Probable Cause:
The Deanonymizing Effects of Approximate DRAM

Amir Rahmati Matthew Hicks Daniel E. Holcomb Kevin Fu
University of Michigan
{rahmati,mdhicks,danholcomb,kevinfu}@umich.edu

Abstract

*Approximate computing research seeks to trade-off the accuracy of computation for increases in performance or reductions in power consumption. The observation driving approximate computing is that many applications tolerate small amounts of error which allows for an opportunistic relaxation of guard bands (*e.g.*, clock rate and voltage). Besides affecting performance and power, reducing guard bands exposes analog properties of traditionally digital components. For DRAM, one analog property exposed by approximation is the variability of memory cell decay times.*

In this paper, we show how the differing cell decay times of approximate DRAM creates an error pattern that serves as a system identifying fingerprint. To validate this observation, we build an approximate memory platform and perform experiments that show that the fingerprint due to approximation is device dependent and resilient to changes in environment and level of approximation. To identify a DRAM chip given an approximate output, we develop a distance metric that yields a two-orders-of-magnitude difference in the distance between approximate results produced by the same DRAM chip and those produced by other DRAM chips. We use these results to create a mathematical model of approximate DRAM that we leverage to explore the end-to-end deanonymizing effects of approximate memory using a commodity system running an image manipulation program. The results from our experiment show that given less than 100 approximate outputs, the fingerprint for an approximate DRAM begins to converge to a single, machine identifying fingerprint.

1. Introduction

Secure system designers tend to focus on the anonymity of communication [25] and take for granted the hardware used to generate the data communicated. Attribution of data is usually done through communication meta-data [2]. While the use of encryption secures the communication against eavesdroppers, it is unable to hide the occurrence of communication.

ISCA'15, June 13-17, 2015, Portland, OR USA

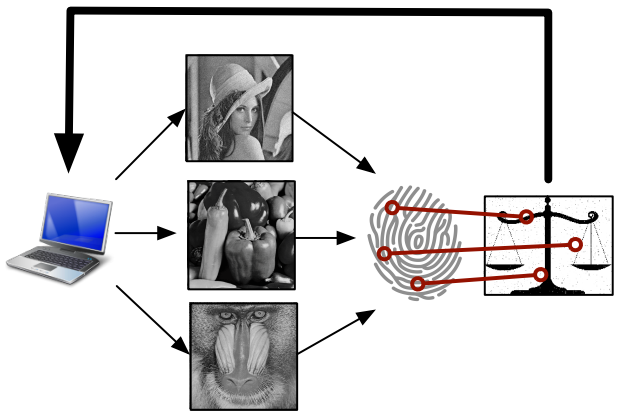

Figure 1: Probable Cause creates a fingerprint of an approximate DRAM system by collecting approximate outputs and stitching together error patterns in those outputs to form a fingerprint for the memory. Attackers can then use this memory fingerprint to identify other approximate outputs as belonging to the system.

Anonymity systems such as Tor [5] try to provide this guarantee over the Internet. Even when software and communication channels are designed to preserve anonymity of users, devices can be deanonymized using intrusive measures such as espionage tools and Trojans [36] or non-intrusively using unique characteristics of analog hardware such as RF fingerprinting [1, 26], clock skew [14], or camera sensor noise [19]. The anonymity of digital computation has not been traditionally a concern since, in general, computer systems are deterministic machines that yield identical results to identical inputs.

The assumption of anonymous computation must be reconsidered with the emergence of approximate computing. The goal of approximate computing is to provide significant performance improvements and/or energy savings by sacrificing the accuracy of computation or storage. In many cases, the error pattern due to approximation depends on hardware variations locked-in at manufacturing time. The dependency of computation result on hardware properties creates an opportunity for an attacker to deanonymize systems that produce approximate results.

Approximate computing adds accuracy as a third dimension to the conventional energy/performance trade-off. Many applications, such as computer vision, machine learning, and

sensor networks, are naturally imprecise and thus accept a range of results, so expending extra time and energy to calculate an exact results is of no advantage. For example, any application that uses floating point numbers already accepts some inaccuracy.

As one of the main components of an approximate system, many works consider the trade off between accuracy and energy saving in Dynamic Random Access Memory (DRAM). Energy saving schemes targeted at DRAM work by lowering the input voltage [3] or by decreasing the refresh rate [17, 18, 40]. These techniques are a key component in future approximate computing systems, especially those that tolerate limited errors in data [6].

While much of the previous work has examined approximate DRAM's impact on correctness, performance, and energy, none of the existing approximate DRAM systems consider their impact on privacy. To this end, we introduce Probable Cause, to our knowledge, the first paper that explores the security implications of approximate DRAM. Probable Cause is an approach to uniquely identify approximate computing systems based on the error pattern imprinted in approximate outputs. Figure 1 provides an overview of how Probable Cause works. The insight driving Probable Cause is that the error pattern imprinted on data reveals the location of the most volatile cells in an approximate memory. Additionally, this volatility is chip-specific and due mainly to process variations locked-in during manufacturing.

To demonstrate the real-world implications of our observation, we implement Probable Cause. Probable Cause consists of an approximate memory system and set of approximate result classification algorithms. We show that Probable Cause reliably deanonymizes approximate results, even with changes in temperature and level of approximation. Additionally, we show that it is possible to dynamically construct a fingerprint for a DRAM by collecting arbitrary approximate results and stitching their individual fingerprints together to form a whole-memory fingerprint.

Our contributions are,

- We present the first work to highlight the privacy implications of approximate DRAM.
- We empirically evaluate the feasibility of our approach by deanonymizing DRAM devices based only on their approximate results.
- We present a mathematical model to quantify the end-to-end information leakage of approximate DRAM, showing how many approximate results an attacker must gather to reliably identify a system.

2. Background

Dynamic Random Access Memory (DRAM) is a type of volatile memory that stores values by holding charge in a capacitor. Figure 2 presents a simplified DRAM structure. The storage capacitor in each DRAM cell has a default/uncharged state and a charged state. The uncharged state of a cell corresponds to either a logical '0' or a logical '1', depending on the DRAM mapping. For each cell, the logical value corresponding to an uncharged capacitor is denoted as the default value. Generally, all cells in the same row have the same default value, and the default value alternates every few rows. Writing a value opposite of the default value charges a cell's storage capacitor. The capacitor then begins to lose its charge. Eventually the capacitor voltage will drop below a detection threshold and return the cell to its default value. To prevent data loss in charged cells, DRAM must perform regular refresh operations. The JEDEC standard [13] specifies a refresh period of 64ms for operating temperatures below 85°C. Refreshes have row granularity (due to the architecture of DRAM). At the hardware level, a refresh operation is a read followed by a write. The write fully charges any data storage capacitors not in the default value.

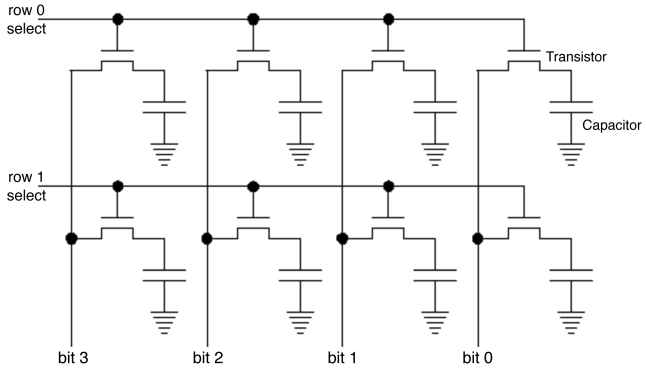

Figure 2: A DRAM cell has a default low value that can be changed by charging the capacitor. DRAM cells need to be constantly refreshed for the value to hold, otherwise capacitor leakage slowly reverts the cell to its default value. All DRAM operations are done at row granularity.

DRAM cells decay at different rates, mainly due to their manufacturing variations. The distribution of how quickly DRAM cells decay follows a Gaussian distribution [27]. There are two types of manufacturing variation that influence the probability of state loss between refresh: (1) variation in the capacitance of the DRAM cell and (2) variation in the leakage current through the access transistor that drains the capacitor. It is possible that some variation in capacitance is mask-dependent, thus replicated across wafers produced in the same fabrication process. On the other hand, the variation in the leakage current is not mask-dependent, because it is caused by threshold voltage variations due to random dopant fluctuations in the channel of the access transistor. Thus, we expect leakage current to be the dominant factor in DRAM cell retention time, *i.e.,* essentially mask independent.

In traditional/exact computing models, a DRAM requires frequent refreshes to prevent decay of the most volatile cells in the most extreme environmental conditions. This results in large overheads because, while some cells decay in less than a tenth of a second, the majority of the cells hold their value for

tens of seconds. Additionally, most systems are not running in extreme environments.

Approximate computing systems take advantage of this opportunity either by lowering the supply voltage of memory or by decreasing the refresh rate. Both of these methods result in energy savings but cause errors in data. Given that the errors are mainly due to capacitor leakage, the ordering of cells that lose their charge is repeatable. This observation drives Probable Cause. In the remainder of the paper, we experimentally show that these orderings are unique, stable given environmental changes, and stable given the amount of error.

3. Threat Model

Probable Cause's threat model assumes that a user has a system with approximate memory. The user wishes to publish data (e.g., post a picture on a forum) created on an approximate system while preserving his or her anonymity. We assume that the user takes all known precautions, such as removing identifying meta-data from the files they post and that they publish data using an anonymity-preserving communication channel (e.g., The Onion Router (Tor) [5]).

A key aspect of the threat model is a resource imbalance between the attacker and the victim: it assumes a sophisticated attacker with abundant resources (*i.e.,* a nation state) that seeks to identify a relatively small set of users (*e.g.,* a dissident) using only those users' approximate outputs. Figure 3 depicts two attack scenarios explored in this paper:

(a) The attacker inserts themselves in the supply chain between the manufacturer and the end user. This encompasses the attacker intercepting complete computer systems or just the DRAM modules themselves. The attacker fingerprints devices completely before they reach the user, thus Probable Cause can deanonymize any public approximate result generated by the system.

(b) The attacker creates a database of all observed approximate outputs. The error patterns in the outputs are stitched together to form whole-system fingerprints. In this scenario, we assume that the attacker has access to the public data and can guess the positions of error in the approximate outputs. While this scenario is less intrusive, it requires collecting many approximate outputs from a system before Probable Cause is able to construct a reliable system-level fingerprint.

Both the supply-chain attack and eavesdropping attack are feasible given real-world precedents [8].

4. Design of Probable Cause

The two scenarios described in Section 3 pose very different attack vectors for the adversary to deanonymize data generated by an approximate memory. Attacking the supply chain is the easier of the two attacks to implement. Giving the adversary physical access to the approximate memory guarantees complete and accurate fingerprinting of the memory. Section 7.1 covers how data only a few memory pages in length can produce a fingerprint powerful enough to differentiate outputs from one DRAM chip from another. The second attack scenario is more challenging since the attacker cannot control what data the victim gives him. This section shows that even with such limitations, Probable Cause still deanonymizes users based solely on user-provided approximate outputs.

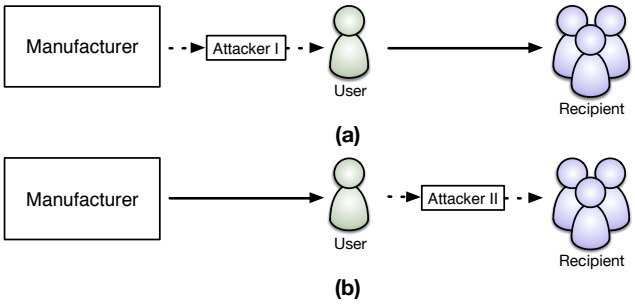

Figure 3: Probable Cause tackles two attack scenarios: (a) the attacker intercepts and fingerprints the entire memory (as a part of a system or a standalone module) in the supply chain and (b) the attacker captures approximate outputs from a deployed system to create a fingerprint.

For the post-deployment attack scenario, we assume the attacker has access to approximate outputs from the device, but does not know which page[1] of memory it emanates from. To formalize this, assume that we have approximate outputs $D_1, D_2, ..., D_n$. Without loss of generality, we assume that these pages are stored in physical memory pages $s_1, s_2, ..., s_n$ and have length of $l_1, l_2, ..., l_n$ consecutive pages. Note that this is not a strong assumption as even operating systems that utilize Address Space Layout Randomization (ASLR) [34] do *not* randomize the location of the pages that make up a file due to the added management overhead.

To create a holistic picture of memory, Probable Cause treats each output as a piece of a puzzle that it puts together to create a fingerprint of the entire memory. Figure 4 depicts how this process works: initially, Probable Cause creates a fingerprint for every page of data that it sees. Therefore, each approximate output will be a contiguous series of page-level fingerprints $FP_1, FP_2, ..., FP_n$ with length of $l_1, l_2, ..., l_n$ pages, respectively. Next Probable Cause tries to stitch these page-level fingerprints together into a system-level fingerprint by searching for overlap among the series of connected page-level fingerprints. If the page-level fingerprints of two approximate outputs match, then there is a range of physical memory pages that held both outputs. Probable Cause uses the page-level fingerprints outside the overlap region to create a combined system-level fingerprint that encompasses the page-level fin-

[1] Our analysis focuses on 4 KB chunks of memory—called a page, because that is the smallest unit of contiguous memory that operating systems manage. Modern operating systems also use larger page sizes, which only makes our analysis easier.

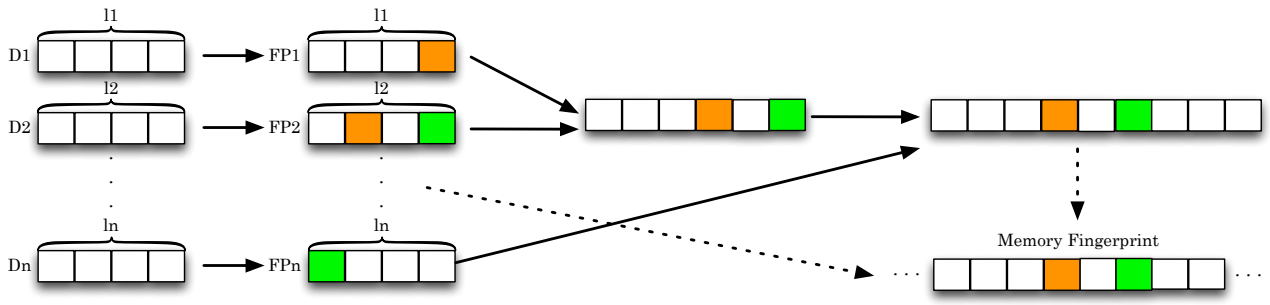

Figure 4: Probable Cause constructs the whole-memory fingerprint by stitching together fingerprints of overlapping approximate outputs. Pages of the same color are the same page and are matched by Probable Cause using page-level fingerprints.

gerprints of each output. As the number of outputs increase, more fingerprints are stitched together. In Section 7.6 we show how, with large enough data and enough overlap, it is possible to create a system-level fingerprint comparable to the supply chain attack. In cases where the approximate outputs were *not* stored in any of the same physical memory pages, Probable Cause must assume that the outputs come from different systems.

Probable Cause stores system-level fingerprints in a database equal to the size of the fingerprinted region of memory. Although we do not imagine storage to be an issue for powerful attackers such as government agencies or Advanced Persistent Threats (APTs), it is possible to reduce the storage requirement by only tracking the fast decaying bits of memory (approximately, 1% of the bits in a memory).

5. Mechanics of Probable Cause

Probable Cause's goal is to identify the origin of approximate data based on the error pattern imprinted by approximate DRAM. Figure 5 presents three example outputs of two approximate DRAMs. For this example, a 200×154 pixel black and white image is stored in two different DRAM chips refreshed at a rate that yields 1% error with worst-case data. Figure 5.a and Figure 5.b show the image produced by the same chip, but at different temperatures, while Figure 5.c shows the output from a second chip.

Even from visual observation, it is possible to distinguish the results coming from a different chip as there are many similarities in the error patterns in Figures 5.a and 5.b, but no real similarity to Figure 5.c. We highlight regions with notable similarities and differences to ease the comparison.

It is not practical to expect a user to analyze the error pattern in every approximate output for similarities to the known error patterns. Thus, this section presents the algorithms used by Probable Cause to cluster approximate results and identify host systems based on known system-level fingerprints and observed approximate outputs. There are three parts to this problem: Section 5.1 covers generating system-level fingerprints for DRAM chips. Section 5.2 covers correlating approximate results and system-level fingerprints. Finally, Section 5.3 covers clustering approximate results with the same system-level fingerprint and determining the system that produced them, even when they have not been previously seen by the attacker.

Algorithm 1 Characterization Algorithm: Creates a fingerprint for a DRAM chip based on the errors from several approximate results.

CHARACTERIZE($approx[\#ofResults][size]$)

1 **for** $i \leftarrow 1$ **to** $\#ofResults$
  $\succ$ exact is a bitstring representing an unapproximated result
2 **do** $errorString[i] \leftarrow \text{XOR}(approx[i], exact)$
  $\succ$ Fingerprint is the intersection of error bits
3 **return** $\wedge_{i=1}^{\#ofResults} errorString[i]$

5.1. Characterization

The first step required for Probable Cause to successfully deanonymize a user is characterization. To characterize an approximate memory, Probable Cause needs a series of approximate results. Based on the adversarial model described in Section 3, there are two possible paths for the attacker to acquire these: (1) the attacker gets physical access to the system or DRAM chip and characterizes it completely using their own inputs, or (2) the attacker collects user-published approximate outputs from the system by eavesdropping or by scraping the web.

Algorithm 1 characterizes a DRAM chip by collecting a series of approximate results from the chip along with their corresponding exact values. Next, it detects the pattern of errors in each of the results and records the intersection of the errors as the fingerprint of the chip. Given that we expect most of the failed bits to match during different runs, using the intersection will minimize the effect of noise—keeping only the most volatile bits. Keeping such a small number errors around as the fingerprint has several advantages: it makes the fingerprint amenable to lightly approximated systems, it provides ample information to correctly classify approximate outputs and identify systems, and it makes DRAM chip clas-

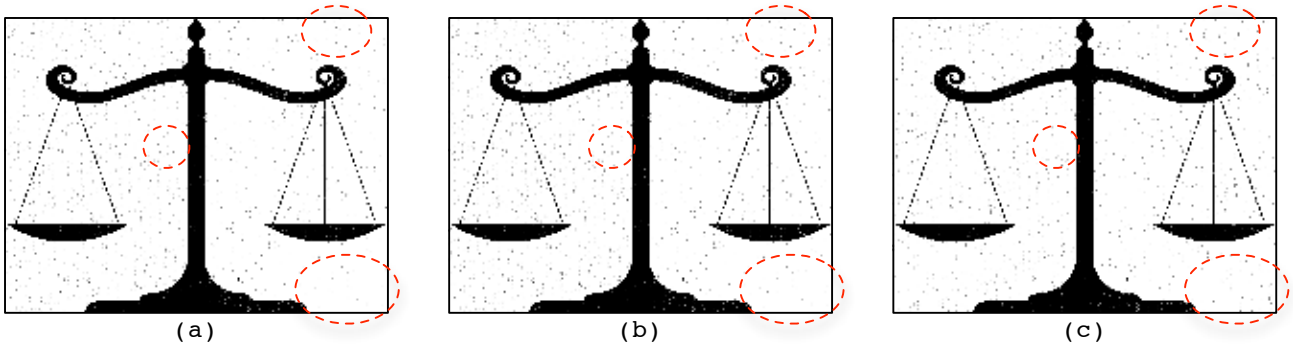

Figure 5: Three identical images after storage in approximate memory. Image (c) is stored in a different chip than (a) and (b). Simple visual inspection reveals similar patterns of errors in results coming from the same chip.

Algorithm 2 Identification Algorithm: Compares an approximate output with fingerprints in a database to identify which DRAM chip produced the output.

```
IDENTIFY(approx[size], fingerprintDB[#ofFPs], exact[size])
1  errorString ← XOR(approx, exact)
2  for i ← 1 to #ofFPs
3      do if DISTANCE(errorString, fingerprintDB[i])
              < threshold
4             do return i
5  return failed
```

sification fast as it takes little time for the first 1% of bits to fail.

5.2. Identification

To correctly match an approximate output with a system-level fingerprint, Algorithm 2 first detects errors in the approximate data by comparing it to the exact data. It then searches a database of system-level fingerprints to see if any match the error pattern of the output. For comparisons, the algorithm uses the distance metric described in Algorithm 3. The algorithm returns the first system-level fingerprint whose distance to the error pattern in the output is below a pre-defined threshold. Section 7 discusses how we experimentally determine this threshold.

For a distance metric, one might think that using the Hamming distance is suitable. Unfortunately, the Hamming distance is unable to perform well in cases where the amount of error in the system-level fingerprint and the approximate output differ dramatically (*e.g.,* the chip is characterized at 99% accuracy while the data is 95% accurate). In such cases, an approximate result from the same chip as the fingerprint, but with much less error will look farther away than an approximate result from another chip that has much more error than the fingerprint. To compensate for this, we designed a custom distance metric (detailed in Algorithm 3) based on Jaccard's

Algorithm 3 Distance Algorithm based on Jaccard index [12].

```
DISTANCE(errorString[size], fingerprint[size])
1  Initialize d ← 0
   ≻ Count the number of errors in fingerprint which are ab-
     sent in errorString
2  for i ← 1 to size
3      do if fingerprint[i] = 1 and errorString[i] = 0
4             do d ← d + 1
5  return d / HAMMINGWEIGHT(errorString)
```

index [12]. Our metric looks for errors that exist in the fingerprint, but are absent in output's error pattern[2]. This result is then normalized to the number of errors in the fingerprint. The result is a distance that ranges from $[0,1]$. Our distance metric does not suffer from the varying approximation problem as it only looks for error bits that should be present if data originated from the fingerprinted memory and ignores any additional errors that could have happened because of mismatch in level of approximation. Our metric is also less prone to noise as it similarly ignores random bit flips that might have occurred because of noise.

5.3. Clustering

To support the second attack where the attacker has not preemptively fingerprinted devices, Probable Cause must be able to cluster results of unknown or previously unseen devices in addition to identifying approximate outputs created by known devices. Our clustering algorithm is similar to the approach discussed in Section 4. Each approximate result creates an error string that is compared to each of the previously identified clusters using the distance metric. If the error string matches any of the clusters, it will be intersected with the fingerprint of the cluster to augment it (similar to approach used in the char-

[2]Without loss of generality, we assume that the fingerprint has less error bits. When the approximate output has less error bits, it can be treated as the "fingerprint".

Algorithm 4 Clustering Algorithm: Creates a fingerprintDB based on a set of approximate results.

```
CLUSTER(approx[#ofResults][size], exact)
1  Initialize cluster ← 0
2  for i ← 1 to #ofResults
3      do j ← 0
4         errorString ← MARKERROR(approx[i], exact)
5         while j < cluster
6             do if DISTANCE(errorString, fingerprintDB[j])
                    < threshold
7                  do fingerprintDB[i]
                       ← fingerprintDB[i] ∧ errorString
8                  goto 2
9         fingerprintDB[cluster] ← errorString
10        cluster ← cluster + 1
11 return fingerprintDB
```

acterization algorithm). In cases where the error string does not match any of the clusters, it will be assigned to a new cluster (representing the system-level fingerprint of a new system). Algorithm 4 describes the pseudo-code of this algorithm. This algorithm has three main benefits: (1) it requires minimum supervision from the user, (2) it is low cost compared to more complicated machine learning techniques, and (3) the chance of a mismatch is low due to the performance of our modified Jaccard distance metric.

6. Experimental Setup

We evaluate our system on both an older DRAM and a DDR2 platform. Because of similarity in results, we postpone our description of the DDR2 setup and the effect of process technology on Probable Cause to Section 8.1. Our DRAM experiments consist of a set of 10 32KB KM41464A DRAM chips [33]. This DRAM stores data as 64K 4-bit words, arranged in 256 columns and 256 rows. We disable automatic refresh, thus the only way to refresh a row is through memory accesses. Other relevant blocks and their roles are,

- The MSP-FET430UIF [38] JTAG Programmer is responsible for programming the microcontroller and later transferring the results back to the analysis computer.
- The MSP430-F2618 [37] microcontroller orchestrates the experiments. Its duties include writing and reading data to and from the DRAM, controlling the timing of refreshes, and analyzing the data from the DRAM for decay.
- The Sun Electronics EC-12 thermal chamber [35] allows us to control temperature for the DRAM experiments. Temperature is the most important environmental factor to control as the rate of decay in DRAM heavily depends on its variations [27].
- The Agilent power supply powers the DRAM.

For experiments not involving image data, we load data that charges every memory cell in the DRAM. Section 2 discusses how each DRAM cell has a charged state which corresponds to logical 1 or 0, depending on the row. Using the charged value of cells has the advantage that it gives every cell the possibility of losing state by decaying to the default value—a worst case scenario.

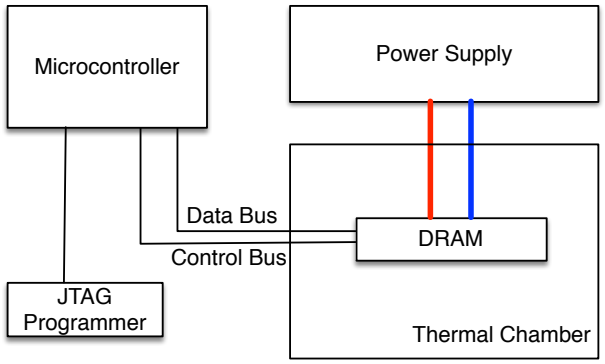

Figure 6: Probable Cause experimental platform. The MSP430 microcontroller controls DRAM read/write functions. The target DRAM is placed inside a thermal chamber to ensure environment consistency across experiments. The JTAG programmer allows us to program the microcontroller and extract the results.

7. Evaluation

To evaluate Probable Cause, we start by examining it with respect to five factors that affect the performance of DRAM fingerprinting. All of these experiments run on the approximate memory platform presented in Section 6. The five factors are

1. **Uniqueness:** How distinguishable are the fingerprints of different chips from each other?
2. **Consistency:** How much variation exists in the fingerprint of a single chip across multiple trials, given the same conditions?
3. **Thermal effect:** How does temperature impact the relative volatility of DRAM cells?
4. **Order of failure:** How do fingerprints coming from data produced on the same chip, but with different levels of approximation correspond to each other?
5. **Accuracy versus privacy:** How do changes in the level of approximation impact the ability of Probable Cause to successfully identify the outputs of a chip?

Then, using the results from the generalized evaluation, we create a mathematical model to evaluate the end-to-end deanonymizing effects of approximate memory using a commodity system with an approximate computing benchmark program.

7.1. Uniqueness

The goal of this experiment is to show that Probable Cause correctly associates an approximate output to the DRAM chip that produced it, given a system-level fingerprint of all DRAM

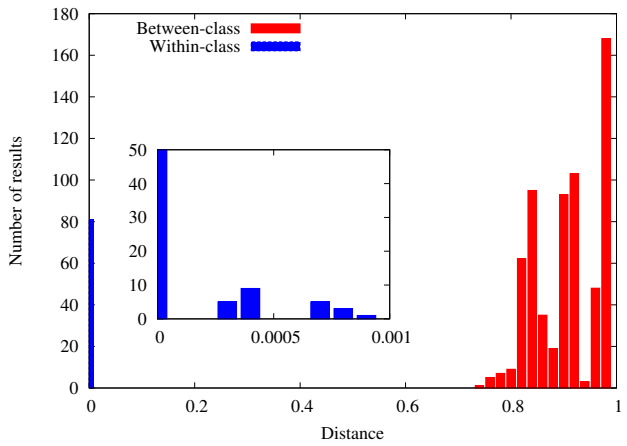

Figure 7: Histogram of fingerprint distances for within-class (same chip) and between-class (other chips) pairings.

chips. To evaluate the uniqueness of fingerprints, we first create a system-level fingerprint for each chip by taking the intersection of the error bits in three outputs created at 1% error and different temperatures. We then create 9 approximate data outputs from each of our 10 DRAM chips, where each output comes from a different combination of temperature (40°C, 50°C, and 60°C) and level of approximation (99%, 95%, and 90%).

For each of the 90 results, we calculate the error bitstring and use Algorithm 3 to calculate the distance between the output and every system-level fingerprint. Figure 7 is a histogram of the within-class (belonging to same chip) and the between-class (belonging to different chips) distance of every pair of fingerprints. The between-class distances are two orders-of-magnitude larger than within-class distances; this allows Algorithm 2 to trivially deanonymize chips from their approximate data.

Uniqueness can also be evaluated theoretically by reasoning about the space of possible fingerprints. If the possible number of fingerprints is low compared to the number of devices, it would be likely for fingerprints of two devices to match or be close enough to make them indistinguishable using our metric.

Assuming an approximate memory of size M bits where A bits of error are tolerated, the total number of unique fingerprints is given by the binomial coefficient in Equation 1.

$$Max\ unique\ fingerprints = \binom{M}{A} \tag{1}$$

Given the existence of noise, fingerprints will not match exactly, and a threshold of T bits is used for matching two fingerprints. Using this threshold, every fingerprint is matchable with $\sum_{i=0}^{T}\binom{M}{i}$ fingerprints that are within Hamming distance T. Taking into consideration that the noise threshold exists for both the system-level fingerprint and the approximate output, the range of possible distinguishable fingerprints is calculated

One page of memory

$M = 32768$ bits, $A = 1\%$, $T = 32$ bits

Max possible fingerprints	8.70×10^{795}
Max unique fingerprints	$\geq 1.07 \times 10^{590}$
Chance of mismatching	$\leq 9.29 \times 10^{-591}$
Total Entropy	2423 bits

Table 1: Results for a page of memory

using the Hamming bound [20]:

$$\frac{\binom{M}{A}}{\sum_{i=0}^{2T}\binom{M}{i}} \leq Max\ distinguishable\ fingerprints \leq \frac{\binom{M}{A}}{\sum_{i=0}^{T}\binom{M}{i}} \tag{2}$$

and the chance of two fingerprints being mistakenly matched is in the range of:

$$\frac{\sum_{i=1}^{T}\binom{M}{i}}{\binom{M}{A}} \leq Chance\ of\ mismatching \leq \frac{\sum_{i=1}^{2T}\binom{M}{i}}{\binom{M}{A}} \tag{3}$$

The surprisingly low chance of misidentification is due to the high amount of entropy in the fingerprints. Assuming that noise and other external factors cause no more than T bit-flips ($A > T$), the amount of entropy per bit of memory is given by Equation 4.

$$entropy/bit \geq \frac{\log_2\left(\frac{\binom{M}{A}}{\sum_{i=0}^{2T}\binom{M}{i}}\right)}{M} \geq \frac{\log_2\binom{M}{A-T}}{M} \tag{4}$$

To put these equations into perspective, Table 1 presents these result for a page of memory ($M = 32768$ bits) with a $A = \frac{1}{100}\ M$ (328 bits), and threshold of $T = \frac{10}{100}\ A$ (32 bits). This threshold value is a safe upper bound chosen based on our experiment results.

7.2. Consistency

The goal of this experiment is to show that, given the same operating conditions, DRAM cells fail in a repeatable fashion. To evaluate the consistency of errors in an approximate DRAM across different runs, we record 21 outputs of a DRAM chip at 99% accuracy and 40°C, then compare the error locations in each output. Figure 8 presents a heatmap of the bits that are not predictable across different trials. In the heatmap, the darker the cell, the more it behaves like noise. Our results show that 98% of bits that fail in any one trial, will also fail in the other 20 trials. This suggests that the errors created by approximate DRAM are mostly repeatable.

7.3. Thermal effect

Temperature variation is known to have a significant effect on the rate of charge decay in DRAM [10]. DRAM refresh rates

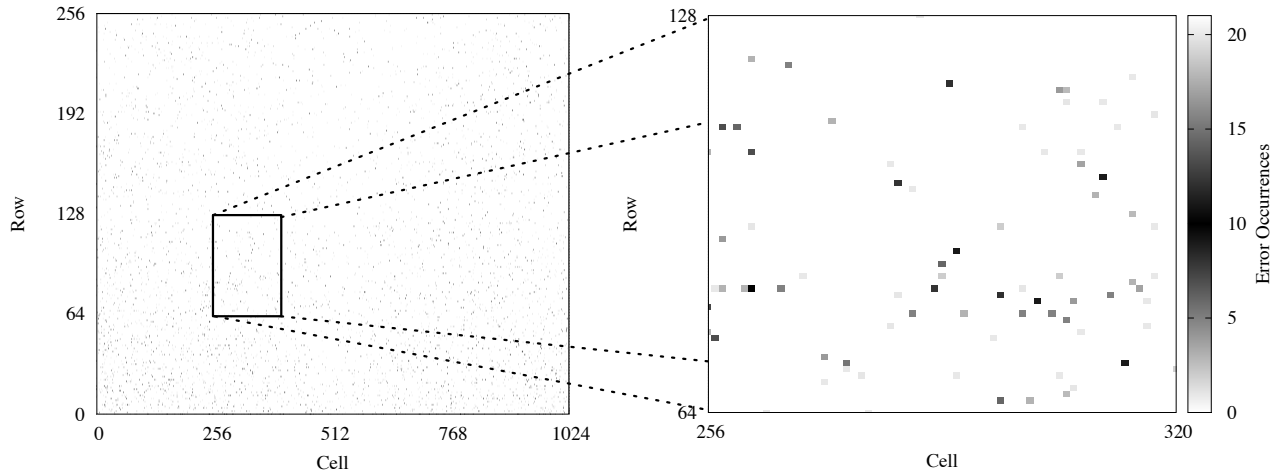

Figure 8: Heatmap of cells unpredictability in a sample DRAM chip. Darker cells behave more like noise. More than 98% of cells behave reliably across all 21 runs.

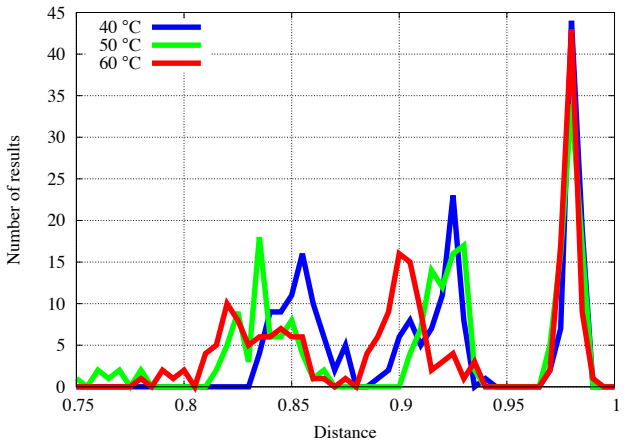

Figure 9: Histogram of between-class (different chips) pair distances grouped by temperature. Temperature has no noticeable effect on distance.

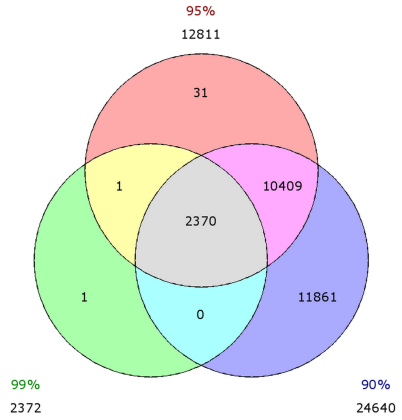

Figure 10: Overlap of a DRAM error locations at different levels of approximation. The results support a rough subset relation $99\% \subset 95\% \subset 90\%$.

account for this either by assuming a worst-case operating environment [13] or by dynamically adjusting the refresh rate to compensate for environmental changes while keeping current consumption minimized [22]. Our approximate DRAM implementation similarly adjusts its refresh rate to maintain a desired accuracy across changes in temperature. To explore whether the change of temperature affects the relative DRAM cell decay rates, we run experiments under different temperatures (40°C, 50°C, and 60°C) and different levels of approximation (99%, 95%, and 90%). Figure 9 shows how variations in temperature affects between-class (different chips) pair distance. Even though the increased temperature causes DRAM cells to decay faster, our approximate DRAM system accounts for these changes to maintain the desired level of approximation. The results show that the relative decay rate of DRAM memory cells is robust to temperature change and thus, does *not* impact Probable Cause.

7.4. Order of failures

Based on the consistency of errors in approximate DRAM, we hypothesize that the decay of cells within each DRAM chip follows a particular order that is mostly consistent across experiments. To verify this, we record failed bits of a chip at three different levels of approximation (99%, 95%, and 90%) and evaluate the overlap in error locations in these results. Figure 10 presents a Venn diagram of the overlaps. Aside from a single outlier, all erroneous cells at 99% accuracy are a subset of the cells that are erroneous at 95% accuracy, which, aside from 32 cells, are a subset of those at 90% accuracy. This result supports our hypothesis about the existence of an ordering in DRAM cell failures.

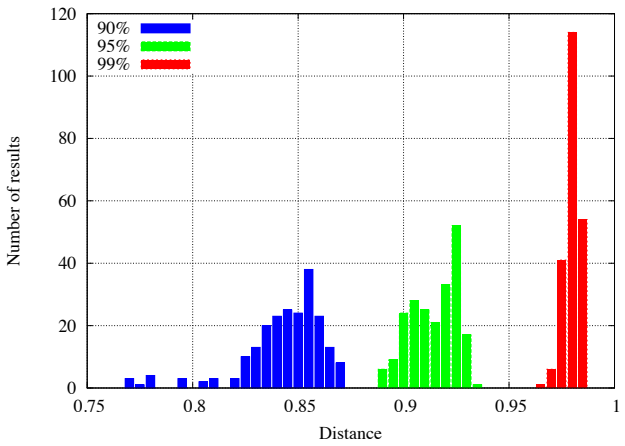

Figure 11: Histogram of between-class chip distance grouped by approximate memory accuracy. The increased chance of bit error overlap causes the average distance to shrink with increases in approximation. Note that these distances are still two orders larger than the largest within-class distance.

7.5. Accuracy versus privacy

Depending on the application, an approximate system may use different levels of accuracy. As the accuracy of data decreases, the number of errors increase proportional to the size of the memory. In contrast, the increased number of error bits creates greater chance of overlap with the fingerprint from out-of-class chips, decreasing the distance between two distinct chips. Going back to our mathematical model from Section 7.1, lowering the accuracy is expected to result in an exponential increase in the fingerprint state space—making a misclassification exponentially more likely. Table 2 presents the maximum chance of mismatch at different accuracies for a page of memory.

We also evaluate the effect of varying accuracies on our distance metric. Figure 11 presents the histogram of between-class (other chips) distances at three different accuracies. As expected, the greater chance of overlap causes the distance to decrease as the accuracy decreases, but at the levels of approximation used in the literature, there is still a vast divide between within-class distances and between-class distances.

7.6. Eavesdropping attacker evaluation

The results up to this point make it clear that it is possible to identify the DRAM chip that produced an approximate output in a variety of operating conditions. The goal of this experiment is to understand the end-to-end deanonymizing effects of approximate memory given the constraints of a commodity system, an approximate computing benchmark, and the more difficult post-deployment attack model. The setup for this experiment is an iMac running Ubuntu 14.04 inside a virtual machine with 1 GB of memory allocated. On this platform, we run a Valgrind [23] instrumented edge-detection program

Accuracy	Chance of mismatch
99%	$\leq 9.29 \times 10^{-591}$
95%	$\leq 8.78 \times 10^{-2028}$
90%	$\leq 4.76 \times 10^{-3232}$

Table 2: Chance of mismatching two pages of memory for different accuracies. Decreasing accuracy causes an exponential increase in fingerprint state space.

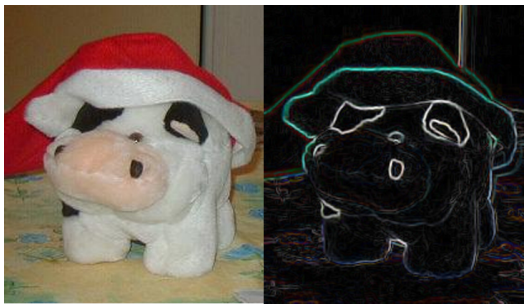

Figure 12: Sample input (left) and output (right) of CImg gradient edge-detection code used to evaluate Probable Cause.

from the CImg open-source image processing library [39]. Figure 12 shows a sample input and output of this program. We run the program and analyze the report from Valgrind to uncover the physical pages the program used to store its approximate outputs. Using this data, along with the mathematical model presented in Section 7.1, we emulate the result of this computation on approximate DRAM.

Our observation using Valgrind is that the operating system's memory mapping causes the edge-detection program to store its results in different memory pages during different runs. Uniqueness of data placement during different runs, makes stitching possible. This allows Probable Cause to create larger fingerprints of memory by observing different samples using the technique described in Section 4. Furthermore these experiments verified our original assumptions that data is stored in consecutive physical pages in main memory and that it does not get remapped to different physical pages during a single run.

As the number of sample data collected increases, Probable Cause stitches together different fingerprints to create larger system-level fingerprints. Figure 13 presents the relation between number of samples and number of clusters identified by our system using $10MB$ data samples (one photo from a digital camera). Because of lack of overlap, Probable Cause clusters the initial fingerprints as unique chips. As the number of approximate outputs observed increases, Probable Cause is able to use overlaps to stitch fingerprints together, decreasing the number of suspected chips. In our experiment, Probable Cause was able to begin fingerprint convergence after approximately 90 samples.

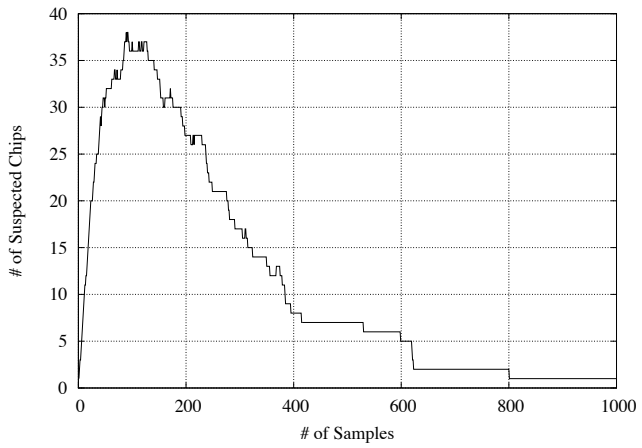

Figure 13: Number of distinct fingerprints generated from a chip of size $1GB$ based on collected samples of size $10MB$ for our edge detection program. As the number of samples increase, Probable Cause is able to connect different partial fingerprints together to create a single system-level fingerprint.

8. Discussion

After presenting the design and evaluation of Probable Cause, there are three issues that require more in-depth discussion. First, for controllability reasons, our evaluation uses DRAM chips two decades past their prime. Do our results hold for more recent DRAM technologies? Second, what are possible defenses against Probable Cause? Third, all of the results in the paper assume that the error locations in an approximate output are known. How can an attacker identify potential error bits from the approximate output alone?

8.1. Effect of DRAM technology

To verify that Probable Cause is not limited to the dated DRAM that we use in our evaluation platform, we construct an FPGA-based platform that uses DDR2 memory. While it is possible to confirm all of our results using this DDR2 platform, running all of our experiments on this platform is cost and time prohibitive. Due to this fact and the similarity of the results, we limited the presentation in Section 7 to the older DRAM platform. Here, we cover the DDR2 platform.

Our DDR2 platform consists of a Xilinx Virtex-5 FPGA with an altered soft-core memory controller. We alter the memory controller to expose an automatic refresh disable signal to the software layer. For memory, we use a Micron MT4HTF3264HY 256MB DDR2 DRAM chip [21]. To control everything, we implement an OR1200-based System-on-Chip [24] on the FPGA. To avoid contaminating program code and data, we add a scratchpad memory to the FPGA fabric that we use as the program's main memory.

We port the MSP430 test code to the OR1200 and run with the same levels of approximation and temperatures that we use in Section 7. The results of these experiments show that, as in the older DRAM, the spatial distribution of volatility is robust to both temperature changes and different levels of approximation. We do notice that the probability distribution of cell volatilities in the DDR2 chip is skewed toward higher volatility where the older DRAM had no skew. While our analysis shows that this difference does not impact the clustering or classification abilities of Probable Cause, it could mean that it is harder to fine-tune the desired level of approximation on DDR2-based systems.

8.2. Defenses against Probable Cause

Probable Cause leverages a side channel that allows an attacker to correlate approximate data to its origin. In this section, we examine three possible methods to protect users against Probable Cause.

8.2.1. Data segregation One possible defense is to to separate sensitive data and general data in memory. This approach suffers from three major drawbacks:

1. It relies on user intervention to identify sensitive data.
2. It does not provide either backward or forward secrecy: there is no way to take back approximate outputs or to change how approximation affects future outputs.
3. It sacrifices system resources by segregating how much memory system can use based on its privacy requirements.

8.2.2. Noise Addition of noise is one of the main approaches researchers use to counteract side-channels [16]. Defending against Probable Cause using this approach requires addition of random noise to the data which further degrades the accuracy of the results. This trade-off is undesirable for a system designer, because it imposes heavy penalties both on possible energy and computational time savings, while deteriorating output quality. Accumulating noise through movement of data in approximate memory also suffers from the same shortcomings. In the end, adding noise only slows the attacker down.

8.2.3. Data scrambling Page-level Address Space Layout Randomization (ASLR) can prevent Probable Cause from deanonymizing data by preventing the stitching of page-level fingerprints into system-level fingerprints. If the granularity of ASLR is at most the size of the smallest fingerprint (*e.g.,* page size for our system), there will be no overlap for Probable Cause to detect. This reduces Probable Cause's classification and clustering accuracy and forces it to flag any page-level fingerprint as a potential match if it was within the threshold distance of any chunk of system-level fingerprint. This can result in an increase in false positives as it makes random matches more likely. Using page-level ASLR comes at the cost of a significant increase memory management overhead.

8.3. Error localization

There are multiple approaches that an attacker can use to estimate the precise outputs based on an approximate output. In scenarios were the output is the result of a computation on known inputs, an attacker can recalculate the exact outputs from the inputs. Another approach has the attacker leveraging

the white Gaussian noise properties of the error due to DRAM approximation. An attacker can use one of various noise detection algorithms to detect potential bit error locations. A final approach works in-conjunction with the previous two approaches. It is possible for an attacker to perform speculative distance calculations to see if any case produces a distance below the threshold with one of existing system-level fingerprints. Probable Cause can leverage any of these techniques to detect potential error patterns in approximate outputs and reconstruct an exact version.

9. Related Work

This section frames Probable Cause with respect to previous research on using physical attributes of digital devices as an system identifying side channel and highlights recent works on approximate memory.

9.1. Analog artifacts in a digital World

Previous research has shown that it is possible to identify image and video recording devices using sensor noise [19, 15] and pixel defects [7]. These works are similar to Probable Cause in that they both exploit stable analog properties that are imprinted on outputs to identify devices, but Probable Cause has greater potential impact on users as it can operate on any output stored in main memory—including data coming from analog sensors.

Manufacturing variations of volatile memory have been suggested as a type of Physical Unclonable Function (PUF) for chip identification. FERNS [11] introduced power-up SRAM state as a method of system identification and also a source of random numbers. Recent work by Rosenblatt et al. [29] extends the idea of FERNS to DRAM to create a DRAM PUF. Like our work, Rosenblatt et al. use the variability in DRAM cell decay times and their spatial stability as the basis for their DRAM PUF. Although the underlying physical mechanism used in a DRAM PUF and Probable Cause are the same, the goals of a PUF and our system are at odds: PUFs use intentional manipulation of digital components for attestation while our work shows how manipulations aimed at achieving approximation create a side channel that unintentionally attests for the machine. Additionally, PUFs rely on complete characterization of DRAM, while our experiments show that it is possible to identify a system by capturing approximate results and stitching them together to form a device fingerprint.

Besides using DRAM cell decay time variation for system identification or random number generation, in the Cold-Boot attack [9], researchers exploit the ability to control decay time through temperature variation to maintain state in DRAM while it is transported between a victim machine and an attacker's machine. This allows attackers to search the victim's DRAM for secret keys in an offline manner. Using the same mechanism as the Cold Boot attack, but swapping controlled and uncontrolled variables is TARDIS [28]. TARDIS is a time keeping scheme for security protocols that uses the relationship between the amount of data decayed in SRAM memory and the amount of time the SRAM has been in a powered-off state to track the amount a time a device has been powered off.

9.2. Approximate memory

Approximate memory is a well studied concept in the field of approximate computing. Esmaeilzadeh et al. [6] proposed a general hardware structure for approximate programming with approximate memory as one of the main components. EnerJ [32] is a model for allowing programs to use both approximate and exact variables safely in the same program.

Various works have proposed energy saving schemes targeting main memory. Most approaches control DRAM refresh rate to save power. The driving insight behind these works is that the refresh rate is set based-upon the fastest decaying memory cell—an outlier. Flikker [18], partitions memory into high-refresh and low-refresh zones and stores error-tolerant data in the low-refresh zone. RAPID [40] ranks and populates memory locations by their data retention time and sets DRAM's refresh rate based on the worst retention time of the populated memory locations. Similar to RAPID, RAIDR [17] leverages the idea that adjacent rows have similar retention times to create a unique refresh rate for groups of rows.

Refresh rate is not the only knob available for reducing memory power consumption. David et al. [3] and Deng et al. [4] propose dynamic voltage/frequency scaling to save energy. Half-wits [30] explores the effects of voltage scaling on Flash memory by writing data at a reduced voltage and checking to see if the write succeeded to avoid pumping the charge to a higher voltage and expending more energy. Sampson et al. [31] propose using multi-level non-volatile memory cells as approximate storage using reduced-cost imprecise write operations.

10. Conclusion

In this paper, we expose the deanonymizing aspects of emerging hardware-based approximate computing systems. To deanonymize a host machine, we leverage the observation that each DRAM chip imprints its own unique physical properties in the errors of an approximate result. Our experiments show that it is possible to both identify the host machine that produced an approximate result and to cluster approximate results by host machine. In our experiments, we have 100% success in both host machine identification and clustering using a basic distance metric. This success rate is a product of the two orders-of-magnitude difference in similarity between the error patterns in approximate results produced by the same DRAM chip compared to the approximate results produced by other DRAM chips. Lastly, experiments show that our identification and clustering algorithms are robust against changes in operating conditions, i.e., temperature, and level of approximation.

The ability to reliably identify the host machine that produced an approximate result shows that current DRAM-based

approximate memory systems are not appropriate for situations where the user wishes to preserve their anonymity. To maintain anonymity, future hardware-based approximate computing systems must facilitate exact computation of privacy sensitive data and expose that decision to the user or future research must design anonymity preserving hardware approximation techniques. At a higher level, our results motivate the need for privacy to be a primary design criteria for future approximate computing systems.

Acknowledgments

The authors would like to thank anonymous reviewers and members of the University of Michigan SPQR lab for reviewing early versions of this paper. This research is supported in part by the Center for Future Architectures Research (C-FAR), one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, NSF CSR-1218586, and NSF CNS-0845874. Any opinions, findings, conclusions, and recommendations expressed in these materials are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM international conference on Mobile computing and networking (MobiCom)*, 2008.

[2] K. Cumming, "Purposeful data: the roles and purposes of recordkeeping metadata," *Records Management Journal*, 2007.

[3] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *International Conference on Autonomic Computing*, ser. ICAC, 2011.

[4] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "Memscale: active low-power modes for main memory," *ACM SIGPLAN Notices*, 2011.

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.

[6] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *ACM SIGARCH Computer Architecture News*, 2012.

[7] Z. J. Geradts, J. Bijhold, M. Kieft, K. Kurosawa, K. Kuroki, and N. Saitoh, "Methods for identification of images acquired with digital cameras," in *Enabling Technologies for Law Enforcement*. International Society for Optics and Photonics, 2001.

[8] G. Greenwald, *No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State*. Metropolitan Books, May 2014.

[9] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: cold-boot attacks on encryption keys," *Communications of the ACM*, 2009.

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

[11] D. Holcomb, W. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Proceedings of the Conference on RFID Security*, 2007.

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

[13] JEDEC, "LPDDR2 SDRAM Specification," 2010.

[14] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Symposium on Security and Privacy*, 2005.

[15] K. Kurosawa, K. Kuroki, and N. Saitoh, "CCD fingerprint method-identification of a video camera from videotaped images," in *Proceedings of the International Conference on Image Processing*, 1999.

[16] T.-H. Le, J. Clédière, C. Servière, and J.-L. Lacoume, "Noise reduction in side channel attack using fourth-order cumulant," *IEEE Transactions on Information Forensics and Security*, 2007.

[17] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "Raidr: Retention-aware intelligent dram refresh," in *International Symposium on Computer Architecture*, ser. ISCA, 2012.

[18] S. Liu, B. Leung, A. Neckar, S. O. Memik, G. Memik, and N. Hardavellas, "Hardware/software techniques for DRAM thermal management," *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA)*, 2011.

[19] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, 2006.

[20] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*, 1977.

[21] Micron, *DDR2 SDRAM SODIMM: MT4HTF1664HY – 128MB, MT4HTF3264HY – 256MB, MT4HTF6464HY – 512MB*, 2005.

[22] Micron Technology Inc., "Mobile DRAM Power-Saving Features and Power Calculations," 2005.

[23] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," in *ACM Sigplan Notices*, 2007.

[24] OpenCores.org, "OpenRISC OR1200 processor," http://opencores.org/or1k/OR1200_OpenRISC_Processor.

[25] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity—a proposal for terminology," in *Designing privacy enhancing technologies*. Springer, 2001.

[26] A. C. Polak, S. Dolatshahi, and D. L. Goeckel, "Identifying Wireless Users via Transmitter Imperfections," *Selected Areas in Communications, IEEE Journal on*, 2011.

[27] A. Rahmati, M. Hicks, D. E. Holcomb, and K. Fu, "Refreshing thoughts on DRAM: Power saving vs. data integrity," in *Workshop on Approximate Computing Across the System Stack (WACAS)*, 2014.

[28] A. Rahmati, M. Salajegheh, D. Holcomb, J. Sorber, W. P. Burleson, and K. Fu, "TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks," in *Proceedings of the 21st USENIX Security Symposium*, ser. Security '12, 2012.

[29] S. Rosenblatt, S. Chellappa, A. Cestero, N. Robson, T. Kirihata, and S. S. Iyer, "A Self-Authenticating Chip Architecture Using an Intrinsic Fingerprint of Embedded DRAM," *IEEE Journal of Solid-State Circuits*, 2013.

[30] M. Salajegheh, Y. Wang, K. Fu, A. Jiang, and E. G. Learned-Miller, "Exploiting half-wits: smarter storage for low-power devices," in *9th USENIX Conference on File and Storage Technologies*, 2011.

[31] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate Storage in Solid-State Memories," *IEEE Micro*, 2013.

[32] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," in *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation (PLDI)*, 2011.

[33] Samsung Electronics, "KM41464A NMOS DRAM."

[34] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security (CCS)*, 2004.

[35] Sun Electronic Systems, Inc., *Model EC1X Environmental Chamber User and Repair Manual*, 2011.

[36] Symantec Security Response, "Regin: Top-tier espionage tool enables stealthy surveillance," 2014.

[37] Texas Instruments Inc., "MSP430F241x, MSP430F261x Mixed Signal Microcontroller," in *Texas Instruments Application Report*, Jun. 2007, revised Nov. 2012.

[38] ——, "MSP430 hardware tools," in *Texas Instruments User's Guide*, May. 2009, revised Feb. 2014.

[39] D. Tschumperlé, "The CImg library," in *IPOL Meeting on Image Processing Libraries*, 2012.

[40] R. K. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM," in *The Twelfth International Symposium on High-Performance Computer Architecture (HPCA)*, 2006.