of the resultant bit flips. In this section, we present the first measurement study of the distribution of bit flips on DDR4

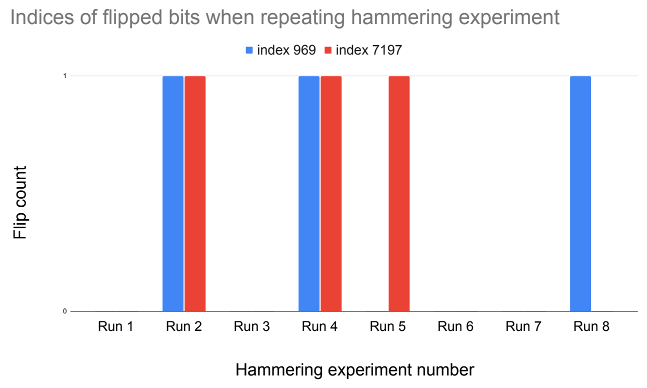

Fig. 7: This plots shows the index of bits that flipped in a contiguous chunk of 2 MB of memory contained within an arbitrarily chosen bank on a single DIMM across multiple experiments. In each experiment, we hammer access addresses within the region $5,000,000$ times and restore the original data between experiments. We see that while the bits that flip are not exact, they are confined to the same set of indices.

DIMMs. Concretely, we first validate our theoretical analysis by measuring the entropy of the distribution of bit flips within a given bank across multiple 2 MB chunks of memory across DIMMs. As mentioned in § I, we find that bit flips are not deterministic and, as a result, merely measuring the entropy of the distribution of bit flips is insufficient to extract a reliable fingerprint. Thus, we also measure the persistence of the distribution of bit flips across repeated measurements to the same chunks across DIMMs.

1) Test Bed: Our test bed for this measurement consists of 15 single rank, 8 width DIMMs from SK Hynix. We discover a non-uniform hammering pattern that can trigger bit flips by fuzzing one of these DIMMs (using Blacksmith's fuzzer [23]). We find that the discovered pattern is able to produce bit flips on all 15 DIMMs. We conduct our experiments by using the discovered pattern to trigger bit flips within a particular bank of 2406 different 2 MB regions across these 15 DIMMs.

2) Methodology: We conduct our experiments in a controlled setting with administrative privileges that allow us to allocate upto 1 GB of contiguous memory. The controlled setting gives us control over where we perform our hammering since we observe that the allocation of huge pages in physical memory rarely changes in Linux (verified using pagemap [26]). We confine our hammering to randomly chosen 2 MB regions within the allocated huge page that lie within an arbitrarily chosen bank. To perform our hammering, we modify the hammering pattern determined by Blacksmith's fuzzer to only trigger bit flips within the randomly chosen 2 MB regions. We provide more details on the hammering parameters in Appendix **??**.

3) Results: Across all 15 DIMMs, we hammered a total of $2,406$ regions. 99.54% of these regions $(2,395$ regions) produced at least one bit flip. Among these regions, the number

Measurement	Value
Percentage of regions showing bit flips	99.54%
Minimum number of bit flips per region	1 flip
Maximum number of bit flips per region	15,191 flips
Average number of bit flips per region	1,488 flips
Measured entropy across 2,395 regions	12 bits
Normalized entropy	1

TABLE I: Summary of the measured distribution of bit flips across 15 SK Hynix DIMMs

of bit flips ranged from 1 bit flip to $15,191$ bit flips at an average of $1,488$ bit flips per region. Across the regions that produced bit flips, we record an entropy of 12 bits for the triggered bit flips. We calculated the entropy in terms of the number of regions that had the same bit set of bit flips as another region. Crucially, we highlight that in our test bed, 12 bits of entropy correspond to a normalized entropy of 1 [2] which shows that each region has a unique set of bit flips. We summarize these findings in Table I.

We use the fact that the bit flips in every region in our experiment was unique to estimate the entropy that we can expect from all such possible regions. Extrapolating our results, based on the average of $1,488$ bit flips per region yields over $16,000$ bits of entropy, which is significantly higher than the 50 bits needed to represent such regions in all the DIMMs present on the planet.

To use the bit flips produced by Rowhammer as a fingerprint, ensuring that they have high entropy on different regions is not sufficient unless they also have high persistence within the same region. In our study, we notice that reinitializing regions with the same data and hammering them again does not guarantee that the same bit will flip again. In other words, we observe that the bits that flip within a given region are not deterministic. Thus, using metrics like Jaccard index (as proposed by existing research [51], [32]) to match fingerprints will likely lead to mistakes, especially in those regions where bit flips are sparse. For example, Figure 7 shows the bit flips observed when hammering the same region 8 times on a particular DIMM (while restoring the data written to the region before beginning a fresh round of hammering). From the Figure, we see that the same set of bits flip only two times out of 8 attempts. Thus, unless there is an exact match, the highest possible Jaccard index that can be obtained when matching between these runs is 0.5.

From the same Figure, we observe that while bit flips are not deterministic, the set of bits that flip are always among the same set of indices. Leveraging this observation, we calculate a probability distribution for the bit flips in each region and match similarity of distributions to measure persistence. To extract a probability distribution from a given region, we hammer the region multiple times and use the count of flips at different indices across all our hammering attempts. Then, we use Jensen-Shannon (JS) divergence [45] to compute similarity of distributions. In Figure 8, we compare the distributions for 3 randomly chosen regions across 3 independent runs. From

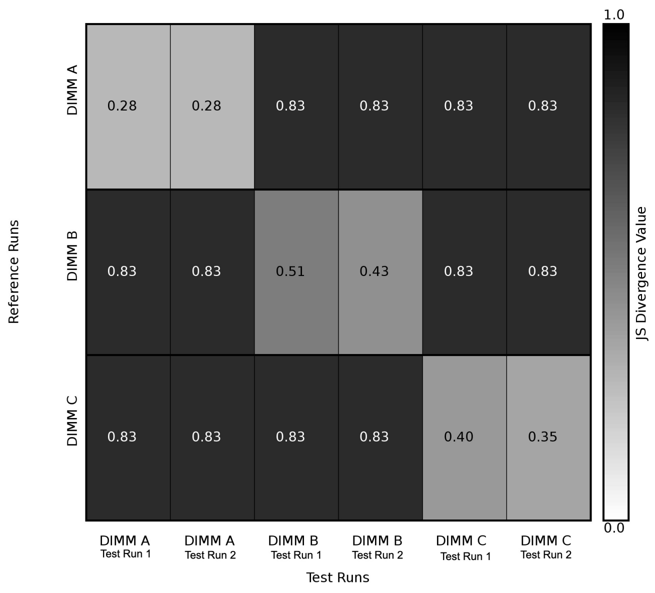

Fig. 8: Visualization of the persistence of bit flips within a given 2 MB chunk of memory containined within a single bank.

the figure, we see that the bit flips within each region, while not deterministic, relatively happen at the same indices when compared to other regions. Thus, from our measurement of the distribution of bit flips, we see that contiguous 2 MB chunks of memory within a given bank has high entropy and high persistence. In the next section, we show how we use this observation to extract a reliable fingerprint.

IV. Threat Model [WiP]

This section describes the threat model for Centauri. Centauri has two distinct in its execution. These are *(a)* training or templating phase, and, *(b)* testing or evaluation phase. The former is performed on the attacker's side. It is essentially the attacker templating hammering patterns on a wide set of DIMMs. For this purpose, the attacker can use any rowhammering software. Since templating is performed on the attacker's own machine, he or she has full authority of the system. Once the attacker has enough patterns covering different vendors and configurations, he or she is ready to deploy the rowhammering software onto victim's machine, and replay different patterns until bits flip.

The evaluation phase on the other hand, is more challenging as the attacker has no control over the rowhammering phase. In our threat model, we assume that the attacker maps the aggressor rows in a double-sided attack pattern using 2 MB huge pages. 2 MB pages does not require administrative privileges on most popular Linux distributions []. In addition, it gives the attacker to manipulate the first 21 bits of the address, which is enough to extract the bank bits (see Appendix **??**). Once the mapping set, the attacker executes each pattern that he or she has until there are bitflips.

V. Centauri

Centauri broadly consists of three phases to fingerprint user devices, namely a templating phase, a hammering phase, and a matching phase. In this section, we provide a detailed account of each of these phases.

A. *Templating phase*

In the templating phase, attackers conduct experiments on their own devices to discover hammering patterns that can overcome Rowhammer mitigations and trigger bit flips. Concretely, in our experiments with DDR4 DIMMs, attackers use the Blacksmith [23] fuzzer to determine non-uniform hammering patterns that can evade Target Row Refresh (TRR). In our experiments, we find that DRAM manufacturers tend to reuse their TRR implementations on the DIMMs they manufacture. Thus, attackers maintain a record of patterns discovered on their own DIMMs and reuse these patterns to trigger bit flips on victim DIMMs.

Concretely, we found patterns on DIMMs from all 3 major DRAM manufacturers that consistently trigger bit flips on other DIMMs from the same manufacturer. We found patterns discovered on SK Hynix to generalize the most in terms of consistently producing bit flips on the largest number of DIMMs from the same manufacturer. 5 patterns discovered on DIMMs from SK Hynix consistently produced bit flips on over 80 DIMMs in our test bed. Similarly, 4 patterns discovered on DIMMs from Samsung consistently triggered bit flips on over 10 DIMMs from Samsung. Interestingly, unlike SK Hynix, with Samsung we find that the choice of the bank where we perform our hammering is important to trigger bit flips. In other words, hammering with the same pattern on some banks of a given DIMM from Samsung produces bit flips, while hammering on some other banks does not. We suspect that Samsung uses a per-bank implementation of TRR that is responsible for this observation. With Micron, we found it difficult to discover patterns that produced bit flips and thus, obtained patterns discovered by the authors of Blacksmith. Their patterns consistently triggered bit flips on at least 2 Micron DIMMs in our test bed.

B. *Hammering phase*

In the hammering phase, the attacker runs some unprivileged code on a victim's device. Victims either install and run an app developed by the attacker on their devices or visit a website maintained by the attacker on their browser. In our experiments, we focus on the former and discuss the latter in Section XX.

The attacker's goal is to trigger bit flips in the victim's device and use the distirbution of bit flips to extract a fingerprint. Since the attacker does not have knowledge of the type of DIMMs (or their corresponding TRR implementations) on the victim's device, they rely on the patterns discovered in the templating phase to trigger bit flips. Blacksmith's fuzzer

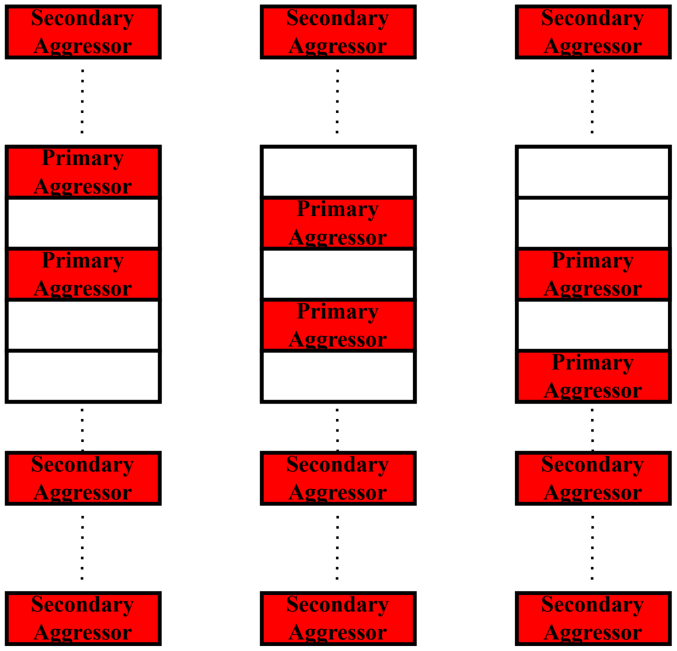

Fig. 9: Visualization of a hammering sweep performed within one bank of a transparent huge page. The central rectangular blocks in the visualization represent rows within the huge page. We map primary aggressors in the discovered patterns to rows within the transparent huge page and secondary aggressors to random rows within the same bank. The hammering sweep involves sequentially hammering all pairs of double-sided aggressors within the bank of the huge page and scanning the other rows for bit flips.

(used in the templating phase) discovers a hammering pattern containing n aggressors, a specific phase at which to access each aggressor, an intensity at which to continuously repeat accesses to each aggressor and a frequency at which to iterate over the pattern [23]. The aggressors, phase, intensity and frequency are determined such that some aggressors serve to engage with TRR (we refer to these as secondary aggressors) and the others serve to trigger bit flips (we refer to these as primary aggressors). To consistently produce bit flips, the primary aggressors are a double-sided aggressor pair, and the secondary aggressors are other addresses within the same bank as the primary aggressors.

To execute the discovered patterns, the attacker's code allocates transparent huge pages to obtain contiguous chunks of 2 MB of memory. Such allocations allow the attacker to access all addresses that can be reached by modifying the lower 21 bits of the address of the start of each huge page. This, in turn allows the attacker to pick double-sided aggressor pairs since such allocations typically provide access to contiguous rows across multiple banks of a DIMM. For example, in case of a victim having a singular single rank DIMM with a width of 8 bits, such an allocation gives the attacker access to 16 contiguous rows within each of the 16 banks on the DIMM. In the remainder of this section, we discuss the hammering phase in context of a single DIMM present on the victim's device. We discuss ways to extend our approach to victim devices having multiple DIMMs across multiple channels in Section YY. To trigger bit flips, the attacker first chooses a particular bank within a huge page. The attacker can do this since most bits in the physical address that represent the bank are contained with the lower 21 bits in most CPU architectures [48], [12]. Then, the attacker maps the primary aggressors in the discovered patterns to double-sided aggressors within the chosen bank of the huge page and secondary aggressors to random addresses within the chosen bank (possibly outside the huge page). While the attacker cannot know the exact row of a given address from the lower 21 bits, they can know the relative position of rows with respect to the row corresponding to the start address of the huge page. This allows them to choose different pairs of rows as their primary aggressors.

Concretely, the attacker sequentially considers all pairs of double-sided aggressors within the bank as primary aggressors and executes the discovered pattern. Before shifting the primary aggressors, the attacker records the bits that flipped and restores the original data to the rows. We refer to this operation of hammering all possible double-sided aggressors within the allocated region as primary aggressors as a hammering sweep. Figure shows a visual representation of the hammering sweep. To account for non-determinism in bit flips, the attacker repeats the hammering sweep multiple times and records the number of times each bit flipped in the allocated region. Upon completion, the attacker's code sends the recorded information about the bits that flipped and their counts back to their server to match fingerprints. Listing summarizes the entire hammering procedure.

C. *Matching phase*

With the observation from §III that the distribution of bit flips within a bank of contiguous 2 MB regions is highly unique and persistent, attackers can match these distributions to fingerprint users. However, attackers cannot guarantee access the same 2 MB regions on a victim's device, since memory allocation is handled by the OS. Thus, if an attacker obtains two different 2 MB chunks of memory on a victim's device during two different sessions and compared their distributions, they would incorrectly conclude that different devices were used during these sessions.

1) Hammering multiple regions: One way to overcome this challenge would be to have attackers hammer multiple 2 MB chunks during each session. For example, suppose the victim's device has 1 GB of memory which corresponds to 512 different 2 MB chunks of memory. Taking inspiration from the birthday paradox, if attackers were to to hammer 64 chunks each in two different sessions, then the probability that at least one session would overlap between the two sessions is over 99.9%. One drawback to this approach is that having to hammering a large number of chunks would prolong the duration of the hammering phase, thereby making it less efficient. However, attackers can overcome this by building up their references across sessions, which would result in them having to hammer fewer chunks in the long term. For example, suppose the attackers have reference distributions to 64 different chunks obtained from one session. In a subsequent session, they hammer 64 other chunks on the same device such that only one chunk overlaps with the reference. They can now combine the distributions of the 63 non-matching regions to their reference to have a fresh reference from 127 different regions from the user's device. When running a subsequent session on the user's device, they have a higher probability that an allocated chunk would match their reference, since the reference size has increased. Upon reaching the limiting case where the attackers have managed to combine references to have references for all possible chunks, merely hammering a one chunk in the hammering phase would be sufficient, thereby resulting in higher efficiency.

One drawback of this approach is that it leaves behind a distinct signature since it initially requires access to multiple chunks of memory. We discuss an alternate approach to over come this limitation.

2) Incrementally building up references: Attackers can confine their hammering to fewer chunks (say, 2 chunks per sessions) to incrementally build up their references. In this case, attackers would initially have multiple sets of references for users without being able to link references that come from the same user. With this approach, attackers would also not be able to fingerprint users during their initial sessions. Eventually, after aggregating distributions from users across multiple sessions, attackers would be able to link sets of references to the same user and also be able to fingerprint users across all their sessions.

We elucidate this approach with an example in Figure 10. Say, during a user's first session when running the attacker's code, the attacker obtains the distribution of bit flips in 2 distinct chunks of 2 MB of memory, chunk A and chunk B. Since each chunk has a unique distribution of bit flips, neither chunk would match with any existing reference chunk maintained by the attacker, and thus, the attacker would create a fresh reference using these two chunks. In the next session with the same user, say the attacker obtains the distributions of 2 other chunks, chunk C and chunk D. Again, since the distribution of bit flips in each chunk is unique, the attacker will not be able to identify that this session corresponds to the same user, and would store them as a separate reference. In the next session with the user, say the attacker obtains the bit flip distribution to chunk B and chunk D. Now, the attacker would be able to fingerprint the user, and also combine the references containing chunks A and B with the references containing chunks C and D as chunks obtained from the same user. In a subsequent session with the same user, say the attacker obtains the bit flip distributions to chunk C and chunk E. For this session too, the attackers would be able to fingerprint the user and also extend their references for the user to contain the distribution of chunk E.

Thus, with the approach of incrementally building up references, in the long run, attackers would be able to fingerprint users while being more stealthy in terms of not leaving behind a signature. However, in order to do so, attackers would have to give up being able to fingerprint users during their initial sessions.

REFERENCES

[1] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. Fpdetective: Dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 1129–1140, New York, NY, USA, 2013. Association for Computing Machinery.

[2] Nampoina Andriamilanto, Tristan Allard, Gaëtan Le Guelvouit, and Alexandre Garel. A large-scale empirical analysis of browser fingerprints properties for web authentication. *ACM Trans. Web*, 16(1), sep 2021.

[3] Apple. User Privacy and Data Use. https://developer.apple.com/app-store/user-privacy-and-data-use/.

[4] Kuljit S. Bains and John B. Halbert. Distributed row hammer tracking, 2012. US Patent US20140095780A1.

[5] S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, 2005.

[6] Brave. Fingerprinting Protection Mode. https://github.com/brave/browser-laptop/wiki/Fingerprinting-Protection-Mode.

[7] Andrei Z. Broder. Some applications of rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*, pages 143–152. Springer-Verlag, 1993.

[8] Yinzhi Cao, Song Li, and Erik Wijmans. (cross-)browser fingerprinting via os and hardware level features. In *NDSS*, 2017.

[9] Chromium Blog. Update on User-Agent String Reduction in Chrome. https://blog.chromium.org/2021/05/update-on-user-agent-string-reduction.html.

[10] Davide Cozzolino and Luisa Verdoliva. Noiseprint: a CNN-based camera model fingerprint, 2018.

[11] Anupam Das, Nikita Borisov, and Matthew C. Caesar. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses. In *NDSS*, 2016.

[12] Finn de Ridder, Pietro Frigo, Emanuele Vannacci, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. SMASH: Synchronized many-sided rowhammer attacks from JavaScript. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1001–1018. USENIX Association, August 2021.

[13] Peter Eckersley. How unique is your web browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, page 1–18, Berlin, Heidelberg, 2010. Springer-Verlag.

[14] European Commission. EU data protection rules. https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en.

[15] FingerprintJS. FingerprintJS. https://github.com/fingerprintjs/fingerprintjs.

[16] Pietro Frigo, Emanuele Vannacc, 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 *2020 IEEE Symposium on Security and Privacy (SP)*, pages 747–762, 2020.

[17] Simson Garfinkel. Fingerprinting Your Files. MIT Technology Review, 2004.

[18] Google. Advertising ID. https://support.google.com/googleplay/android-developer/answer/6048248.

[19] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer.js: A remote software-induced fault attack in javascript. In *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9721*, DIMVA 2016, page 300–321, Berlin, Heidelberg, 2016. Springer-Verlag.

[20] Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu. Uncovering in-dram rowhammer protection mechanisms:a new methodology, custom rowhammer patterns, and implications. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 1198–1213, New York, NY, USA, 2021. Association for Computing Machinery.

[21] Hynix Semiconductor. Datasheet for 1Gb (32Mx32) GDDR5 SGRAM H5GQ1H24AFR. Technical Report H5GQ1H24AFR, 2009.

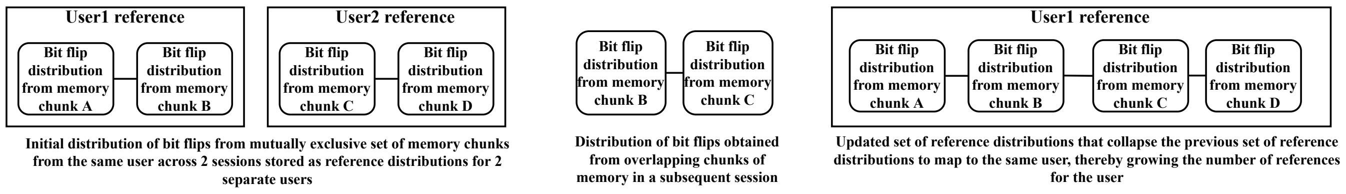
User1 reference
Bit flip distribution from memory chunk A
Bit flip distribution from memory chunk B
User2 reference
Bit flip distribution from memory chunk C
Bit flip distribution from memory chunk D
Initial distribution of bit flips from mutually exclusive set of memory chunks from the same user across 2 sessions stored as reference distributions for 2 separate users
Bit flip distribution from memory chunk B
Bit flip distribution from memory chunk C
Distribution of bit flips obtained from overlapping chunks of memory in a subsequent session
User1 reference
Bit flip distribution from memory chunk A
Bit flip distribution from memory chunk B
Bit flip distribution from memory chunk C
Bit flip distribution from memory chunk D
Updated set of reference distributions that collapse the previous set of reference distributions to map to the same user, thereby growing the number of references for the user

Fig. 10: Visualization of incrementally building up reference fingerprints. In this figure, we assume that the attackers first obtained bit flip distributions of 2 separate sets of contiguous chunks of memory from the same user across 2 separate sessions. Each set contains the distribution of bit flips observed on 2 such chunks. Since the distribution of bit flips on each chunk is unique, the attackers treat these sets as belonging to two different users. When they subsequently obtain bit flip distributions from chunks that overlap across both sets, they collapse them into a single set that pertains to the same user.

[22] Bruce Jacob, Spencer Ng, and David Wang. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[23] Patrick Jattke, Victor Van Der Veen, Pietro Frigo, Stijn Gunter, and Kaveh Razavi. Blacksmith: Scalable rowhammering in the frequency domain. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 716–734, 2022.

[24] JEDEC. DDR5 SDRAM. Technical Report JESD79-5B, August 2022.

[25] kernel development community. Transparent Hugepage Support. https://www.kernel.org/doc/html/latest/admin-guide/mm/transhuge.html.

[26] kernel.org. pagemap, from the userspace perspective. https://www.kernel.org/doc/Documentation/vm/pagemap.txt.

[27] 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. *SIGARCH Comput. Archit. News*, 42(3):361–372, jun 2014.

[28] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[29] Tomer Laor, Naif Mehanna, Antonin Durey, Vitaly Dyadyuk, Pierre Laperdrix, Clé mentine Maurice, Yossi Oren, Romain Rouvoy, Walter Rudametkin, and Yuval Yarom. DRAWN APART : A device identification technique based on remote GPU fingerprinting. In *Proceedings 2022 Network and Distributed System Security Symposium*. Internet Society, 2022.

[30] Eojin Lee, Ingab Kang, Sukhan Lee, G. Edward Suh, and Jung Ho Ahn. TWiCe: Preventing Row-Hammering by Exploiting Time Window Counters. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, page 385–396, New York, NY, USA, 2019. Association for Computing Machinery.

[31] J.W. Lee, Daihyun Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, pages 176–179, 2004.

[32] Dawei Li, Di Liu, Yangkun Ren, Ziyi Wang, Zhenyu Guan, and Jianwei Liu. Device identification in multimedia systems based on dram fingerprinting, 2022.

[33] Micron. DDR4 SDRAM. https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/8gb_ddr4_sdram.pdf.

[34] Micron Technology. TN-46-12: Mobile DRAM Power-Saving Features and Power Calculations. Technical Report TN46_12, 2005.

[35] Micron Technology. DDR2 SDRAM . Technical Report MT47H512M4,MT47H256M8,MT47H128M16, 2006.

[36] Micron Technology. TN-41-01: Calculating Memory System Power for DDR3. Technical Report TN41_01DDR3, January 2007.

[37] Micron Technology. 1.35V DDR3L SDRAM SODIMM. Technical Report MT16KTF51264HZ, MT16KTF1G64HZ, 2011.

[38] Micron Technology,. DDR4 SDRAM. Technical Report MT40A2G4, MT40A1G8, MT40A512M16, 2015.

[39] Micron Technology. TN-ED-03: GDDR6: The Next-Generation Graphics DRAM . Technical Report TN-ED-03: GDDR6, 2017.

[40] Mike Perry, Erinn Clark, Steven Murdoch and Georg Koppen. The Design and Implementation of the Tor Browser. https://2019.www.torproject.org/projects/torbrowser/design/.

[41] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. Fingerprinting information in javascript implementations.

[42] Mozilla. Using HTTP cookies. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.

[43] Mozilla Wiki. Changing the UA String. https://wiki.mozilla.org/Changing_the_UA_String.

[44] Shaoor Munir, Sandra Siby, Umar Iqbal, Steven Englehardt, Zubair Shafiq, and Carmela Troncoso. Cookiegraph: Measuring and countering first-party tracking cookies, 2022.

[45] Notes on AI. Jensen-Shannon Divergence. https://notesonai.com/Jensen%E2%80%93Shannon+Divergence.

[46] Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, and Jae W. Lee. Graphene: Strong yet lightweight row hammer protection. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13, 2020.

[47] David A. Patterson and John L. Hennessy. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[48] Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. DRAMA: Exploiting DRAM addressing for Cross-CPU attacks. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 565–581, Austin, TX, August 2016. USENIX Association.

[49] Samsung Electronics. DDR4 SDRAM. Technical report, 2014.

[50] Iskander Sanchez-Rola, Igor Santos, and Davide Balzarotti. Clock Around the Clock: Time-Based Device Fingerprinting. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1502–1514, New York, NY, USA, 2018. Association for Computing Machinery.

[51] Andre 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 *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, may 2017.

[52] S. M. Seyedzadeh, A. K. Jones, and R. Melhem. Counter-Based Tree Structure for Row Hammering Mitigation in DRAM. *IEEE Computer Architecture Letters*, 16(1):18–21, 2017.

[53] Mungyu Son, Hyunsun Park, Junwhan Ahn, and Sungjoo Yoo. Making DRAM Stronger Against Row Hammering. In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, New York, NY, USA, 2017. Association for Computing Machinery.

[54] State of California Department of Justice. California Consumer Privacy Act (CCPA). https://oag.ca.gov/privacy/ccpa.

[55] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clementine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic rowhammer attacks on mobile platforms. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1675–1689, New York, NY, USA, 2016. Association for Computing Machinery.

[56] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. Fp-stalker: Tracking browser fingerprint evolutions. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 728–741, 2018.

[57] Whatismybrowser. The latest user agents for Safari. https://www.whatismybrowser.com/guides/the-latest-user-agent/safari.

[58] A. Giray Yağlikçi, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu. Blockhammer: Preventing rowhammer at low cost by blacklisting rapidly-accessed dram rows. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 345–358, 2021.

[59] Jiexin Zhang, Alastair R. Beresford, and Ian Sheret. SensorID: Sensor Calibration Fingerprinting for Smartphones. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 638–655, 2019.

[60] B. Zhao, Y. Du, J. Yang, and Y. Zhang. Process variation-aware nonuniform cache management in a 3d die-stacked multicore processor. *IEEE Transactions on Computers*, 62(11):2252–2265, 2013.