Attacking and Defending Emerging Computer Systems Using the Memory Remanence Effect

by

Amir Rahmati

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2017

Doctoral Committee:

Professor Atul Prakash, Chair
Professor J. Alex Halderman
Professor Peter Honeyman
Professor Vineet Kamat

Amir Rahmati

rahmati@umich.edu

ORCID iD: 0000-0001-7361-1898

To my family,

for their unconditional love and support.

ACKNOWLEDGEMENTS

It is said that every life is a story. If that is the case, my time as a graduate student will be definitely categorized as an adventure. I am in debt to many great people in it.

I thank my advisor, Atul Prakash, for believing and investing in me. His constant support was a definite factor in bringing this dissertation to its completion.

Matthew Hicks, a collaborator, mentor, and friend, whose support and guidance helped me through some of the toughest times in my Ph.D. career.

Members of my thesis committee, J. Alex Halderman, Peter Honeyman, and Vineet Kamat for their insightful suggestions, comments, and support.

I am grateful to Jaeyeon Jung whose deep understanding of the scientific method and problems and solutions in the IoT space taught me valuable lessons on how to conduct systems research. I am equally thankful to Harsha Madhyastha, who helped me expand my knowledge in Distributed Systems and intellectually challenged me to perfect my system designs. Morley Mao, Wayne Burleson, Jacob Sorber, and Dan Holcomb, whose scientific insights helped me throughout my research career.

This journey would have never been the same without my close friend and brother-in-arms, Earlence Fernandes, who always challenged me and pushed through project after project with me, and my labmate and friend, Kevin Eykholt, whose views added new dimensions to our work.

I thank Kevin Fu, who taught me the ABCs of research and helped me start my journey toward my Ph.D. My collaborator and friend Mastooreh Salajegheh, who

mentored me on my first research projects, and all members of the SPQR lab whom I worked with: Benjamin Ransford, Shane Clark, Andres Molina-Markham, Denis Foo Kune, Shane Guineau, Joel Van Der Woude, Aravind Vadrevu, Michael Rushanan, Tim Trippel, Ofir Weisse, and Sai R. Gouravajhala.

I'm also grateful to my friends and colleagues Minghe Yu, Jie Song, Zhongjun Jin, Nikita Bhutani, Shichang Xu, Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, and my fellow Iranians and friends at the University of Michigan and the University of Massachusetts Amherst. Some of them are, Ashkan Nikravesh, Mehrdad Moradi, Hamed Yousefi, Armin Sarabi, Mahdi Aghadjani, Hamed Soroush, Banafsheh Seyed-aghazadeh, and Parya Pourazarm.

Finally, I thank my father, Abdolreza Rahmati, mother, Fariba Khashehchi, and brother, Ahmad for being the constant pillars of support in my life, my extended family, friends at Sharif and Allemeh Helli, and to my dear friends who helped me bring this adventure to its end.

"Our revels now are ended. These our actors,
As I foretold you, were all spirits and
Are melted into air, into thin air:
And, like the baseless fabric of this vision,
The cloud-capp'd towers, the gorgeous palaces,
The solemn temples, the great globe itself,
Yea, all which it inherit, shall dissolve
And, like this insubstantial pageant faded,
Leave not a rack behind. We are such stuff
As dreams are made on, and our little life
Is rounded with a sleep."
— William Shakespeare, The Tempest

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Attacking and Defending Emerging Computer Systems Using the Memory Remanence Effect

by

Amir Rahmati

Chair: Atul Prakash

In computer systems, manufacturing variances and hardware effects are typically abstracted away by the software layer. This dissertation explores how these effects, specifically memory remanence, can be used both as an attack vector and a tool to defend emerging computing systems. To achieve this, we show how time-keeping, anonymity, and authenticity can be affected by memory remanence. In terms of attacks, we explore the deanonymizing effect of approximate computing in the context of approximate memory in Probable Cause. We show how data passing through an approximate memory is watermarked with a device specific tag that points the attacker back to the device. In terms of defenses, we first present TARDIS: an approach to provide a notion of time for transiently powered embedded devices without requiring any hardware modification using remanence effect of SRAM. TARDIS allows these devices to keep a coarse-grained notion of time without the need for a running clock. Second, we propose data retention voltage of memory cells as a new type of physical unclonable function that allows for low-cost authentication and counterfeit resistance in computer systems.

CHAPTER I

Introduction

Memory remanence refers to the residual data or its traces that remain in memory after it is expected to be lost. In volatile memories where the integrity of the data relies on continuous access to a source of energy and/or refresh operations, memory remanence shows itself as the gradual decay of data after loss of power, or errors that may appear due to reduced input voltage or refresh rate in energy saving mechanism. A prominent example of memory remanence effect on security of computer systems was presented by Halderman et al. [42] where they used effect of memory remanence to extract secret cryptographic keys from DRAM memories.

This thesis focuses on security implications and potentials of memory remanence, exploring how various security primitives can be challenged or obtained using its effects. Table 1.1 provides an overview of this work. First, we look at Approximate Computing as an emerging field that seeks to trade computational accuracy for energy and performance, and show how use of approximation in memory can create a privacy risk by watermarking any data passing through the memory with a signature specific to that device. Second, we tackle the problem of time-keeping in transiently powered embedded devices in TARDIS. We show how limited access to energy makes these devices incapable of performing simple security functionalities such as rate-limiting and time-out and show how this capability can be regained in software by taking

System	Type	Security Primitive	Contribution
Probable Cause	Attack	Anonymity	Showing that use of Approximate Computing creates inherent privacy risks
TARDIS	Defense	Time	Providing a notion of time to transiently powered devices
DRV-Fingerprinting	Defense	Authenticity	Creating an unforgeable fingerprint using data retention voltage of memory

Table 1.1: Summary of work

advantage of memory remanence in hardware.

In this dissertation, *we take a principled approach to examine the effects of memory remanence and build attacks and security mechanisms that take advantage of its effects.* We build open-source experimental platforms to implement attacks and security mechanisms. We use these platforms to evaluate these systems and to examine their robustness against physical and environmental factors. Finally, we prototype these systems in real-life scenarios to showcase their effectiveness.

1.1 Contributions of This Dissertation

1.1.1 Probable Cause: Deanonymizing Effect of Approximate Memory

Approximate computing research seeks to trade 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 Probable Cause, 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.

1.1.2 TARDIS: Providing a Notion of Time to Transiently Powered Devices

Lack of a locally trustworthy clock makes security protocols challenging to implement on batteryless embedded devices such as contact smartcards, contactless smartcards, and RFID tags. A device that knows how much time has elapsed between queries from an untrusted reader can better protect against attacks that depend on the existence of a rate-unlimited encryption oracle.

The TARDIS (Time and Remanence Decay in SRAM) helps the system locally maintain a sense of time elapsed without power and without special-purpose hardware. The TARDIS software computes the expiration state of a timer by analyzing the decay of existing on-chip SRAM. The TARDIS enables coarse-grained, hourglass-like timers such that cryptographic software can more deliberately decide how to throttle its response rate. Our experiments demonstrate that the TARDIS can measure time ranging from seconds to several hours depending on hardware parameters. We

address key challenges in implementing a practical TARDIS include compensating for temperature and handling variation across hardware.

Our contributions are (1) the algorithmic building blocks for computing elapsed time from SRAM decay; (2) characterizing TARDIS behavior under different temperatures, capacitors, SRAM sizes, and chips; and (3) three proof-of-concept implementations that use the TARDIS to enable privacy-preserving RFID tags, to deter double swiping of contactless credit cards, and to increase the difficulty of brute-force attacks against e-passports.

1.1.3 DRV-Fingerprinting: Using Data Retention Voltage for Chip Identification

Physical unclonable functions (PUFs) produce outputs that are a function of minute random physical variations. Promoted for low-cost authentication and resistance to counterfeiting, many varieties of PUFs have been used to enhance the security and privacy of embedded systems. To different extents, applications for both identification and authentication require a PUF to produce a consistent output over time. As the sensing of minute variations is a fundamentally noisy process, much effort is spent on error correction of PUF outputs. We propose a new variant of PUF that uses well-understood properties of common memory cells as a fingerprint. Our method of fingerprinting SRAM cells by their data retention voltage improves the success rate of identification by 28% over fingerprints based on power-up state.

CHAPTER II

Probable Cause: Deanonymizing Effect of Approximate Memory

2.1 Introduction

Secure system designers tend to focus on the anonymity of communication [86] and take for granted the hardware used to generate the data communicated. Attribution of data is usually done through communication meta-data [23]. While the use of encryption secures the communication against eavesdroppers, it is unable to hide the occurrence of communication. Anonymity systems such as Tor [26] 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 [115] or non-intrusively using unique characteristics of analog hardware such as RF fingerprinting [17, 87], clock skew [57], or camera sensor noise [71]. 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 accu-

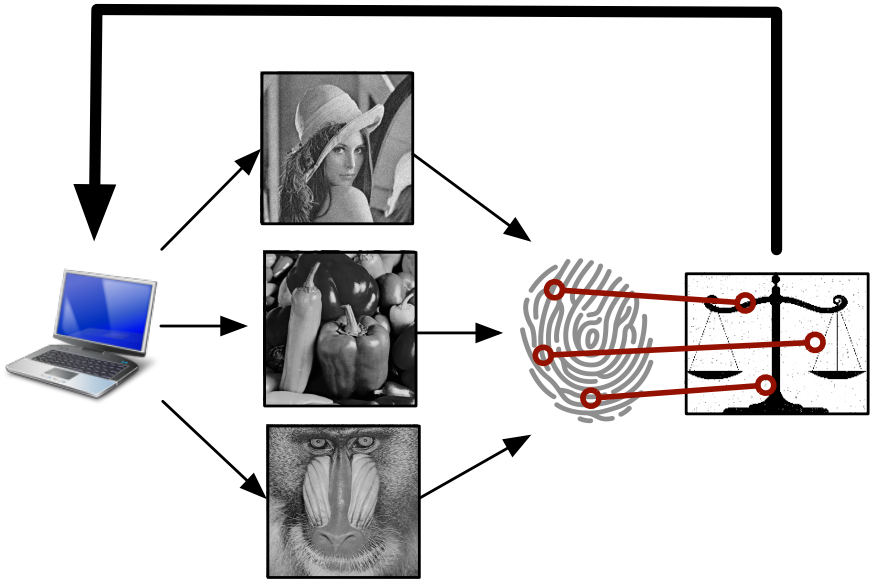


Figure 2.1: Probable Cause creates a fingerprint of an approximate DRAM system by collecting approximate outputs and stitching together error patterns in those outputs to form a fingerprint for the memory. Attackers can then use this memory fingerprint to identify other approximate outputs as belonging to the system.

racy 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 [24] or by decreasing the refresh rate [67, 68, 125]. These techniques are a key component in future approximate computing systems, especially those that tolerate limited errors in data [28].

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 2.1 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.

2.2 Background

Dynamic Random Access Memory (DRAM) is a type of volatile memory that stores values by holding charge in a capacitor. Figure 2.2 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 [53] 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.

DRAM cells decay at different rates, mainly due to their manufacturing variations. The distribution of how quickly DRAM cells decay follows a Gaussian distribution [90]. 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

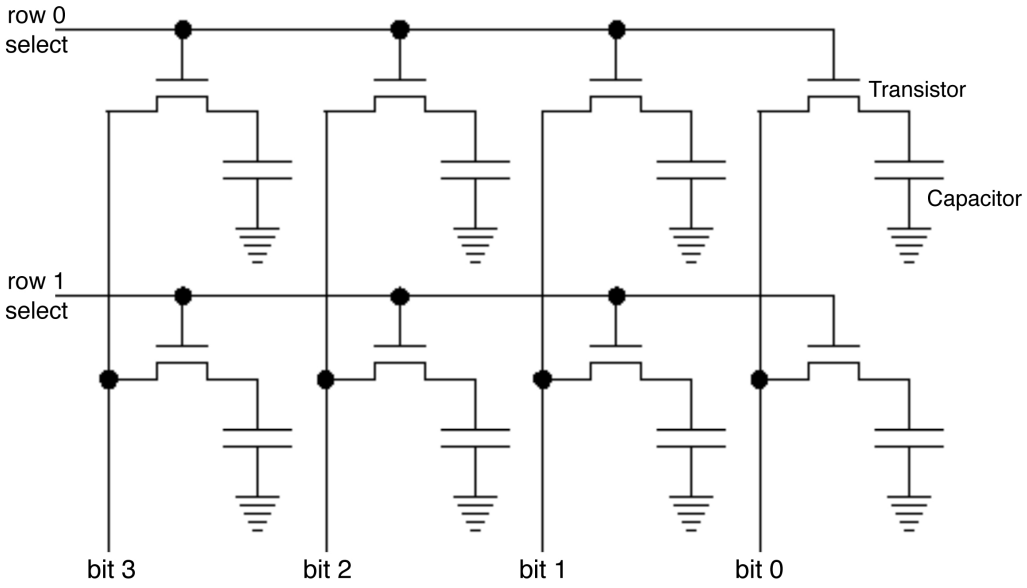


Figure 2.2: A DRAM cell has a default low value that can be changed by charging the capacitor. DRAM cells need to be constantly refreshed for the value to hold, otherwise capacitor leakage slowly reverts the cell to its default value. All DRAM operations are done at row granularity.

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 this chapter, we experimentally show that these orderings are unique, stable given environmental

changes, and stable given the amount of error.

2.3 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) [26]).

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 2.3 depicts two attack scenarios explored in this chapter:

(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.

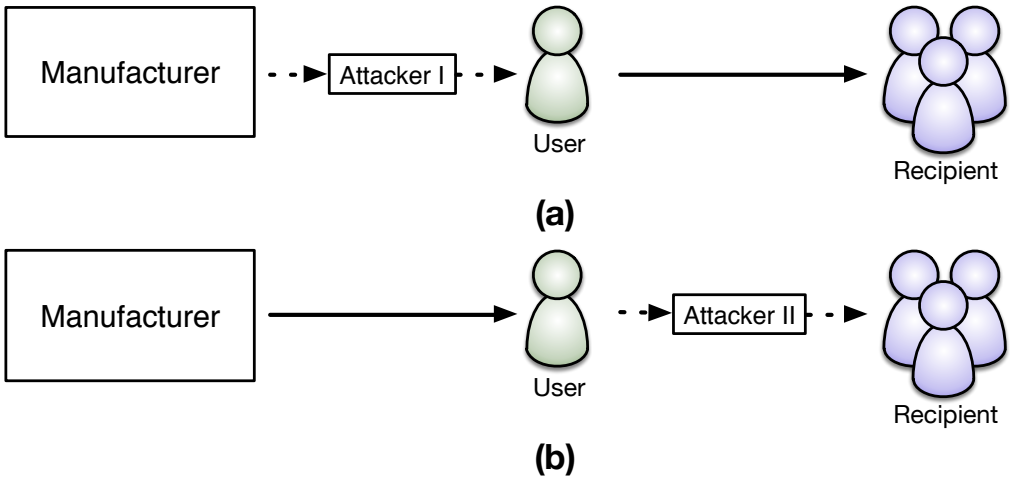


Figure 2.3: Probable Cause tackles two attack scenarios: (a) the attacker intercepts and fingerprints the entire memory (as a part of a system or a standalone module) in the supply chain and (b) the attacker captures approximate outputs from a deployed system to create a fingerprint.

Both the supply-chain attack and eavesdropping attack are feasible given real-world precedents [38].

2.4 Design of Probable Cause

The two scenarios described in Section 2.3 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 2.7.1 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.

For the post-deployment attack scenario, we assume the attacker has access to

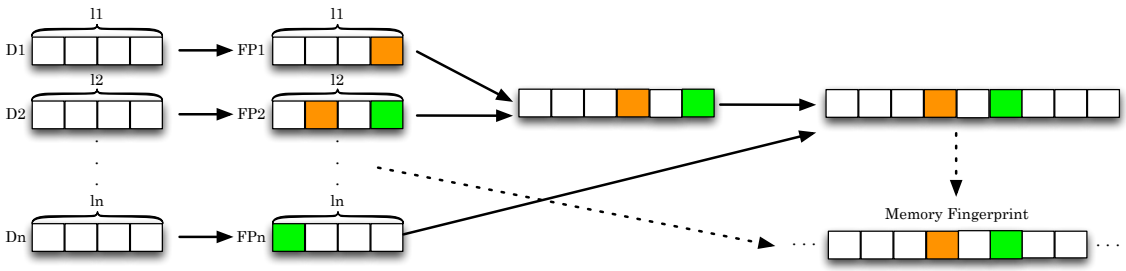


Figure 2.4: Probable Cause constructs the whole-memory fingerprint by stitching together fingerprints of overlapping approximate outputs. Pages of the same color are the same page and are matched by Probable Cause using page fingerprints.

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) [106] 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 2.4 depicts how this process works: initially, Probable Cause creates a fingerprint for every page of data that it sees. Therefore, each approximate output will result in a contiguous series of page-size fingerprints $FP_1, FP_2, ..., FP_n$ with length of $l_1, l_2, ..., l_n$ pages, respectively. We will explain the process of fingerprinting in Section 2.5.2. Next Probable Cause tries to stitch these page-size fingerprints together into a system-level fingerprint by searching for overlap among the series of connected page fingerprints. This algorithm is explained in detail in Section 2.5.3. If the page fingerprints of two approximate outputs match, then there is a range of physical memory pages that held both outputs. Probable Cause uses the page fingerprints outside the overlap region

[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.

to create a combined system-level fingerprint that encompasses the page fingerprints of each output. As the number of outputs increase, more fingerprints are stitched together. In Section 2.7.6 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).

2.5 Mechanics of Probable Cause

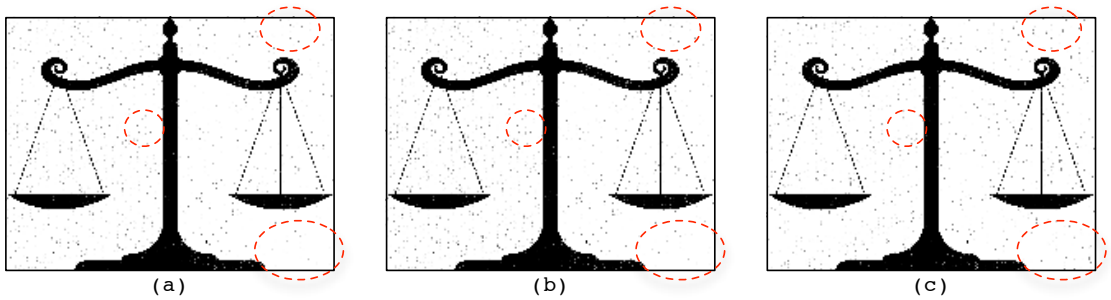


Figure 2.5: Three identical images after storage in approximate memory. Image (c) is stored in a different chip than (a) and (b). Simple visual inspection reveals similar patterns of errors in results coming from the same chip.

Probable Cause's goal is to identify the origin of approximate data based on the error pattern imprinted by approximate DRAM. Figure 2.5 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%

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

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

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

error with worst-case data. Figure 2.5.a and Figure 2.5.b show the image produced by the same chip, but at different temperatures, while Figure 2.5.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 2.5.a and 2.5.b, but no real similarity to Figure 2.5.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 2.5.1 covers generating system-level fingerprints for DRAM chips. Section 2.5.2 covers correlating approximate results and system-level fingerprints. Finally, Section 2.5.3 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.

2.5.1 Characterization

The first step required for Probable Cause to successfully deanonymize a user is characterization. To characterize an approximate memory, Probable Cause needs a

Algorithm 2 Error Marking Algorithm: The algorithm compares approximate result with exact data, returning a bitstring with errors marked as 1.

```
markError(approx[size], exact[size])
1  for i ← 1 to size
2        do if approx[i] ≠ exact[i]
3              do errorString[i] ← 1
4           else  errorString[i] ← 0
5  return errorString
```

series of approximate results. Based on the adversarial model described in Section 2.3, 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 in the form of a bit vector. Algorithm 2 explains this process. 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 classification fast as it takes little time for the first 1% of bits to fail.

2.5.2 Identification

To correctly match an approximate output with a system-level fingerprint, Algorithm 3 first detects errors in the approximate data by comparing it to the exact data.

Algorithm 3 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
```

We assume the availability or calculability of exact value in our threat model. 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 4. The algorithm returns the first system-level fingerprint whose distance to the error pattern in the output is below a pre-defined threshold. Section 2.7 discusses how we experimentally determine this threshold.

Designing a suitable distance metric requires for Probable Cause requires special considerations. Mainly, our distance metric needs to consider cases where the amount of error in the system-level fingerprint and the approximate output differ dramatically (*e.g.,* the chip is fingerprinted 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. This makes simple distance metrics such as Hamming distance unsuitable for our system.

To compensate for this, we designed a custom distance metric (detailed in Algorithm 4 and Algorithm 5) based on Jaccard's index [52]. Equation 2.1 provides the formal definition of the Jaccard index. Our metric looks for errors that exist in the fingerprint, but are absent in output's error pattern[2]. This makes the distance

[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 Distance Algorithm based on Jaccard index [52].

```
DISTANCE(errorString[size], fingerprint[size])
1  Initialize d ← 0
   ▷ Count the number of errors in fingerprint which are absent 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(fingerprint)
```

Algorithm 5 Hamming Weight Algorithm: Counts the the hamming weight of an error string.

```
HAMMINGDISTANCE(errorString[size])
1  Initialize d ← 0
2  for i ← 1 to size
3        do if errorString[i] = 1
4              do d ← d + 1
5  return d
```

metric focus on identifying errors in data which can determine if it originates from a previously fingerprinted memory. This result is then normalized to ranges from $[0, 1]$ by division to the total number of errors in the fingerprint (*i.e.,* the Hamming weight of the fingerprint). Algorithm 5 provides the steps to calculate this value.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

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.

2.5.3 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 2.4. 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 characterization 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 6 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. Note that this algorithm assumes that a page fingerprint is unique. We will provide the mathematical proof behind this assumption in Section 2.7.1.

2.6 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 2.8.1. Our DRAM experiments consist of a set of 10 32KB KM41464A DRAM chips [103]. 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,

Algorithm 6 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
```

- The MSP-FET430UIF [118] JTAG Programmer is responsible for programming the microcontroller and later transferring the results back to the analysis computer.

- The MSP430-F2618 [117] 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 [113] 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 [90].

- 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 2.2 discusses how each DRAM cell has a charged state which corresponds to logical 1 or 0, depending on the row. Using the charged value

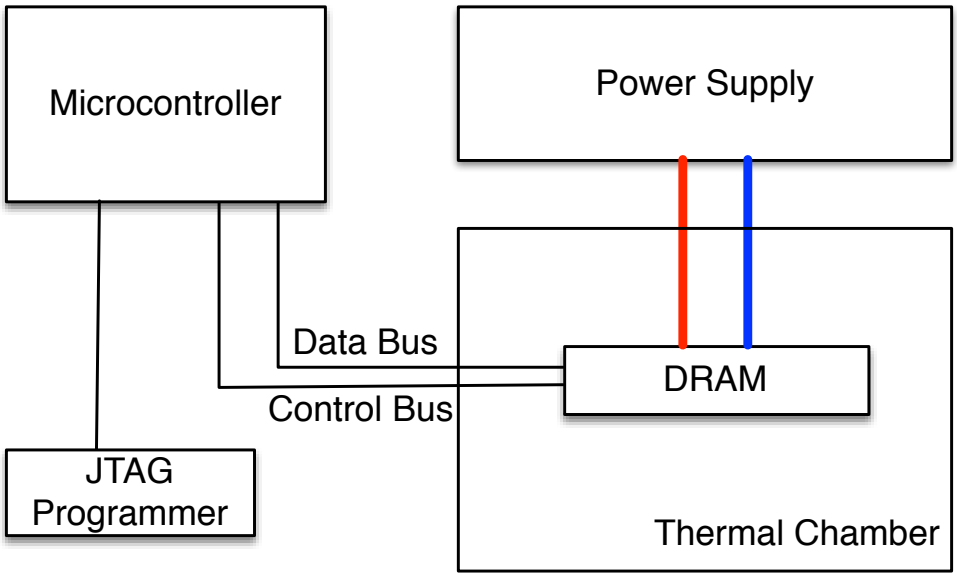


Figure 2.6: Probable Cause experimental platform. The MSP430 microcontroller controls DRAM read/write functions. The target DRAM is placed inside a thermal chamber to ensure environment consistency across experiments. The JTAG programmer allows us to program the microcontroller and extract the results.

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.

2.7 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 2.6. 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.

2.7.1 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 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 4 to calculate the distance between the output and every system-level fingerprint. Figure 2.7 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 3 to trivially deanonymize chips from their approximate data.

Uniqueness can also be evaluated theoretically by reasoning about the space of

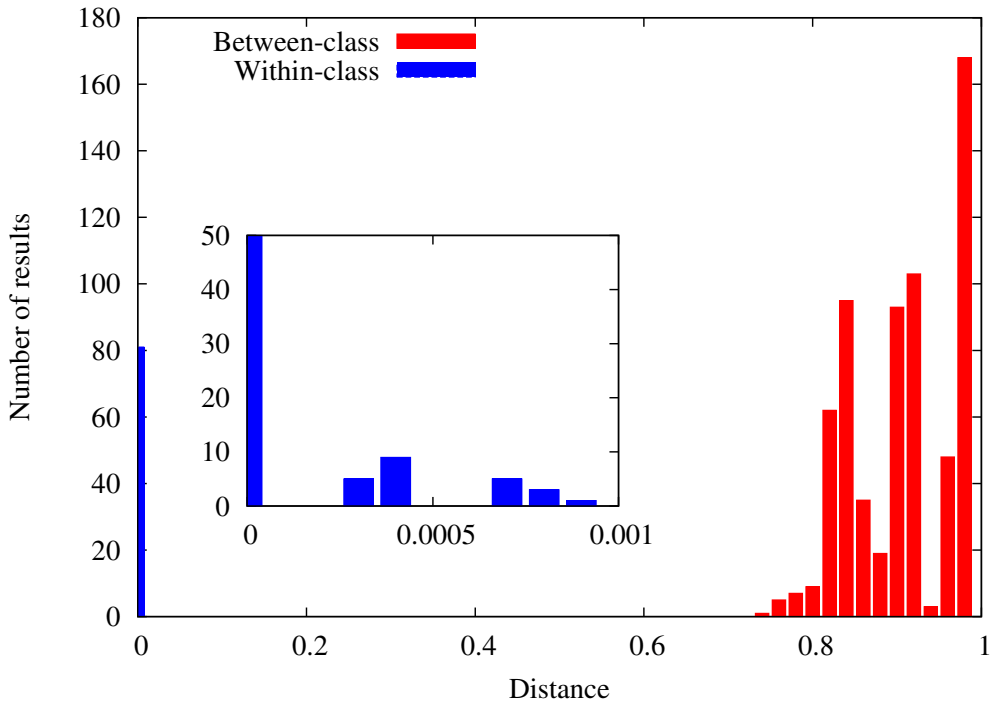


Figure 2.7: Histogram of fingerprint distances for within-class (same chip) and between-class (other chips) pairings.

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 2.2.

$$Max\ unique\ fingerprints = \binom{M}{A} \tag{2.2}$$

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

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 2.1: Results for a page of memory

calculated using the Hamming bound [72]:

$$\frac{\binom{M}{A}}{\sum_{i=0}^{2T}\binom{M}{i}} \leq Max\ distinguishable\ fingerprints \leq \frac{\binom{M}{A}}{\sum_{i=0}^{T}\binom{M}{i}} \tag{2.3}$$

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

$$\frac{\sum_{i=1}^{T}\binom{M}{i}}{\binom{M}{A}} \leq Chance\ of\ mismatching \leq \frac{\sum_{i=1}^{2T}\binom{M}{i}}{\binom{M}{A}} \tag{2.4}$$

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 2.5.

$$entropy/bit \geq \frac{\log_2\left(\frac{\binom{M}{A}}{\sum_{i=0}^{2T}\binom{M}{i}}\right)}{M} \geq \frac{\log_2\binom{M}{A-T}}{M} \tag{2.5}$$

To put these equations into perspective, Table 2.1 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.

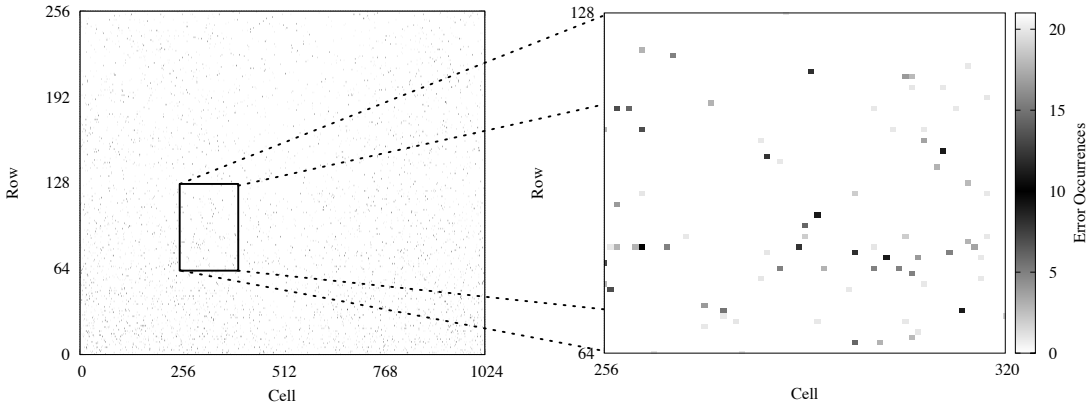


Figure 2.8: Heatmap of cells unpredictability in a sample DRAM chip. Darker cells behave more like noise. More than 98% of cells behave reliably across all 21 runs.

2.7.2 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 2.8 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.

2.7.3 Thermal effect

Temperature variation is known to have a significant effect on the rate of charge decay in DRAM [44]. DRAM refresh rates account for this either by assuming a worst-case operating environment [53] or by dynamically adjusting the refresh rate to compensate for environmental changes while keeping current consumption minimized [77]. Our approximate DRAM implementation similarly adjusts its refresh rate to maintain a desired accuracy across changes in temperature. To explore

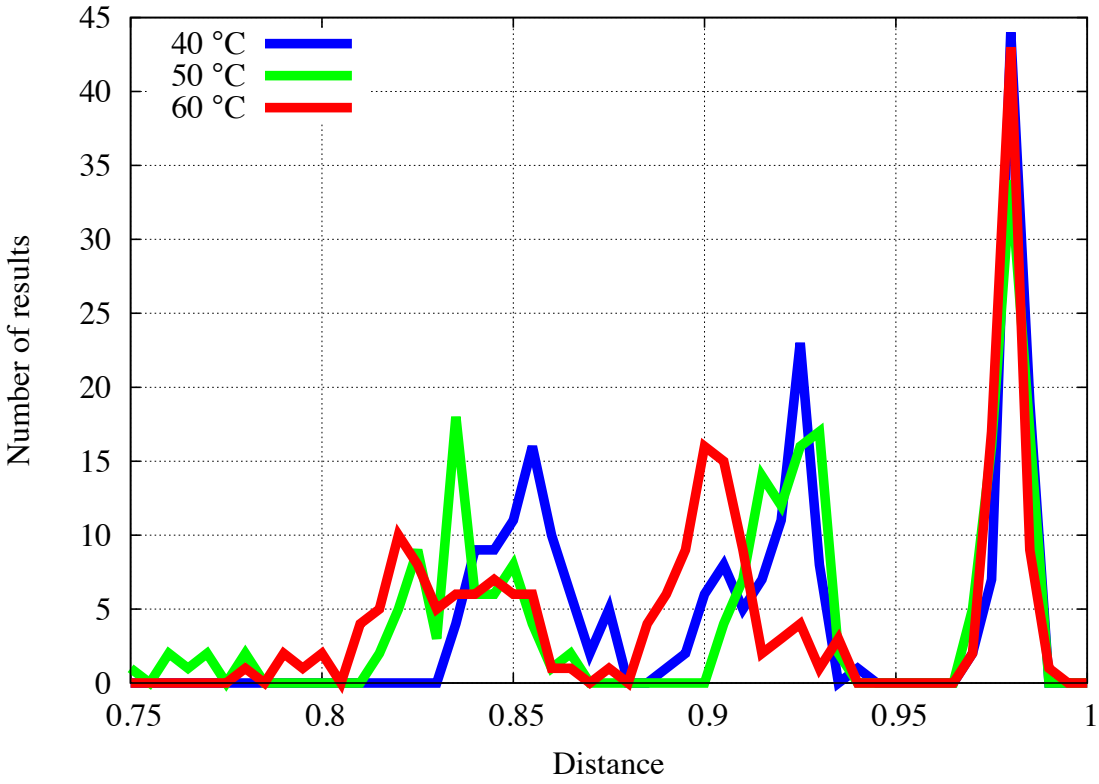


Figure 2.9: Histogram of between-class (different chips) pair distances grouped by temperature. Temperature has no noticeable effect on distance.

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 2.9 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.

2.7.4 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

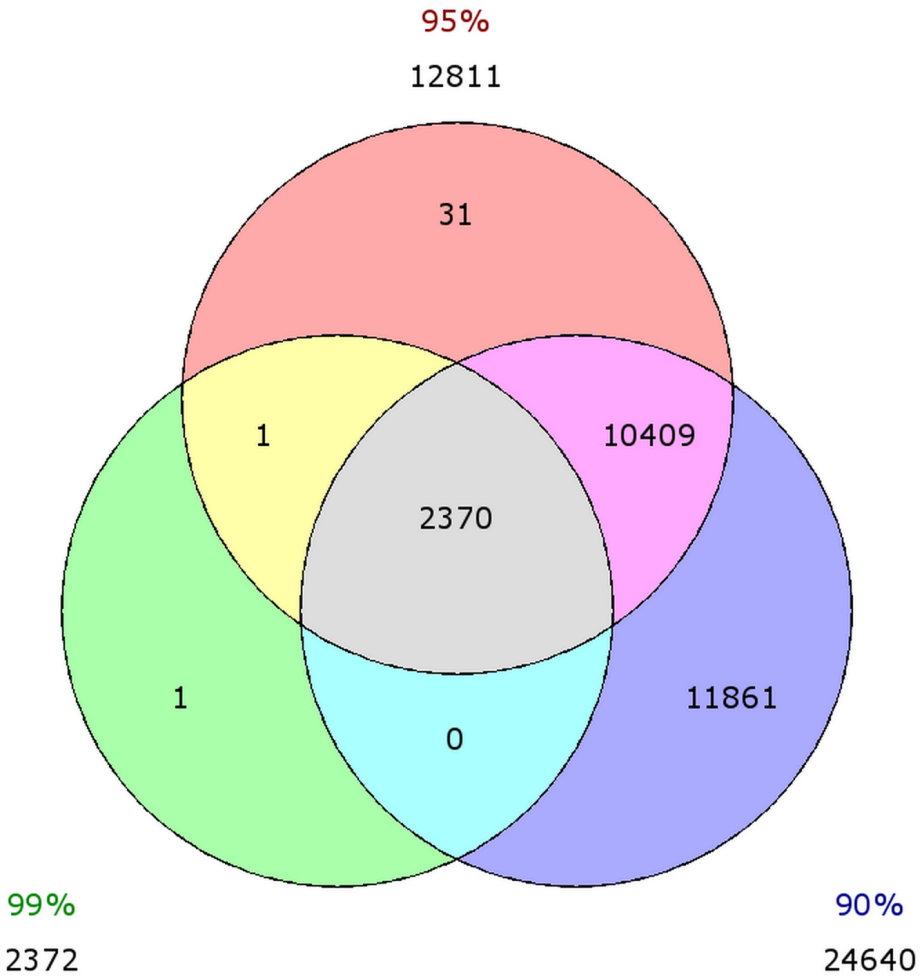


Figure 2.10: Overlap of a DRAM error locations at different levels of approximation. The results support a rough subset relation 99% $\subset$ 95% $\subset$ 90%.

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 2.10 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.

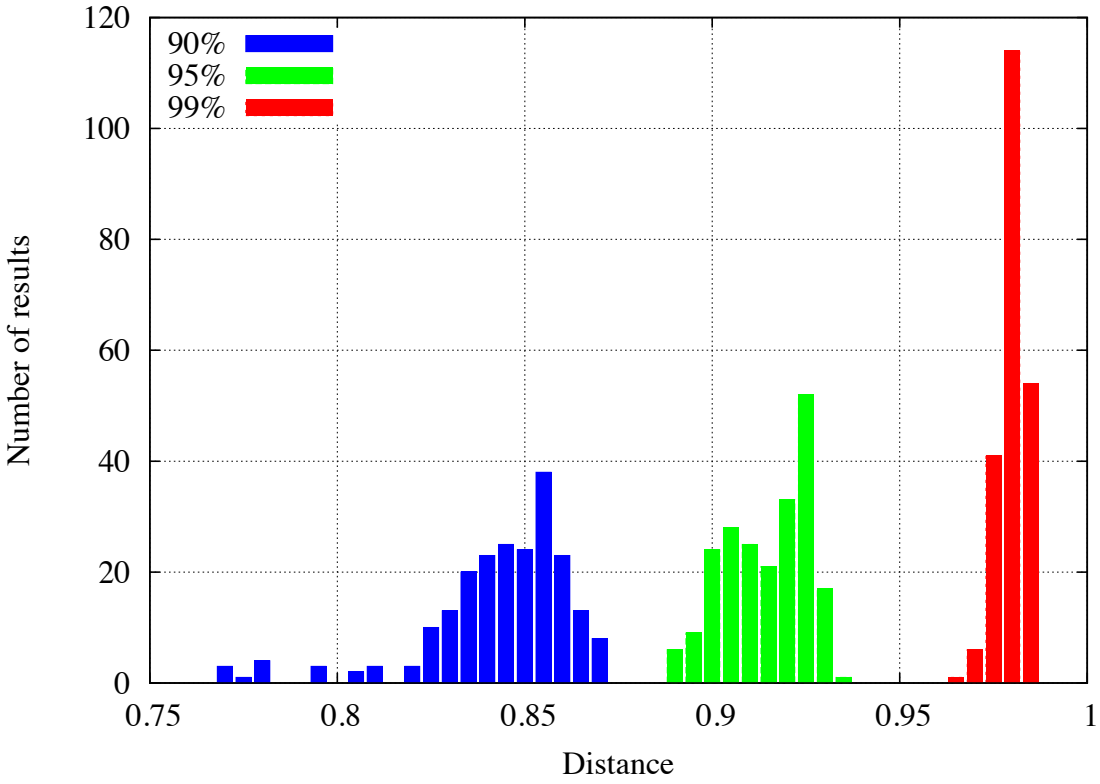


Figure 2.11: Histogram of between-class chip distance grouped by approximate memory accuracy. The increased chance of bit error overlap causes the average distance to shrink with increases in approximation. Note that these distances are still two orders larger than the largest within-class distance.

2.7.5 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 2.7.1, lowering the accuracy is expected to result in an exponential increase in the fingerprint state space—making a misclassification exponentially more likely. Table 2.2 presents the maximum chance of mismatch at different accuracies for a page of memory.

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

We also evaluate the effect of varying accuracies on our distance metric. Figure 2.11 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.

2.7.6 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 [78] instrumented edge-detection program from the CImg open-source image processing library [121]. Figure 2.12 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 2.7.1, we emulate the result of this computation on approximate DRAM.

Our observation using Valgrind is that the operating system's memory mapping

Figure 2.12: Sample input (left) and output (right) of CImg gradient edge-detection code used to evaluate Probable Cause.

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 2.4. 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 2.13 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.

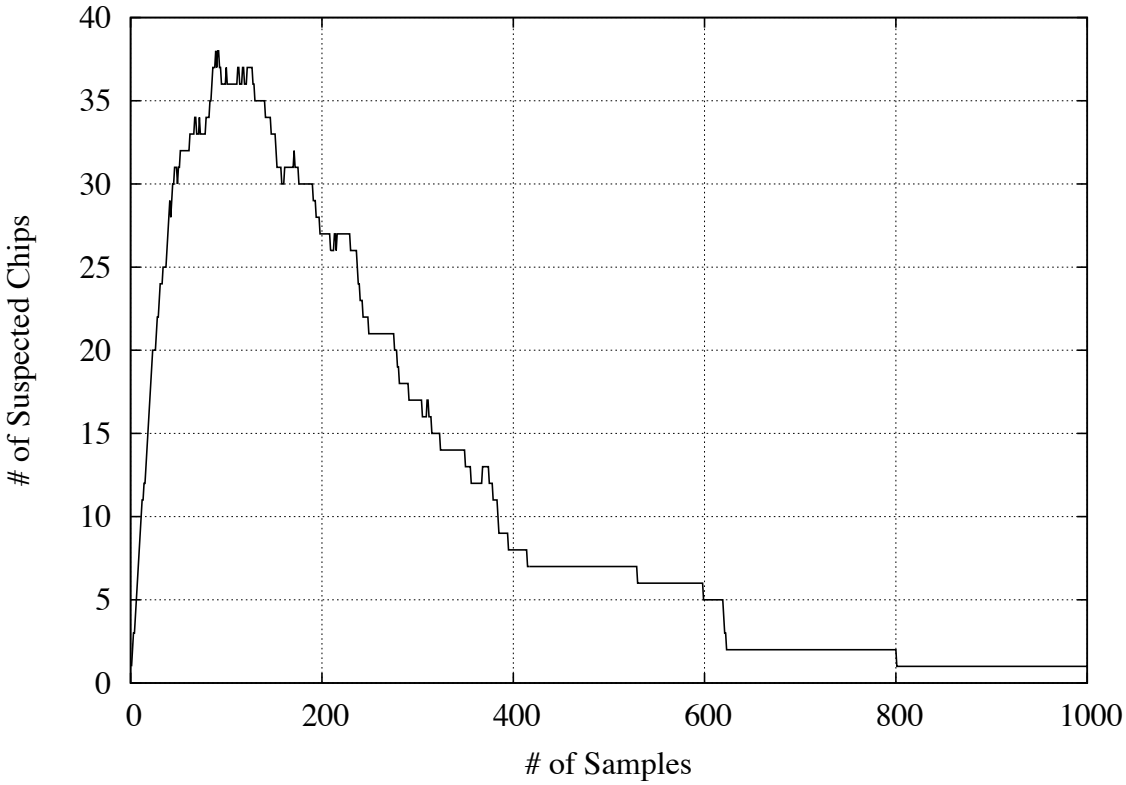


Figure 2.13: Number of distinct fingerprints generated from a chip of size $1GB$ based on collected samples of size $10MB$ for our edge detection program. As the number of samples increase, Probable Cause is able to connect different partial fingerprints together to create a single system-level fingerprint.

2.8 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 this chapter assume that the error locations in an approximate output are known. How can an attacker identify potential error bits from the approximate output alone?

2.8.1 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 2.7 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 [76]. To control everything, we implement an OR1200-based System-on-Chip [83] 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 2.7. 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.

2.8.2 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.

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.

Noise

Addition of noise is one of the main approaches researchers use to counteract side-channels [60]. 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.

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.

2.8.3 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.

2.9 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.

2.9.1 Analog artifacts in a digital World

Previous research has shown that it is possible to identify image and video recording devices using sensor noise [71, 58] and pixel defects [36]. 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 [48] introduced power-up SRAM state as a method of system identification and also a source of random numbers. Recent work by Rosenblatt et al. [96] 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 [42], 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 [91]. 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.

2.9.2 Approximate memory

Approximate memory is a well studied concept in the field of approximate computing. Esmaeilzadeh et al. [28] proposed a general hardware structure for approximate programming with approximate memory as one of the main components. EnerJ [102] 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 [68], partitions memory into high-refresh and low-refresh zones and stores error-tolerant data in the low-refresh zone. RAPID [125] 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 [67] 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. [24] and Deng et al. [25] propose dynamic voltage/frequency scaling to save energy. Half-wits [99] 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. [101] propose using multi-level non-volatile memory cells as approximate storage using reduced-cost imprecise write operations.

2.10 Conclusion

In this chapter, 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.

CHAPTER III

TARDIS: Providing a Notion of Time to Transiently Powered Embedded Devices

3.1 Introduction

> "Timestamps require a secure and accurate
> system clock—not a trivial problem in itself."
> –Bruce Schneier, *Applied Cryptography* [105]

Even a perfect cryptographic protocol can fail without a trustworthy source of time. The notion of a trustworthy clock is so fundamental that security protocols rarely state this assumption. While a continuously powered computer can maintain a reasonably accurate clock without trusting a third party, a batteryless device has no such luxury. Contact smartcards, contactless smartcards, and RFIDs can maintain a locally secured clock during the short duration of a power-up (e.g., 300 ms), but not after the untrusted external reader removes power.

It's Groundhog Day! Again.

Unawareness of time has left contactless payment cards vulnerable to a number of successful attacks (Table 3.1). For instance, Kasper et al. [85] recently demonstrated how to extract the 112-bit key from a MIFARE DESFire contactless smartcard (used

Platform	Attack	#Queries
MIFARE Classic	Brute-force [32]	≥1,500
MIFARE DESFire	Side-channel [85]	250,000
UHF RFID tags	Side-channel [84]	200
TI DST	Reverse eng. [15, 14]	∼75,000
GSM SIM card	Brute-force [37]	150,000

Table 3.1: Practical attacks on intermittently powered devices. These attacks require repeated interactions between the reader and the device. Throttling the reader's attempts to query the device could mitigate the attacks.

by the Clipper all-in-one transit payment card[1]). The side channel attack required approximately 10 queries/s for 7 hours. Some RFID credit cards are vulnerable to replay attacks because they lack a notion of time [45]. Oren and Shamir [84] show that power analysis attacks on UHF RFID tags can recover the password protecting a "kill" command with only 200 queries. At USENIX Security 2005, Bono et al. [14] implemented a brute-force attack against the Texas Instruments Digital Signature Transponder (DST) used in engine immobilizers and the ExxonMobile SpeedPass™. The first stage of the attack required approximately 75,000 online "oracle" queries to recover the proprietary cipher parameters [15].

A batteryless device could mitigate the risks of brute-force attacks, side-channel attacks, and reverse engineering by throttling its query response rate. However, the tag has no access to a trustworthy clock to implement throttling. A smartcard does not know whether the last interrogation was 5 seconds ago or 5 days ago.

Enter the TARDIS.

To enable security protocols on intermittently powered devices without clocks, we propose Time and Remanence Decay in SRAM (TARDIS) to keep track of time without a power source and without additional circuitry. The TARDIS relies on the behavior of decaying SRAM circuits to estimate the duration of a power failure (Figure 3.1). Upon power-up, the TARDIS initializes a region in SRAM of an intermittently powered

[1]No relation to the Clipper Chip [63].

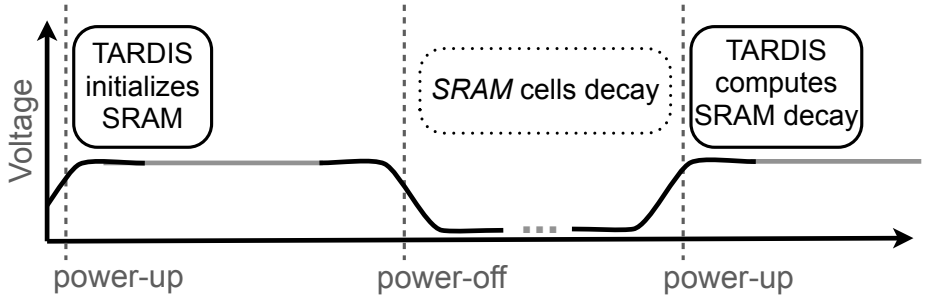


Figure 3.1: TARDIS estimates time by counting the number of SRAM cells that have a value of zero in power-up (*computes SRAM decay*). Initially, a portion of SRAM cells are set to one (*initializes SRAM*) and their values decay during power-off. The dots in the power-off indicate the arbitrary and unpredictable duration of power-off.

device. Later, during power-off, the SRAM starts to decay. Upon the next power-up, the TARDIS measures the fraction of SRAM cells that retain their state. In many ways, TARDIS operation resembles the functioning of an hourglass: the un-decayed, decaying, and fully decayed stages of SRAM are analogous to full, emptying, and empty hourglass states.

Contributions.

Our primary contributions are:

- Algorithmic building blocks to demonstrate the feasibility of using SRAM for a trustworthy source of time without power.
- Empirical evaluation that characterizes the behavior of SRAM-based timekeeping under the effects of temperature, capacitance, and SRAM size.
- Enabling three security applications using SRAM-based TARDIS: sleepy RFID tags, squealing credit cards, and forgiving e-passports.

State of the Art.

Today, batteryless devices often implement monotonically increasing counters as a proxy for timekeeping. RFID credit cards occasionally include transaction counters to defend against replay attacks. Yet the counters introduce vulnerabilities for denial

of service and are difficult to reset based on time elapsed; one credit card ceases to function after the counter rolls over [45]. While one can maintain a real-time clock (RTC) with a battery on low-power mobile devices [97], batteryless platforms do not support RTCs across power failures [80, 100, 18] because of the quiescent current draw.

While a timer of just a few seconds would suffice to increase the difficulty of brute-force attacks (Table 3.1), our experimental results indicate that an SRAM timer can reliably estimate the time of power failures from a few seconds up to several hours. For example, using a 100 μF capacitor at room temperature, the TARDIS expiration time can exceed 2 hours of time. We evaluate the energy and time overhead of the TARDIS, its security against thermal and power-up attacks, and its precision across different platforms.

The primary novelty of the TARDIS is that a moderately simple software update can enable a sought-after security primitive on existing hardware without power. While data remanence is historically considered an undesirable security property [41], the TARDIS uses remanence to improve security. At the heart of the TARDIS are SRAM cells, which are among the most common building blocks of digital systems. The ubiquity of SRAM is due in part to ease of integration: in contrast with flash memory and DRAM, SRAM requires only a simple CMOS process and nominal supply voltage.

3.2 Intermittently Powered Devices: Background, Observations, and Challenges

New mobile applications with strict size and cost constraints, as well as recent advances in low-power microcontrollers, have given rise to a new class of intermittently powered device that is batteryless and operates purely on harvested energy. These

	SRAM	DRAM
Purpose	Fast local memory	Large main memory
Location	Usually on-chip w/ CPU	Usually off-chip
Applications	CPU caches, microcontrollers	PC, notebooks, servers
Storage technology	Cross-coupled transistors	Capacitors
Normal operation	Constantly powered	Intermittently refreshed
Decay state	50% zero/one bits	All zero bits

Table 3.2: Because CPUs of embedded devices generally do not have on-chip DRAM, the TARDIS operates on SRAM. SRAM and DRAM differ fundamentally in their manufacture, operation, intended use, and state of decay.

devices—including contact and contactless smart cards and computational RFID tags (CRFIDs) [93, 100, 132, 130]— typically have limited computational power, rely on wireless transmissions from a reader both for energy and for timing information, and lose power frequently due to minimal energy storage. For example, when a contactless transit card is brought sufficiently close to a reader in a subway, the card gets enough energy to perform the requested tasks. As soon as the card is out of the reader range, it loses power and is unable to operate until presented to another reader. Since a tag loses power in the absence of a reader, it doesn't have any estimation of time between two interactions with a reader.

A typical secure communication between a reader and a tag is shown in Figure 3.2. The tag will only respond to the reader's request if it has authenticated itself by correctly answering the challenge sent by the tag. Two problems arise in this scheme:

- The tag is unaware of the amount of time spent by the reader to answer the challenge, so an adversary has an unlimited amount of time to crack a challenge.
- The tag is unaware of the time between two different queries, so an adversary can send a large number of queries to the tag in a short time space. This can make various brute-force attacks possible on these devices.

Traditionally, computing devices have either had a direct connection to a reliable power supply or large batteries that mask disconnections and maintain a constant

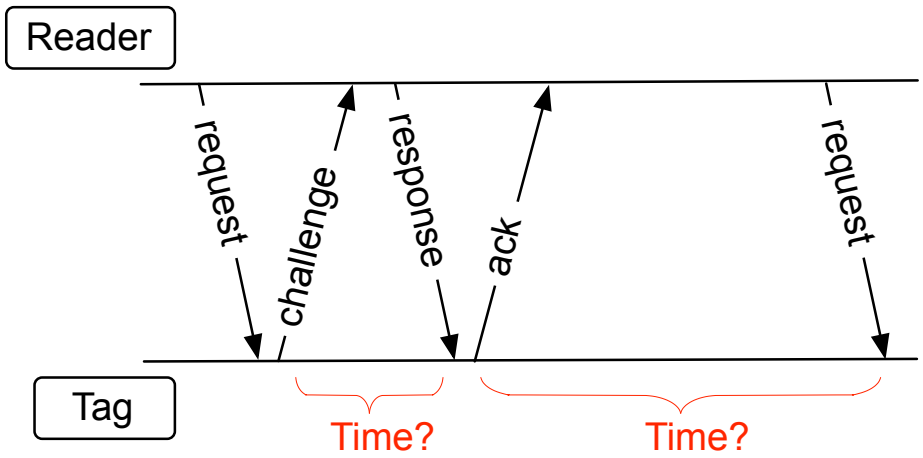


Figure 3.2: The tag cannot determine the time between a challenge and a response or the time between two sessions. The reader could respond to the tag as tardily as it likes or query the tag as quickly as it wants.

supply of power to the circuit. In either case, a reliable sense of time can be provided using an internal clock. Time measurement errors, due to clock drift or power failures, can be corrected by synchronizing with a trusted peer or other networked time source. Current embedded systems address the timekeeping issue in one of the following ways:

1. A system can power a real-time clock (RTC); however, this is not practical on intermittently powered devices due to their tight energy budget. Even if the system uses a low-power RTC (e.g., NXP PCF2123 RTC chip [81]), the RTC component has to be constantly powered (for example, using a battery). This choice also increases the cost of manufacturing and it does not benefit devices that are already deployed.

2. A system can keep time by accessing an external device (e.g., an RFID tag reader) or by secure time synchronization [31, 111]. This option introduces security concerns and may either require significant infrastructure or severely limit range and mobility.

3.2.1 Threat Model and Assumptions

> "...if the attack surface includes an awful lot of clocks that you do not control, then it's worth some effort to try and make your system not depend on them anymore."–Ross Anderson [75]

The primary goal of the adversary in our model is to distort the TARDIS timekeeping. Our threat model considers semi-invasive attacks common to smart cards [32, 85]. We will not discuss attacks such as buffer overflows which are against the systems that would integrate the TARDIS; we focus on the attacks aimed at the TARDIS itself. Our adversarial model considers two classes of attacks: (1) thermal attacks that use heating and cooling [41] to distort the speed of memory decay; and (2) power-up attacks that keep the tag partially powered to prevent memory decay.

3.3 The TARDIS Algorithms

The TARDIS exploits SRAM decay during a power-off to estimate time. An example of the effect of time on SRAM decay in the absence of power is visualized in Figure 3.3. In this experiment, a 100×135 pixel bitmap image of a different TARDIS [1] was stored into the SRAM of a TI MSP430 microcontroller. The contents of the memory were read 150, 190, and 210 seconds after the power was disconnected. The degree of image distortion is a function of the duration of power failure.[2]

Figure 3.1 shows the general mechanism of the TARDIS. When a tag is powered up, the TARDIS initializes a region in SRAM cells to 1. Once the power is cut off, the SRAM cells decay and their value might reset from 1 to 0. The next time the tag is powered up, the TARDIS tracks the time elapsed after the power loss based

[2]The 14.6 KB image was too large to fit in memory, and therefore was divided into four pieces with the experiment repeated for each to get the complete image. The microcontroller was tested in a circuit shown in Figure 3.6 with a 10 μF capacitor at $26°C$. No block transfer computation was necessary.

on the percentage of cells remaining 1. Algorithm 7 gives more details about the implementation of the TARDIS.

MEASURE_TEMPERATURE: To detect and compensate for temperature changes that could affect the decay rate (Section 3.6), the TARDIS uses the on-board temperature sensor found on most microcontrollers. The procedure MEASURE_TEMPERATURE stores inside-the-chip temperature in the flash memory upon power-up. The procedure DECAY calls the TEMPERATURE_ANALYZE function to decide if the temperature changes are normal.

TIME: The TARDIS TIME procedure returns *time* and *decay*. The precision of the *time* returned can be derived from the *decay*. If the memory decay has not started ($decay = 0$), the procedure returns $\{time, 0\}$ meaning that the time duration is less than *time*. If the SRAM decay has started but has not finished yet ($0 \leq decay \leq 50\%$), the return value *time* is an estimate of the elapsed time based on the *decay*. If the SRAM decay has finished ($decay \simeq 50\%$), the return result is $\{time, 50\}$ meaning that the time elapsed is greater than *time*.

ESTIMATION: The procedure ESTIMATE uses a lookup table filled with entries of decay, temperature, and time stored in non-volatile memory. This table is computed based on a set of experiments on SRAM in different temperatures. Once the time is looked up based on the measured decay and the current temperature, the result is returned as *time* by the ESTIMATE procedure. The pre-compiled lookup table does not necessarily need to be calibrated for each chip as we have observed that chip-to-chip variation affects decay only negligibly (Section 3.6).

3.3.1 TARDIS Performance

The two most resource-consuming procedures of the TARDIS are INIT (initializing parts of the SRAM as well as measuring and storing the temperature) and DECAY (counting the zero bits and measuring the temperature). Table 3.3 shows that energy

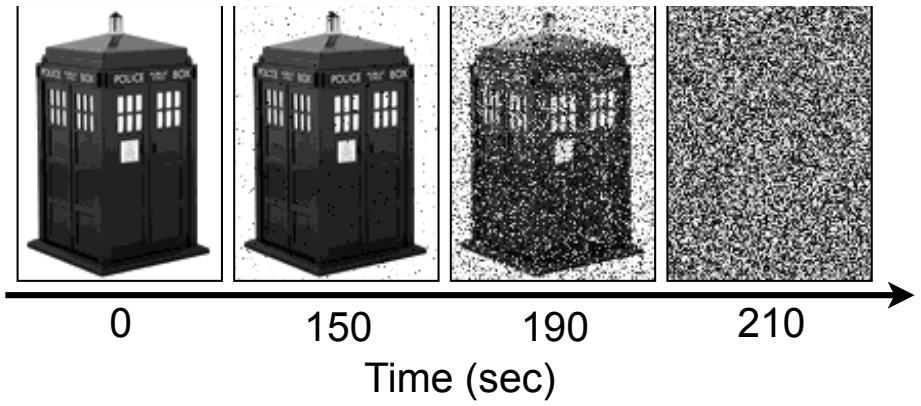


Figure 3.3: Programs without access to a trustworthy clock can determine time elapsed during a power failure by observing the contents of uninitialized SRAM. These bitmap images of the TARDIS [1] represent four separate trials of storing the bitmap in SRAM, creating an open circuit across the voltage supply for the specified time at $26°C$, then immediately returning a normal voltage supply and reading uninitialized SRAM upon reboot. The architecture of a contactless card is modeled using a 10 μF capacitor and a diode in series with the MSP430 microcontroller's voltage supply pin. The degree of decay is a function of the duration of power failure, enabling hourglass-like timekeeping precision without power. No TARDIS was harmed or dematerialized in this experiment.

consumed in total by these two procedures is about 48.75 μJ and it runs in 15.20 ms.

Our experiments of time and energy measurements are performed on Moo RFID[132] sensor tags that use an MSP430F2618 microcontroller with 8 KB of memory, and a 10 μF capacitor. A tag is programmed to perform one of the procedures, and the start and end of the task is marked by toggling a GPIO pin. The tag's capacitor is charged up to 4.5 V using a DC power supply and then disconnected from the power supply so that the capacitor is the only power source for the tag. In the experiments, the DC power supply is used instead of an RF energy supply because it is difficult to disconnect the power harvesting at a precise capacitor voltage. We measured the voltage drop of the capacitor and the GPIO pin toggling using an oscilloscope. The energy consumption of the task is the difference of energy ($\frac{1}{2} \times CV^2$) at the start and end of the task. The reported measurement is the average of ten trials.

Algorithm 7 TARDIS Implementation

```
INIT(addr, size)
1  for i ← 1 to size
2        do memory(addr + i −1) ← 0xFF
3  temperature ← MEASURE_TEMPERATURE()

DECAY(addr, size)
1  decay ← COUNT0S(addr, size)
2  ▷ Proc. COUNT0S counts the number of 0s in a byte.
3  if TEMPERATURE_ANALYZE(temperature)
4  ▷ This procedure decides if the temperature changes are expected considering
5  ▷ the history of temperature values stored in flash memory.
6      then return decay
7      else  return error

EXPIRED(addr, size)
1  decay ← DECAY(addr,size)
2  if (decay ≥ %50 × 8 × size)
3      then return true
4      else  return false

TIME(addr, size, temperature)
1  ▷ Estimate the passage of time by comparing the percentage of decayed bits to
2  ▷ a precompiled table.
3  decay ← DECAY(addr,size)/(8 × size)
4  time ← ESTIMATE(decay,temperature)
5  return {time, decay}
```

3.4 Securing Protocols with the TARDIS

There are many cases where the security of real-world applications has been broken because the adversary could query the device as many times as required for attack. Table 3.1 gives a summary of today's practical attacks on intermittently powered devices. By integrating the TARDIS, these applications could throttle their response rates and improve their security.

We discuss six security protocols that could strengthen their defense against brute-force attacks by using the TARDIS. To demonstrate the ease of integrating

Procedure	Energy Cost	Exec. Time
INIT	$11.53\ \mu J \pm 2.47$	$2.80\ ms \pm 0.0\bar{0}$
DECAY	$37.22\ \mu J \pm 9.31$	$12.40\ ms \pm 1.10$

Table 3.3: Overhead of TARDIS INIT and DECAY procedures measured for TARDIS size of 256 bytes.

the TARDIS, we have implemented and tested three of these security protocols on the Moo, a batteryless microcontroller-based RFID tag with sensors but without a clock [132]. Our prototypes demonstrate the feasibility of the TARDIS and its capabilities in practice.

Sleepy RFID Tags:

To preserve the users privacy and prevent traceability, one could use a "kill" command to permanently deactivate RFID tags on purchased items [55]. However, killing a tag disables many features that a customer could benefit from after purchase. For example, smart home appliances (e.g., refrigerators or washing machines) may no longer interact with related items even though they have RFID tags in them. One could temporarily deactivate RFID tags by putting them to "sleep." However, lack of a simple and practical method to wake up the tags has made this solution inconvenient [55]. By providing a secure notion of time, the TARDIS makes it possible to implement *sleepy tags* that can sleep temporarily without requiring additional key PINs or cryptographic solutions. We consider a time resolution on the order of hours more appropriate for this application.

To extend the sleep time of sleepy tags, one could use a counter along with the TARDIS as follows: upon power-up, the tag checks the TARDIS timer, and it does not respond to the reader if the timer has not expired. If the TARDIS timer has expired, the tag decreases the counter by one and initializes the TARDIS again. This loop will continue while the counter is not zero. For example, using a counter initially set to 1000 and a TARDIS resolution time of 10 seconds, the tag could maintain more

than 2 hours of delay. Since the tag exhausts its counter every time it wakes up, the reader interacting with the tag has to query the tag intermittently.

The TARDIS could prevent yet another attack on Electronic Product Code (EPC) tags that use "kill" commands. To prevent accidental deactivation of tags, a reader must issue the right PIN to kill a tag [27]. An adversary could brute-force the PIN (32 bits for EPC Class1 Gen2 tags). The TARDIS enables the RFID tag to slow down the unauthorized killing of a tag by increasing the delay between queries and responses.

Squealing Credit Cards:

Today, a consumer cannot determine if her card has been used more than once in a short period of time unless she receives a receipt. This is because a card cannot determine the time elapsed between two reads as the card is powered on only when it communicates with the reader. The TARDIS enables a "time lock" on the card such that additional reads would be noticed. Thus a consumer could have some assurance that after exposing a card to make a purchase, an accidental second read or an adversary trying to trick the card into responding would be revealed. *Squealing credit cards* would work similarly to today's credit cards, but they are empowered by the TARDIS to estimate the time between queries and warn the user audibly (a cloister bell) if a second read is issued to the card too quickly. A time lock of about one minute can be considered enough for these applications.

Forgiving E-passports:

RFID tags are used in e-passports to store holder's data such as name, date of birth, biometric ID, and a unique chip ID number. E-passports are protected with techniques such as the Basic Access Control (BAC) protocol, shielding, and passive authentication. However, in practice, e-passports are not fully protected. An adversary can brute-force the BAC key in real time by querying the passport 400 times per minute for a few weeks [8]. Another attack can accurately trace a specific passport by sending hundreds of queries per minute [21].

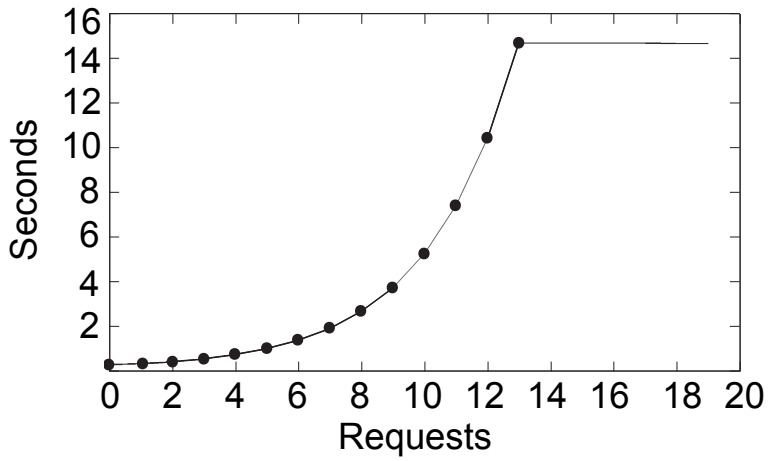


Figure 3.4: Measured response time of a 2010-issued French passport [7]. The passport imposes up to 14 seconds of delay on its responses after unsuccessful execution. The delay will remain until a correct reading happens even if the passport were removed from the reader's field for a long time.

To mitigate the effect of brute-force attacks, French e-passports have implemented a delay mechanism—we imagine using a counter—to throttle the read rate [7]. This delay increases to 14 seconds after 14 unsuccessful attempts (Figure 3.4) and would occur even if the passport was removed from the RF field for several days. Once the tag is presented with an authorized reader, the delay will be enforced and then reset to zero. The TARDIS provides a time-aware alternative that delays unauthorized access but ignores the previous false authentication attempts if the passport has been removed from the reader's range for an appropriate duration. A time duration matching the maximum implemented delay (14 seconds for French passports) would be enough to implement this function.

Passback - Double-tap Prevention:

In mass transportation and other similar card entry systems, the goal of the operator is to prevent multiple people from accessing the system simultaneously using the same card. To achieve this goal, systems are typically connected to a central database that

prevents a card from being used twice in a short time frame.[3] Using the TARDIS, a card could implement delay before permitting re-entry rather than requiring the system to check a central database.

Resurrecting Duckling:

Secure communication in ad-hoc wireless networks faces many obstacles because of the low computing power and scarce energy resources of these devices. Stajano et al. [108] proposed a policy in which these devices would transiently accept a new owner. The devices will later return to an unprogrammed status when the owner no longer needs them, they receive a kill command, or another predefined reset condition is met. Later, others can reclaim and reuse these devices.

For wirelessly powered devices, the TARDIS can provide a sense of time, allowing them to be "reborn" with a new owner only if there is an extended power outage. A legitimate user can continue to power the device wirelessly, but if she wishes to transfer ownership to another entity, she must power it down for a long enough time (defined by the user). Otherwise, the RFID tag refuses to interact with anyone not possessing the present cryptographic key. An example of this application is secure pairing for computational contact lenses [47]. The controller could be cryptographically bound until power disappears for more than a few minutes. Another use of this application is to make stealing SIM cards difficult [37]. The card could refuse to boot if it has been unpowered for a fair amount of time.

Time-out in Authentication Protocols:

Because RFID tags rely on a reader as their source of energy, they cannot measure the delay between a request to the reader and its corresponding response. The tag ignorance gives the reader virtually unlimited time to process the request and response in an authentication algorithm. Having unlimited response time enables the adversary to employ various attacks on the request message with the goal of breaking it. Using

[3]Houston METRO system: `http://www.ridemetro.org/fareinfo/default.aspx`

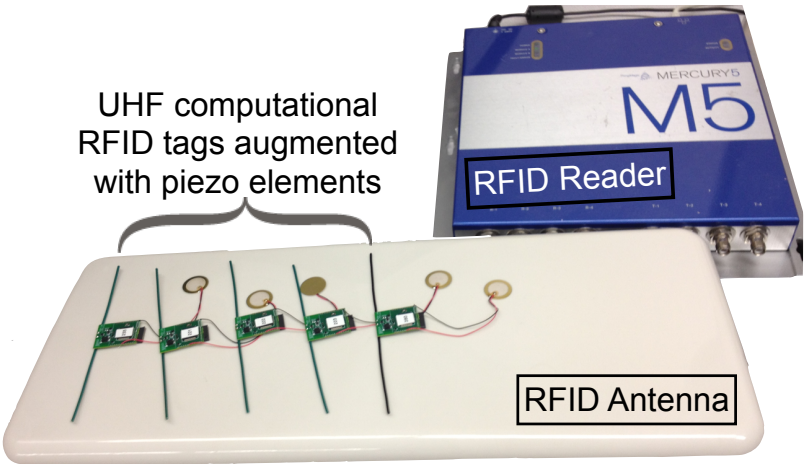


Figure 3.5: Our applications are implemented and tested on the Moo RFID sensors and are remotely powered by a RFID reader (ThingMagic M5 [120]).

the TARDIS will limit the adversary time frame for a successful attack. An example of these protocols can be seen in the e-passport BAC protocol where the reader and passport create a session key for communication. Using The TARDIS would enable passports to enforce expiration of these keys.

3.4.1 Implementation and Evaluation

For the implementation of *sleepy tags*, *squealing credit cards*, and *forgiving e-passports*, we have chosen the Moo, a batteryless microcontroller-based RFID tag. We have augmented this tag with a piezo-element [43] so that it can audibly alert the user to events.

Implementation:

We have implemented a TARDIS library that provides the procedures INIT and EXPIRE listed in Algorithm 7. For the three implemented protocols, a 1-bit precision of time–whether or not the timer had expired–was enough. The programs used for all three protocols are similar and are shown in Algorithm 8. The tag was programmed to call the EXPIRE procedure upon power-up; if the timer had expired, it would respond to the reader and call INIT; otherwise, the tag would buzz its piezo-element. In the case

Algorithm 8 An example of TARDIS usage in a protocol.

```
TARDIS_EXAMPLE(addr, size)
1  if EXPIRED(addr,size)
2     then RESPOND_TO_READER()
3          INIT(addr,size)
4     else BUZZ_PIEZO_ELEMENT()
```

of the *squealing credit cards* protocol the tag was programmed to respond to the reader after buzzing, but for the two other applications, the tag stopped communicating with the reader.

We used a ThingMagic reader [120] and its corresponding antenna to query the tag. When the tag was queried for the first time upon removal from the RF field, it buzzed. The tag stayed quiet whenever it was queried constantly or too quickly.

Experimental Setup:

To measure the TARDIS resolution time on this platform, we powered up the tag to 3.0 V using an external power supply and then disconnected it. We observed the voltage drop over time on an oscilloscope and measured the elapsed time between loss of power and when SRAM decay has finished.[4] We conducted our experiments on five tags, which use a 10 μF capacitor as its primary power source. The TARDIS resolution time on average was 12.03 seconds with a standard deviation of 0.11 seconds. A similar tag, which uses 100 mF, yields a TARDIS resolution time of 145.85 seconds. These time measurements are specific to the platform we have chosen for our experiment. The resolution could potentially be extended to hours using additional capacitors (Table 3.5).

[4]Our experiments (Section 3.6) have shown that SRAM decay finishes when the tag voltage reaches 50 mV.

3.5 Security Analysis

Depending on the application, the adversary may wish either to slow down or to speed up the expiration of the TARDIS. We discuss four different attacks that try to distort the TARDIS interpretation of time.

Cooling Attacks.

An adversary might try to reduce the system's temperature, aiming to slow down the memory decay rate. Other works [41] have used this technique to prevent data decay in DRAM for the purpose of data extraction. Cooling attacks might target the TARDIS timer in cases where the adversary needs to slow the passage of time. As explained in Algorithm 7, the TARDIS measures and records a device's temperature over time and therefore it can prevent cooling attacks by observing unexpected temperature changes.

Heating Attacks.

In contrast to cooling attacks, an attacker might need to speed up the TARDIS timer. For example, someone might try to decrease the delay between queries in order to speed up brute-force attacks. Similarly to the defense against cooling attacks, the TARDIS will report an error indicating unexpected temperature changes.

Pulse Attacks.

A more sophisticated attack is a combination of the cooling and heating attacks such that the temperature would remain the same in the beginning and the end of the attack. It should be noted that this is not a trivial attack because the adversary needs to restore the original internal temperature to prevent the thermal sensor from noticing any difference. A defense against pulse attacks is to implement a thermal fuse [20] on the chip that will activate when the chip is exposed to a high temperature. The activation of this fuse will then either notify the TARDIS of temperature tampering on the next boot-up or possibly prevent the system from booting up at all.

Voltage Control Attack.

Another possible attack scenario would be to power up the system wirelessly to a

minimum voltage that is not sufficient for booting up but sufficient for stopping the memory decay. This would prevent the device from noticing the unauthorized reader and it would stop the memory from decaying further (see Figure 3.8). The voltage control attack can freeze the TARDIS timer at a specific time as long as it sustains the power supply. We imagine that this attack is difficult to implement because of the inherent design of the readers. Many factors (e.g., distance) affect the voltage received by the tags and tags are very sensitive to environmental effects. The readers are also generally designed to flood the targeted environment with energy to provide the tags in range with more than the maximum required power [129]. Excessive power that may have been generated by these devices is then filtered out in tags using voltage regulators. To implement this attack, we imagine the adversary would need to control the input voltage to the tag with a very high precision. If the tag voltage for any reason drops, the SRAM will decay irreversibly. At the same time, the adversary would need to prevent the tags from fully powering up and noticing the unauthorized reader.

3.6 Factors Affecting SRAM Decay

In our evaluation of the TARDIS, we examine the decay behavior of SRAM and three factors that have major effects on this behavior. All experiments use the same circuit (Figure 3.6), and follow the same general procedure.

Experimental Setup:

A microcontroller runs a program that sets all available memory bits to 1. The power is then effectively disconnected for a fixed amount of time (*off-time*). When power is reapplied to the chip, the program records the percentage of remaining 1-bits to measure memory decay, and then it resets all bits to 1 in preparation for the next time power is disconnected. A Data Acquisition (DAQ) unit from Agilent (U2541A series) precisely controls the timing of power-ups and power-downs between 3 and 0 Volts,

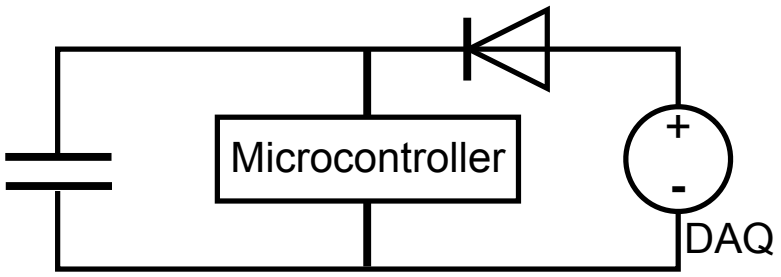


Figure 3.6: General circuit used during the experiments. The microcontroller is held in an environmental chamber to ensure consistent temperature during the tests. The Data Acquisition (DAQ) unit both provides power to the microcontroller and records the voltage decay.

and also measures the voltage across the microcontroller throughout the experiment. An inline diode between the power supply and microcontroller models the diode at the output of the power harvesting circuit in RFIDs; it also prevents the DAQ from grounding VCC during the off-time when the DAQ is still physically connected but is not supplying power. In all experiments, microcontrollers from the TI MSP430 family are used to ensure maximum consistency. The microcontroller used in all experiments is MSP430F2131 with 256 B of SRAM unless stated otherwise.

In all of the experiments, temperature is controlled by conducting all tests inside of a Sun Electronics EC12 Environmental Chamber [112] capable of creating a thermally stable environment from $-184°C$ to $+315°C$ with $0.5°C$ precision. We use an OSXL450 infrared non-contact thermometer [82] with $\pm 2°C$ accuracy to verify that our microcontroller has reached thermal equilibrium within the chamber before testing. For all the experiments, we have collected at least 10 trials.

Defining Stages of Decay:

Three distinct stages of decay are observed in all experiments. Figure 3.7 illustrates the three stages of SRAM decay measured on a TI MSP430F2131 with 256 B of SRAM and a 10 μF capacitor, at $26°C$. We vary the *off-time* from 0 to 400 seconds in 20-second increments. In the first stage, no memory cells have decayed; during

Term	Definition
SRAM Decay	Change of value in SRAM cells because of power outag.e
Decay Stage 1	Time before the first SRAM cell decays.
Decay Stage 2	Time between the decay of first SRAM cell and last one.
Decay Stage 3	Time after the last SRAM cell decays
Ground State	The state that will be observed in an SRAM cell upon power-up, after a very long time without power.
DRV	Data Retention Voltage, minimum voltage at which each cell can store a datum.
DRV Probability(v)	Probability that a randomly chosen cell will have a DRV equal to v and a written state that is opposite its ground state.

Table 3.4: Definition of the terms used to explain the behavior of SRAM decay and the theory behind it.

the second stage, a fraction of the cells, but not all, have decayed; by the third stage the cells have decayed completely (see Table 3.4 for a summary of term definitions). Observations made during Stages 1 or 3 provide a single bit of coarse information, indicating only that Stage 2 has not yet begun or else that Stage 2 has already been completed. Observations made during Stage 2 can provide a more accurate notion of time based on the percentage of decayed bits.

Decay vs. Voltage:

The decay rate of SRAM is expected to depend only on its voltage level (Section 3.7). Temperature, SRAM size, and circuit capacitance all affect the rate of voltage depletion and thus only have secondary effects on memory decay. Our experimental results (Figure 3.8) for five sets of tests (each at least 10 trials) support this hypothesis. The same setup as explained before was used and five different temperatures (one with a 10 mF capacitor and four of them without) were tested.

Impact of Temperature:

The work of Skorobogatov [107] shows that low temperature can increase the remanence time of SRAM, and the work of Halderman et al. [41] similarly shows that low temperature can extend the remanence time of DRAM. For the TARDIS using SRAM

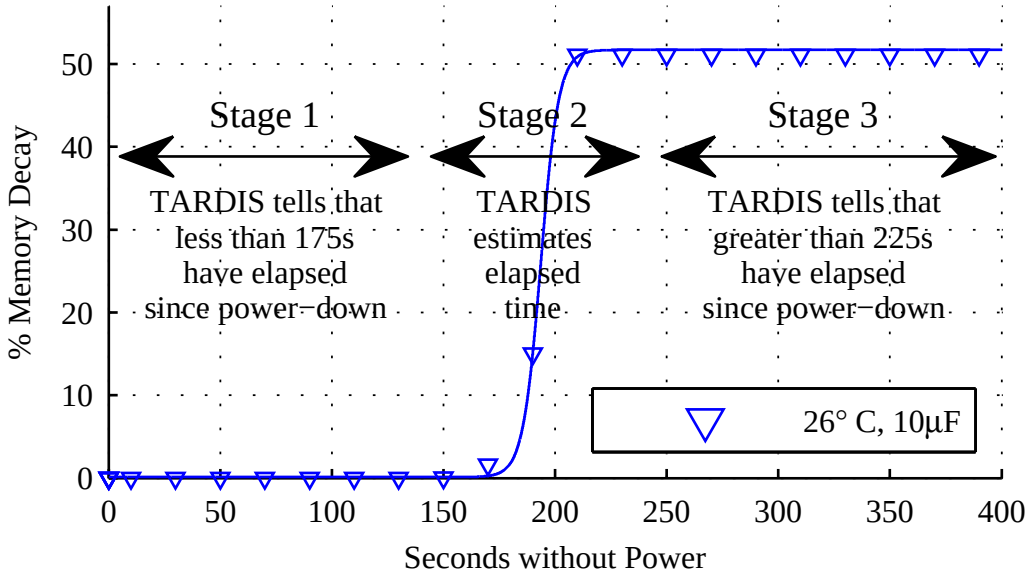


Figure 3.7: The TARDIS presents a three-stage response pattern according to its amount of decay. Before 175 seconds, the percentage of bits that retain their 1-value across a power-off is 100%. For times exceeding 225 seconds, the TARDIS memory has fully decayed. The decay of memory cells between these two thresholds can provide us with a more accurate measurement of time during that period. This graph presents our results measured on a TI MSP430F2131 with 256 B of SRAM and a 10 μF capacitor at 26°C.

decay to provide a notion of time, the interesting question is the opposite case of whether high temperature can decrease remanence. We use the same experimental setup as before (without using capacitors) to investigate how decay time varies across five different elevated temperatures (in the range of $28°C - 50°C$). The off-time of the microcontroller varied from 0 to a maximum of 5 seconds. Figure 3.9 shows that the decay time is non-zero across all temperatures. This indicates that the TARDIS could work at various temperatures as long as changes in the temperature are compensated for. For the TARDIS, this compensation is done by using temperature sensors which are available in many of the today's microcontrollers.[5]

Impact of Additional Capacitance:

[5]According to the TI website, 37% of their microcontrollers are equipped with temperature sensors.

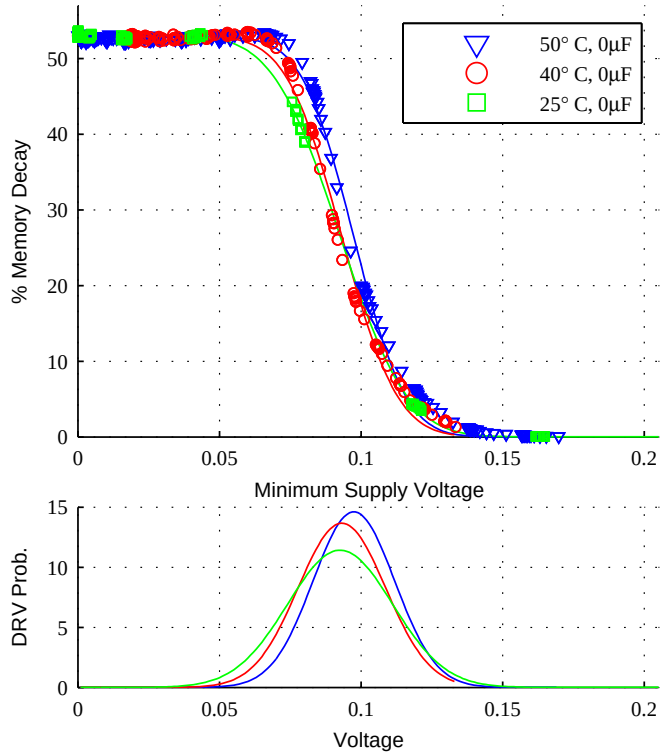


Figure 3.8: Regardless of temperature, the amount of decay depends almost entirely on the minimum supply voltage reached during a power-down. The bottom graph shows the 3-parameter DRV probabilities (Equation 3.4) that best predict the observed relationships between decay and minimum supply voltage for each of the three temperatures. The fit lines in the upper graph show the relationships between decay and minimum supply voltage that are predicted by these DRV models (Section 3.9).

Capacitors can greatly extend the resolution time of the TARDIS. In our experiment, we have tested five different capacitors ranging from 10 μF to 10 mF at 26.5°C. For this experiment, the capacitors were fully charged in the circuit and their voltage decay traces were recorded. These traces were later used in conjunction with our previous remanence-vs.-decay results (Section 3.6) to calculate the time frame achievable with each capacitor. Table 3.5 summarizes the results for the duration of TARDIS Stage 1 and 2 based on capacitor size. The voltage decay traces, our conversion function (DRV Prob.), and the resulting SRAM-decay-over-time graph can be seen in Figure 3.10.

Results ranging from seconds to days open the path for a wide variety of applications

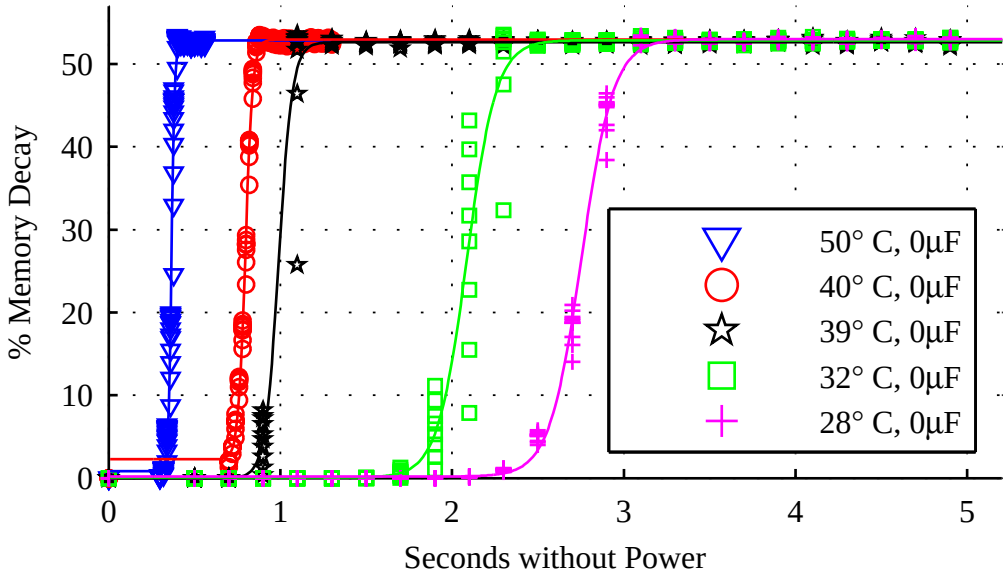


Figure 3.9: The duration of SRAM decay is non-zero across all temperatures even when no capacitor is used. For any given temperature, the duration of SRAM decay is consistent across trials. Increasing the temperature from 28°C to 50°C reduces the duration of both Stage 1 and Stage 2 decay by approximately 80%.

for the TARDIS, as it can now be tweaked to work in a specific time frame. Current RFID-scale devices generally use capacitors ranging from tens of picofarads to tens of microfarads (e.g., [2] [3]). Although a 10 mF capacitor size might be large compared to the size of today's transiently powered devices, the progress in capacitors' size and capacity may very well make their use possible in the near future.

Impact of SRAM Size:

Our hypothesis is that SRAM size has an inverse relation with decay time. This is expected because a larger SRAM will have a larger leakage current and thus will drain the capacitor more quickly. We tested three different models of MSP430 microcontroller with SRAM sizes of 256 B, 2 KB, and 8 KB at 28°C with no capacitor. The DAQ sweeps off-time from 0 to a maximum of 5 seconds. The experiment results are consistent with our hypothesis and are shown in Figure 3.11. It should be noted that SRAM size is not the only difference between these three models, as they also have

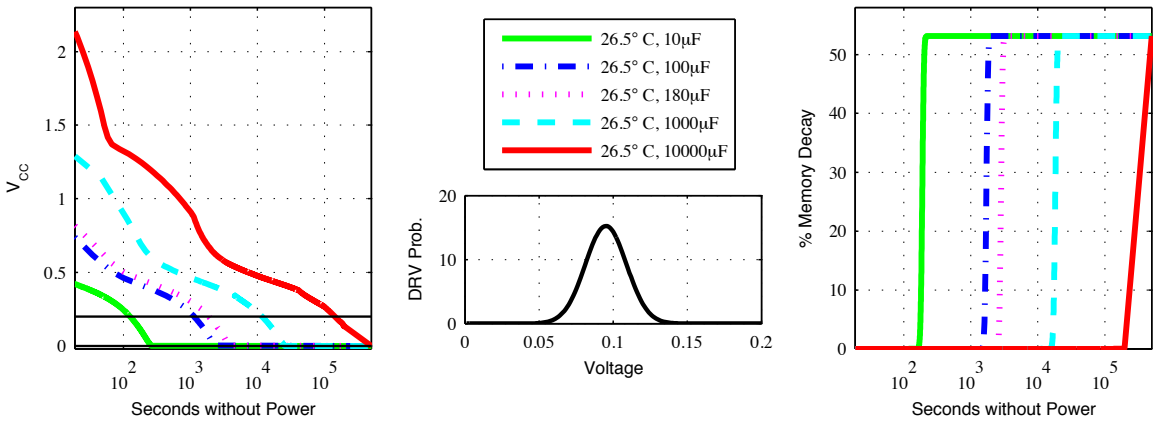


Figure 3.10: For five different capacitor values, measured supply voltage traces are combined with a pre-characterized DRV distribution to predict decay as a function of time. The decaying supply voltages after power is turned off are shown at left. The known DRV probabilities (Equation 3.4) for $26.5°C$ are shown at center. Equation 3.5 maps every supply voltage measurement to a predicted decay, thus creating the memory-decay-vs.-time plots shown at right. The two horizontal lines in the left image at approximately 150 and 50 mV are the voltages where the first and last bits of SRAM will respectively decay.

slightly different power consumptions.

Impact of Chip Variation:

The chip-to-chip variation of the same microcontroller model is not expected to have a major effect on the TARDIS. We tested three instances of the MSP430F2131 with 256 B of memory and no capacitor at $27°C$. The off-time changes from 0 to a maximum of 2.5 seconds with increments of 0.2 seconds. The result shown in Figure 3.12 matches our expectation and shows that changes in decay time due to chip-to-chip variation are insignificant (notice that no capacitor is used and the temperature for one of the chips is one degree higher). This result indicates that TARDIS would work consistently across different chips of the same platform and can be implemented on a system without concern for chip-to-chip variation.

TARDIS Simulation:

We verified the TARDIS mechanism using SPICE simulation of a small SRAM array of 50 cells; the transistor models are 65 nm PTM, the power pin is connected to V_{CC}

Cap. Size	Stage 1 (s)	Stage 2 (s)
0 μF	1.22e0	8.80e-1
10 μF	1.75e2	5.00e1
100 μF	1.13e3	8.47e2
1000 μF	1.17e4	9.50e3
10000 μF	1.43e5	>5.34e4*

* Test was interrupted.

Table 3.5: Estimated time in Stage 1 and Stage 2 of the TARDIS increases as capacitor size increases. The experiments are done on a MSP430F2131 microcontroller at 26.5°C and an SRAM size of 256 B. Stage 1 is the time after the power failure but before the SRAM decay. Stage 2 represents the duration of SRAM decay.

through a D1N4148 diode, and the decoupling capacitor is 70 nF. Each transistor is assigned a random threshold voltage deviation chosen uniformly from range ± 100 mV. Each line in Figure 3.13 plots the voltage difference across the two state nodes A and B for a single SRAM cell. Because all state nodes remain between 0V and V_{CC} during the discharge, the differential voltage is roughly enveloped by $\pm V_{CC}$ as shaded in grey. A positive differential voltage indicates a stored state of 1 (the written state), and a negative differential is a state of 0. Some of the nodes are observed to flip state, starting when V_{CC} reaches 200 mV at 0.55 seconds after power is disconnected. As V_{CC} discharges further, more cells decay by crossing from state 1 to 0. When V_{CC} is powered again at 1.05 seconds, each cell locks into its current state by fully charging either A or B and discharging the other; this is observed in Figure 3.13 as an increase in the magnitude of the differential voltage of each cell.

3.7 Inside an SRAM Cell

Each SRAM cell holds state using two cross-coupled inverters as shown in Figure 3.14; the access transistors that control reading and writing to the cell are omitted from the figure. The cross-coupled inverters are powered via connections to the chip's power supply node. The two states of the SRAM cell, representing a logical 1 and

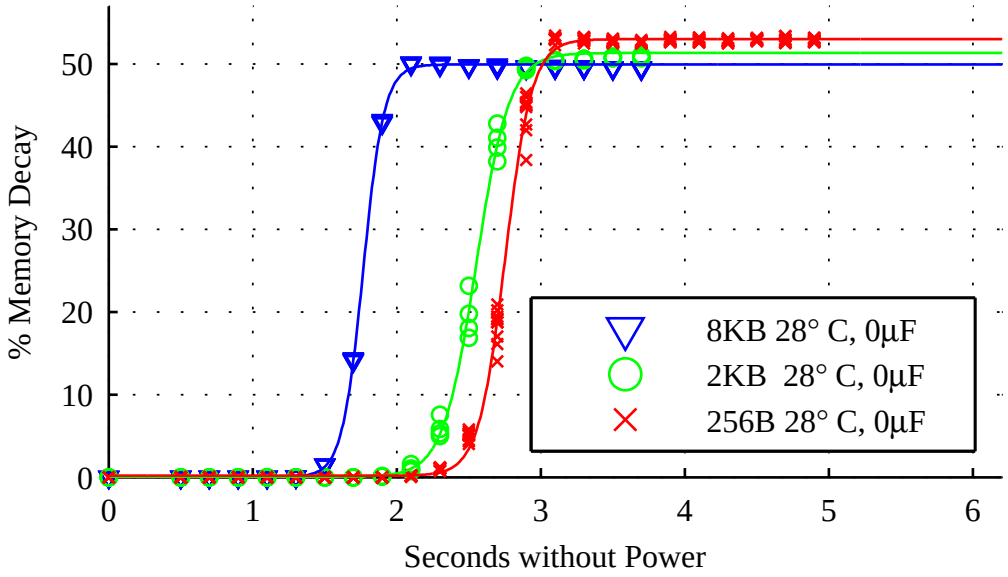


Figure 3.11: Different microcontrollers within the TI MSP430 family with different SRAM sizes exhibit different decay times, but follow the same general trend. The MSP430F2618, MSP430F169, and MSP430F2131 respectively have 8 KB, 2 KB, and 256 B of SRAM.

logical 0, are symmetrical. In each state, under normal conditions, the voltage of either A or B is approximately V_{cc} while the voltage of the other is approximately $0V$.

Data Retention Voltage:

The minimum voltage at which each cell can store either a 0 or 1 is referred to as the cell's data retention voltage (DRV) [89]. Since DRV depends on random process variation, any set of SRAM cells will have a distribution of DRVs. Although the actual DRV distribution depends on process and design parameters, typical values fall within the range of 50 mV to 250 mV; a published design in 0.13 μm has a distribution of DRVs ranging from 80 mV to 250 mV, and our own analysis in this work estimates a majority of DRVs to be in the range of 50 mV to 160 mV (Figure 3.8).

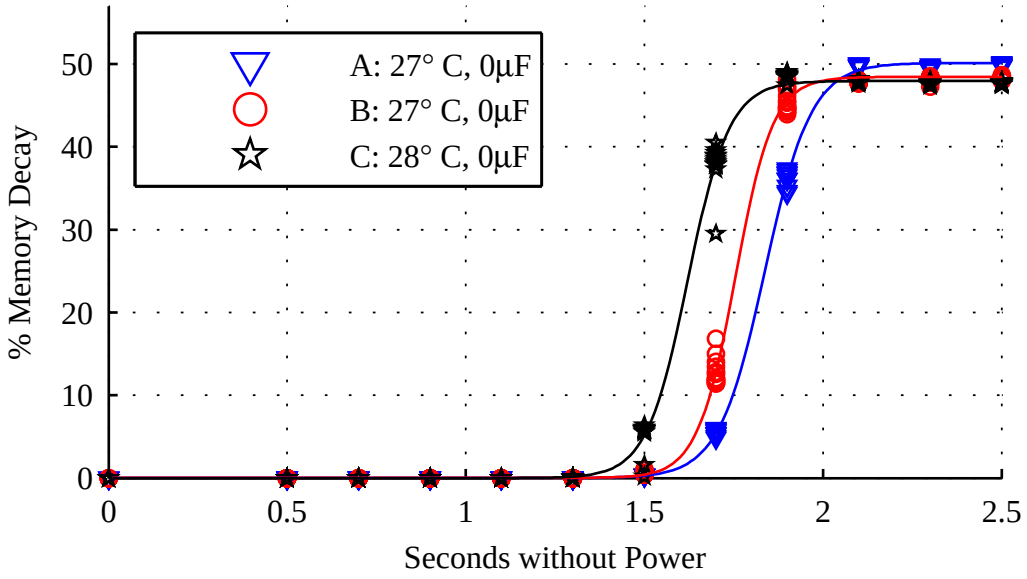


Figure 3.12: Decay versus time in 3 different instances of the MSP430F2131 microcontroller at similar temperatures. The durations of Stage 1 and Stage 2 decay match closely across instances.

3.7.1 Memory Decay Mechanisms

Memory decay occurs in SRAM when a cell loses its state during a power cycle and subsequently initializes to the opposite state upon restoration of power. Given that each cell typically favors one power-up state over the other [49, 39], memory decay can be observed only when the last-written state opposes the favored power-up state. We denote the favored power-up state as the *ground state*, since this is the value an SRAM cell will take at power-up after a very long time without power. We say that a cell written with the value opposite its ground state is *eligible* for memory decay. Each eligible cell will decay once the supply voltage falls below the cell's DRV. Cells that are randomly assigned very low DRVs thus do not decay until the supply voltage is very low. With sufficient capacitance, it can take days for all eligible cells to decay.

Supply voltage decays according to Equation 3.1, where V_{CC}, I_{CC}, and C_{CC} represent the supply voltage, current, and capacitance of the power supply node. The voltage decay is slowed by a large capacitance and low current, and the following

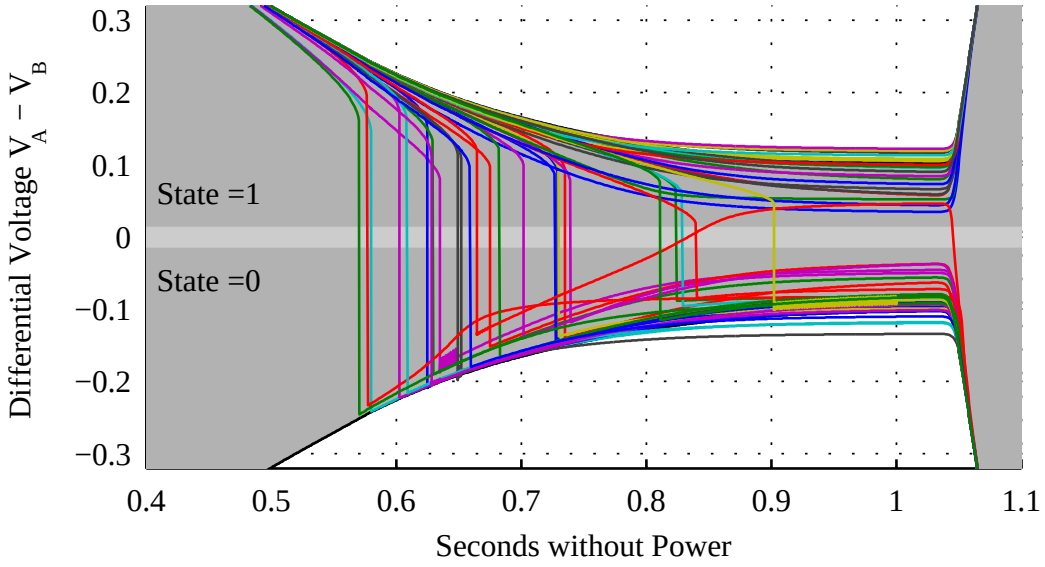


Figure 3.13: The differential voltage of SRAM cells during decay. The envelope of $\pm V_{CC}$ is shaded in grey. All cells are in the 1 state when power is first turned off. As V_{CC} decays, some cells flip from 1 to 0. The cells stabilize when power is restored. The number of zeros after the restoration of power is used to estimate the duration of the power outage.

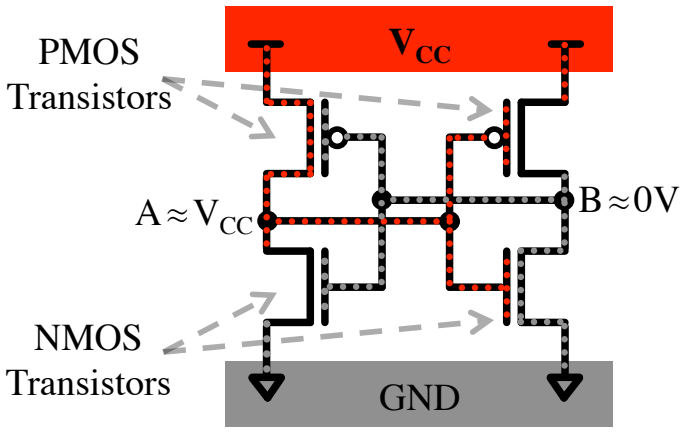


Figure 3.14: The state-holding portion of an SRAM cell consists of two cross-coupled inverters tied to the chip's power and ground nodes.

paragraphs explain why both are present in our TARDIS application.

$$\frac{dv_{CC}}{dt} = \frac{I_{CC}}{C_{CC}} \tag{3.1}$$

Large Capacitance:

The large amount of charge stored on the power supply node is due to the decoupling capacitance that designers add between V_{CC} and *gnd*. During normal operation, this capacitance serves to stabilize the supply voltage to the functional blocks of the chip, including SRAM. In some experiments, the time ranges measurable by the TARDIS are further extended by supplementing the standard decoupling capacitors with additional explicit capacitance.

Low Leakage Current:

The total current I_{CC} comprises the operating current of the microcontroller and the SRAM's data-retention current; both currents are functions of the supply voltage. The current during the voltage decay is shown in Figure 3.15, and explained here:

Immediately after power is disconnected, supply voltages are above 1.4 V and the microcontroller is operational. The observed current is between 250 μA and 350 μA, consistent with the 250 μA current specified for the lowest-power operating point (1.8 V with 1 MHz clock) of the MSP430F2131 [119]. The SRAM current is negligible by comparison. The high current consumption causes the voltage to decay quickly while the microcontroller remains active.

As the voltage drops below 1.4 V, the microcontroller deactivates and kills all clocks to enter an ultra-low power RAM-retention mode in an attempt to avoid losing data. The nominal current consumed in this mode is only the data-retention current, specified to be 0.1 μA for the 256 B of SRAM in the MSP430F2131 [119]. In our observations, I_{CC} is between 0.5 μA and 10 μA during the time that V_{CC} is between 0.5 V and 1.4 V. This current is $1.5 - 3$ orders of magnitude smaller than the current when the microcontroller is active. With so little current being consumed, the supply

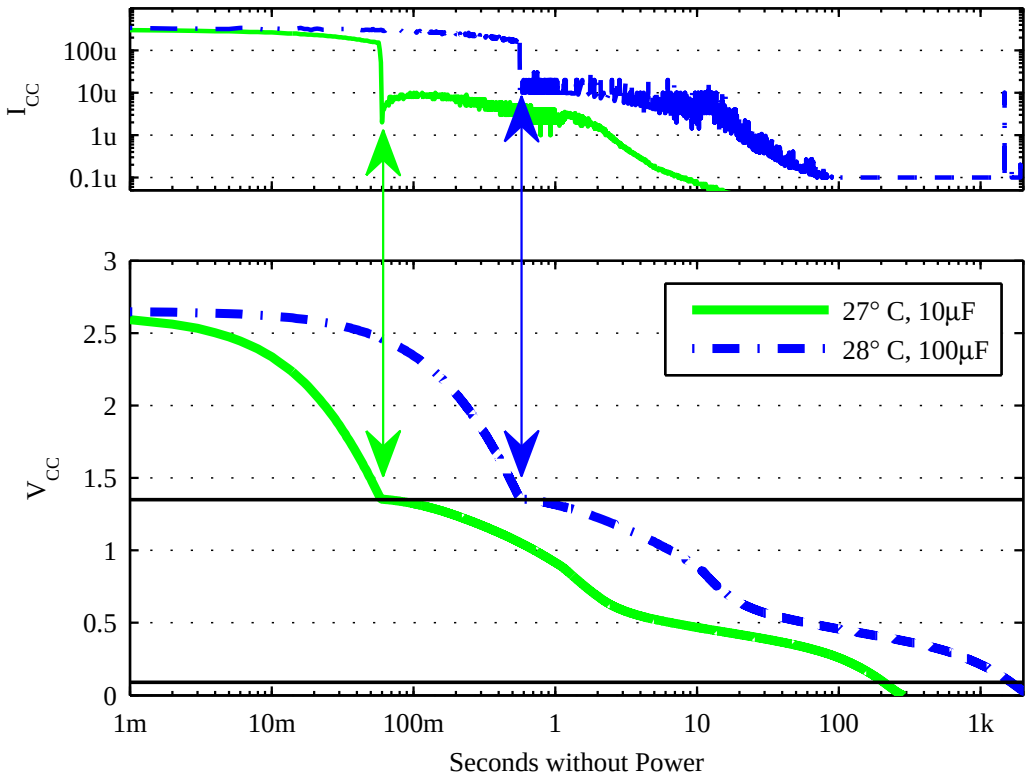


Figure 3.15: Supply voltage and current during two power-down events with different capacitors. The voltage V_{CC} is measured directly, and the current I_{CC} is calculated per Equation 3.1 using the measured $\frac{dV_{CC}}{dt}$ and known capacitor values. The voltage initially decays rapidly due to the high current draw of the microcontroller. When V_{CC} reaches $1.40V$ the microcontroller turns off and I_{CC} drops by several orders of magnitude, leading to a long and slow voltage decay. At the time when V_{CC} crosses the horizontal line at 0.09V, approximately half of all eligible cells will have decayed.

voltage decays very slowly. The current further decreases as the supply voltage drops into subthreshold, and cells begin to experience memory decay.[6]

Impact of Temperature:

Increasing the temperature leads to more rapid memory decay for two reasons. First, increasing the temperature increases the leakage currents that persist through data-retention mode. Increased leakage currents lead to a faster supply voltage decay,

[6]Note that setting V_{CC} to 0 V during the power-down, instead of leaving it floating, reduces voltage and memory decay times by at least an order of magnitude [107] by providing a low impedance leakage path to rapidly drain the capacitance; we have observed this same result in our experiments as well.

causing the supply voltage to drop below DRVs sooner. Second, temperature expedites memory decay by increasing the DRV of SRAM cells [89], causing them to decay at slightly higher supply voltages. Prior work shows a modest 13mV increase in DRV when temperature increases from $27°C$ to $100°C$ [89].

3.7.2 Choosing a State to Write

It is possible to increase the maximum observable memory decay by making every cell eligible for decay. This would be accomplished by characterizing the ground state of each SRAM cell over many remanence-free trials [39, 49], and then writing each cell with its non-ground state in order to make its memory decay observable. In contrast to writing a uniform 1 to all cells, this approach can extract more timing information from the same collection of SRAM cells. However, this alternative requires storing the ground states in non-volatile memory (or equivalently storing written states in non-volatile memory) in order to evaluate whether or not a cell has decayed. Our approach of writing a uniform 1 to all cells makes it possible to evaluate memory decay without this overhead simply by evaluating the Hamming Weight of the SRAM state.

3.8 Alternative Approaches

The more general question of how to keep time without a power source is fundamental and has numerous applications in security and real-time computing. Techniques for keeping time without power or with very reduced power typically rely on physical processes with very long time constants. In CMOS, the most obvious process with a long time constant is the leakage of charge off of a large capacitor through a reverse-biased diode or MOSFET in the cut-off region.

An unexplored alternative to the TARDIS is charging a capacitor whenever the device is active, and checking the capacitor's voltage at a subsequent power-up to determine whether the device has been active recently. The power-up measurement

can be performed using an ADC if available, or else by checking whether or not the remaining voltage is sufficient to register as a logical 1. This approach differs from the TARDIS in incurring monetary and power costs due to the use of a dedicated capacitor and dedicated input-output pins for charging the capacitor and sensing its voltage. Furthermore, the capacitor voltage is still dynamic after power-up, leaving the measurement sensitive to timing variations caused by interrupts. By comparison, the TARDIS uses no dedicated capacitor or input-output pins; its measurement materializes in SRAM at power-up and remains static thereafter until being read and subsequently overwritten.

The EPC Gen2 protocol [27] requires UHF RFID tags to maintain four floating-gate based "inventorial flags" used to support short power gaps without losing the selected/inventoried status. An interesting alternative approach could co-opt these flags to provide a notion of time; however, the flags only persist between 500ms and 5s across power failures. In comparison, the SRAM-based approach in the TARDIS has a resolution time from seconds to hours and has a temperature compensation mechanism. Another advantage of the TARDIS is that it works on any SRAM-based device regardless of the existence of special circuits to support inventorial flags.

3.9 Model of Decay Probabilities

Knowing the DRV distribution of a collection of SRAM cells makes it possible to predict the amount of memory decay that will result from reaching any known minimum supply voltage during a power cycle. We propose a simple and intuitive 3-parameter (α, μ, σ) model to characterize the DRV distribution. We chose the parameters such that the model predictions agree with empirical data relating memory decay to minimum supply voltage.

Cells eligible for memory decay after being written with a value of 1 are those with a ground state of 0. We use $g = 0$ to denote cells with a 0 ground state, and use α to

denote the fraction of cells with this ground state; α is therefore the largest fraction of cells that can decay after writing a 1 to all cells.

$$\Pr(g = 0) = \alpha \tag{3.2}$$

Among cells that are eligible for memory decay, we assume that DRVs are normally distributed with mean μ and standard deviation σ (Equation 3.3).

$$DRV \mid (g = 0) \ \sim \ \mathcal{N}\left(\mu, \sigma^2\right) \tag{3.3}$$

The probability of a randomly selected cell being eligible for memory decay and having $DRV = v$ is given by Equation 3.4. This is an α-scaled instance of the PDF of a normally distributed random variable, and we refer to this as the "DRV probability" of voltage v.

$$\Pr\left((g = 0) \wedge (DRV = v)\right) = \frac{\alpha}{\sigma\sqrt{2\pi}} e^{-(v-\mu)^2/\left(2\sigma^2\right)} \tag{3.4}$$

If the minimum voltage of a power cycle is known, then the 3-parameter model can predict the memory decay. The cells that will decay are eligible cells with a DRV that is above the minimum supply voltage reached during the power cycle. A closed-form equation for predicting the memory decay from the minimum voltage and model parameters is then given by Equation 3.5; this equation is 1 minus the CDF of a normally distributed random variable, scaled by α.

$$D_{PRED}(v_{min}, \alpha, \mu, \sigma) = \alpha \left(1 - \frac{1 + erf\left(\frac{v_{min}-\mu}{\sigma\sqrt{2}}\right)}{2}\right) \tag{3.5}$$

A 3-parameter model is evaluated according to how well its predicted memory decay matches empirical data. The evaluation is performed using a set of n observations $\langle v_0, D(v_0)\rangle, \langle v_1, D(v_1)\rangle, \ldots, \langle v_{n-1}, D(v_{n-1})\rangle$; each observation is a measurement of the

minimum supply voltage reached during a power cycle, and the memory decay observed across that power cycle. The prediction error of any model is defined according to Equation 3.6. We initially use the set of measurements to find the model parameters that minimize the prediction error (see Figure 3.8).

$$ERR(\alpha, \mu, \sigma) = \sum_{i=0}^{n-1} \left(D_{PRED}\left(v_i, \alpha, \mu, \sigma\right) - D\left(v_i\right)\right)^2 \tag{3.6}$$

After measurements are used to fit the model parameters to empirical data, the model is subsequently used to predict memory-decay-vs.-time curves from voltage-vs.-time measurements (see Figure 3.10).

3.10 Related Work

RFID Security and Privacy:

The inability of intermittently powered devices to control their response rates has made them susceptible to various attacks. An RFID tag could be easily "killed" by exhausting all possible 32-bit "kill" keys. Such unsafe "kill" commands could be replaced with a "sleep" command [55]; however, lack of a timer to wake up the tag in time has made the use of the "sleep" command inconvenient. The key to e-passports can be discovered in real time by brute-force attacks [8]. The attack could be slowed down if the e-passport had a trustworthy notion of time. The minimalist model [54] offered for RFID tags assumes a scheme that enforces a low query-response rate. This model could be implemented using the TARDIS.

Secure Timers:

To acquire a trustworthy notion of time, multiple sources of time can be used to increase the security level of a timer [97]; but this requires the device to interact actively with more than one source of time, which is not practical for RFID tags that use passive radio communication. The same issues prevent us from using the

Lamport clock and other similar mechanisms that provide order in distributed systems [59]. This inability to acquire secure time precludes the use of many cryptographic protocols, including timed-release cryptography [74] [95]

Ultra-low Power Clocks:

With the rise of pervasive computing come a need for low-power clocks and counters. Two example applications for low-power clocks are timestamping secure transactions and controlling when a device should wake from a sleep state. The lack of a rechargeable power source in some pervasive platforms requires ultra-low power consumption. Low voltage and subthreshold designs have been used to minimize power consumption of digital circuits since the 1970s [114]. Circuits in wristwatches combine analog components and small digital designs to operate at hundreds of nW [126]. A counter designed for smart cards uses adiabatic logic to operate at 14KHz while consuming 11nW of power [116]. A gate-leakage-based oscillator implements a temperature-invariant clock that operates at sub-Hz frequencies while consuming 1pW at 300mV [66]. A TI-recommended technique [92] for the MSP430 is to charge a dedicated external capacitor from the microcontroller while in a low-power sleep mode with clocks deactivated; the microcontroller is triggered to wake up when the capacitor voltage surpasses a threshold. But all of these solutions, while very low-power, still require a constant supply voltage and hence a power source in the form of a battery or a persistently charged storage capacitor. However, embedded systems without reliable power and exotic low-power timers may still benefit from the ability to estimate time elapsed since power-down.

Attacks Based on Memory Remanence:

Processes with long time constants can also raise security concerns by allowing data to be read from supposedly erased memory cells. Drowsy caches [29] provide a good background on the electrical aspects of data retention. Gutmann stated that older SRAM cells can retain stored state for days without power [40]. Gutmann also suggest

exposing the device to higher temperatures to decrease the retention time. Anderson and Kuhn first proposed attacks based on low-temperature SRAM data remanence [5]. Experimental data demonstrating low-temperature data remanence on a variety of SRAMs is provided by Skorobogatov [107], who also shows that remanence is increased when the supply during power-down is left floating instead of grounded. More recent freezing attacks have been demonstrated on a 90nm technology SRAM [122], as well as on DRAM [41]. Data remanence also imposes a fundamental limit on the throughput of true random numbers that can be generated using power-up SRAM state as an entropy source [104]. The TARDIS, in finding a constructive use for remanence and decay, can thus be seen as a counterpoint to the attacks discussed in this section. The TARDIS is the first *constructive* method that takes advantage of SRAM remanence to increase the security and privacy of intermittently powered devices.

3.11 Conclusions

A trustworthy source of time on batteryless devices could equip cryptographic protocols for more deliberate defense against semi-invasive attacks such as differential power analysis and brute-force attacks. The TARDIS uses remanence decay in SRAM to compute the time elapsed during a power outage—ranging from seconds to hours depending on hardware parameters. The mechanism provides a coarse-grained notion of time for intermittently powered computers that otherwise have no effective way of measuring time. Applications using the TARDIS primarily rely on timers with hourglass-like precision to throttle queries. The TARDIS consists purely of software, making the mechanism easy to deploy on devices with SRAM. A novel aspect of the TARDIS is its use of memory decay or data remanence for improved security rather than attacking security. Without the TARDIS, batteryless devices are unlikely to give you the time of day.

CHAPTER IV

DRV-Fingerprinting: Using Data Retention Voltage of SRAM Cells for Chip Identification

4.1 Introduction

RFID circuits can be identified or authenticated using static identifiers stored in non-volatile memory or through the use of identifying physical characteristics. Physical characteristics have several security advantages over static identifiers, including immutability and resistance to cloning and tampering. The physical characteristics can be viewed as an identifying fingerprint of a given device. More formally, physical fingerprints are a component of a particular type of physical unclonable function (PUF) that is originally described as a physically obfuscated key [35], and more recently as a weak PUF [39].

If used for identification or constructing secret keys, fingerprint observations must be consistent over time. Sensing the microscopic variations that make each device unique while also minimizing the impact of noise is a fundamental concern in PUFs. Much effort is spent on error correction of somewhat-unreliable fingerprints or PUF outputs. Error correcting codes are expensive in terms of the number of raw bits required to create a reliable key, and more so if the number of correctable errors must be large. Toward this goal, we present a new fingerprinting method that is more

reliable across trials than comparable previous approaches.

In this work we propose a new method for chip fingerprinting that uses Data Retention Voltage (DRV) in SRAM as the identifier. The DRV of an SRAM is the minimum voltage at which its cells can retain state. DRV fingerprints are found to be more informative than other approaches for fingerprinting SRAM that have been proposed in research [39, 49] and commercially.[1] The physical characteristics responsible for DRV are imparted randomly during manufacturing and therefore serve as a natural barrier against counterfeiting. The proposed technique has the potential for wide application, as SRAM cells are among the most common building blocks of nearly all digital systems including smart cards and programmable RFID tags.

The contributions of this work are as follows:

- Demonstrating that the DRVs of SRAM cells are consistent fingerprints capable of identifying devices among a population.
- Demonstrating that DRV fingerprints make use of physical variations in a way that is similar to SRAM power-up fingerprints, but that DRV fingerprints have the potential for more accurate identification.

The remainder of this chapter is structured as follows: Section 4.2 introduces data retention voltage. Section 4.3 explains how the DRVs of SRAM cells are characterized. Section 4.4 evaluates DRV fingerprinting using experimental data. Sections 4.5 and 4.6 review related work and present directions for future work.

4.2 Data Retention Voltage

A data retention failure is said to occur when an SRAM cell spuriously flips state due to insufficient supply voltage. The data retention voltage (DRV) of an SRAM array signifies the minimum supply voltage at which all SRAM cells can store

[1] http://www.intrinsic-id.com/

arbitrary state. DRV is studied in the literature as a limit to supply voltage scaling. Various simulation models [127, 19, 79] and silicon measurements [89] show modern SRAM DRVs to be under 300mV. Most previous literature focuses on cases where the supply voltage of the circuit remains safely above DRV. While remaining above DRV, the supply voltage can be adjusted to reduce leakage power [30], compensate for manufacturing variability [79], or compensate for environmental variations [127].

Each SRAM cell uses the positive feedback of cross-coupled inverters to hold state on two complementary storage nodes. Retention failures occur at low supply voltages because the low voltage weakens the positive feedback of the cross-coupled inverters. Due to asymmetric process variation, at some low supply voltages a transition from a written state to the opposite state becomes inevitable; observations about the direction of such transitions and the voltages at which they occur are the basis for DRV fingerprints. Any collection of SRAM cells has a distinctive DRV fingerprint because of its unique random process variation.

4.3 Characterizing the DRV of an SRAM Cell

The DRVs of SRAM cells are characterized by repeatedly lowering the SRAM supply voltage and observing the highest voltage at which each cell fails. If the SRAM supply node also supplies the processing core, then the low voltages used for the characterization will cause the core to reset and lose its state. Our experiments avoid this difficulty by using non-volatile memory to maintain persistency across the low voltages. However, a custom integrated circuit designed for DRV fingerprinting can also avoid this difficulty by using an SRAM supply node that is decoupled from the nominal supply node of the processor. This is often done, for example, in power-gated circuits where unused on-chip functional blocks are turned off entirely while the chip as a whole remains powered.

We characterize the DRV of an SRAM cell c with a pair $\langle v_c^0, v_c^1 \rangle$. Each v_c^w in the

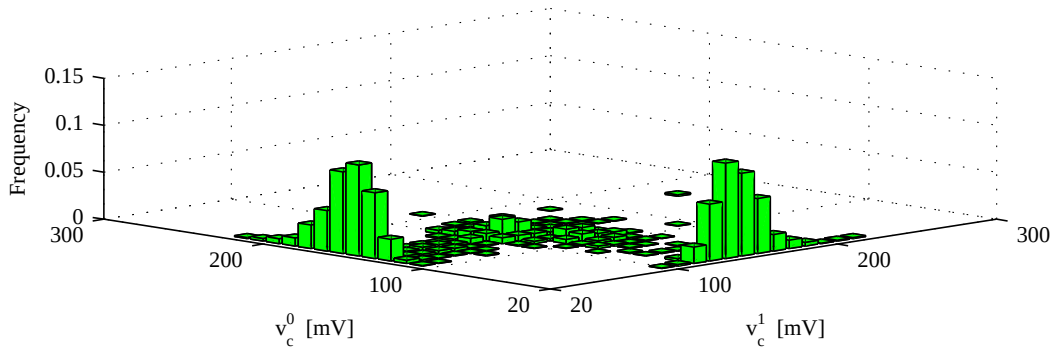


Figure 4.1: The joint probability distribution function over all cells of the two variables (v_c^0 and v_c^1) comprising a DRV characterization. The distribution is determined experimentally using Algorithm 9, and shows that a large fraction of cells have the minimum possible value of 20mV for either v^0 or v^1, but none have the minimum value (or near-minimum values) for both. A cell with a minimum value for v^0 or v^1 is a cell that retains one written state across all test voltages.

pair represents the highest voltage at which cell c will have a retention failure after state w is written to it. In principle, v_c^0 and v_c^1 are real-valued; but in practice, we approximate each using one of $N = (300mV - 20mV)/\Delta$ discrete values as shown in Algorithm 9. With Δ set at 10mV, the $N = 28$ possible values for v_c^0 and v_c^1 are $\{20mV, 30mV, \ldots, 290mV\}$. The frequency of observing different DRV pairs is shown in the joint probability distribution function of variables v_c^0 and v_c^1 in Fig. 4.1.

4.3.1 Experimental Setup

We examine the DRV of SRAM cells using Algorithm 9 implemented as follows: A microcontroller runs a program that sets all available memory bits to either 1 or 0. The supply voltage is then decreased to a value between 300mV and 20mV ($\Delta = 10mV$) for 5 seconds. When supply voltage is restored to 3V, the program stores the content of SRAM to the flash memory. Note that we conservatively use $t_{wait} = 5s$ to avoid missing marginal failures. Simulations by Nourivand et al. [79] using a procedure similar to Algorithm 9 show that waiting for $t_{wait} = 2ms$ at a reduced supply voltage is sufficient to observe retention failures. An Agilent U2541A-series data acquisition

Algorithm 9 Characterize the DRV fingerprint of a set of SRAM cells.

Prerequisite: C – a set of SRAM cells

Ensure: v_c^0, v_c^1 – the DRV characterizations of each SRAM cell $c \in C$.

Let V_{nom} be the nominal supply voltage (V_{dd}) for the chip
Let s_c refer to the logical state of SRAM cell $c \in C$.
Let s'_c refer to the logical state of NVM cell that corresponds to SRAM cell c.
for $w = 0, 1$ **do**
  for $c \in C$ **do**
    $s_c \leftarrow w$ {write w into SRAM cell}
    $s'_c \leftarrow w$ {write w into NVM cell}
    $v_c^w \leftarrow 0$ {value used if no retention failure observed}
  end for
  $v_{test} \leftarrow 300mV$ {initialize test voltage}
  while $v_{test} > 20mV$ **do**
    lower chip voltage from V_{nom} to v_{test}
    wait for t_{wait} seconds
    raise chip voltage from v_{test} to V_{nom}
    for $c \in C$ **do**
      if $(s_c = \neg w) \wedge (s'_c = w)$ **then**
        SRAM cell c had a retention failure from state w at voltage v_{test}, but previously had no failure at voltage $v_{test} + \Delta$. Therefore v_{test} approximates the largest voltage that induces a retention failure after writing w.
        $v_c^w \leftarrow v_{test}$
      end if
      $s'_c \leftarrow s_c$ {write SRAM to NVM}
    end for
    $v_{test} \leftarrow v_{test} - \Delta$ {try a lower voltage next}
  end while
end for

(DAQ) unit controls the supply voltage and the timing of when voltage is raised and lowered. Thermal tests are conducted inside of a Sun Electronics EC12 Environmental Chamber [112], and an OSXL450 infrared non-contact thermometer [82] with $\pm 2°C$ accuracy is used to verify the temperature. All experiments use instances of Texas Instruments MSP430 F2131 microcontrollers with 256 bytes of SRAM, of which 240 bytes are available for DRV fingerprinting. The DRV of each cell is characterized 20 times. The total runtime to characterize all 240 bytes of SRAM on a chip once using Algorithm 9 is given by t_{proc} in Eq. 4.1, and is 140 seconds for the conservative case of $\Delta = 10mV$ and $t_{wait} = 5s$.

$$t_{proc} = t_{wait} \times \frac{300mV - 20mV}{\Delta} \tag{4.1}$$

4.3.2 Information Content of SRAM Cell DRV

The DRV of each cell has N^2 possible outcomes representing all combinations of N outcomes for v_c^0 and the N outcomes for v_c^1 (in our case $N = 28$). The DRV of each cell is then a random variable X with N^2 outcomes denoted x_0 through x_{N^2-1}. The total entropy $H(X)$ is the expected information value of the DRV of an unknown cell. Entropy depends (per Eq. 4.2) on the probabilities of each DRV outcome, denoted $p(x_i)$. In the ideal case where all N^2 outcomes are equally likely (e.g. $p(x_i) = 1/N^2$ for all x_i), each DRV would have almost 10 bits of entropy. Applying Eq. 4.2 to the decidedly non-uniform outcome probabilities of Fig. 4.1 shows the actual entropy of a DRV to be 5.12 bits. The most frequently observed DRV outcomes are given in Table. 4.1.

Eq. 4.1 shows that runtime is inversely proportional to Δ, so we consider the information loss from making Δ larger than 10mV. Fig. 4.2 shows the ideal and actual entropy of DRV characterizations when different values of Δ are used. In the extreme case where $\Delta = 140mV$, variables v_c^0 and v_c^1 are each restricted to the values $\{20mV, 160mV\}$, so the ideal entropy of the DRV is equivalent to 2 flips of a fair coin. The values of Δ used in Fig. 4.2 are chosen on account of being unambiguously recreatable from the $\Delta = 10mV$ data.

$$H(X) = -\sum_{i=1} p(x_i) \log p(x_i) \tag{4.2}$$

Table 4.1: The 4 most commonly observed weak and strong DRV characterizations, and the probability of observing each in a randomly selected trial.

(a) Most common weak DRVs

Outcome $\langle v_c^0 , v_c^1 \rangle$	Freq.
$\langle 130mV , 100mV \rangle$	0.0096
$\langle 120mV , 100mV \rangle$	0.0076
$\langle 130mV , 110mV \rangle$	0.0070
$\langle 120mV , 110mV \rangle$	0.0070

(b) Most common strong DRVs

Outcome $\langle v_c^0 , v_c^1 \rangle$	Freq.
$\langle 20mV , 130mV \rangle$	0.0893
$\langle 20mV , 120mV \rangle$	0.0719
$\langle 130mV , 20mV \rangle$	0.0685
$\langle 20mV , 140mV \rangle$	0.0651

4.3.3 Observations about Strong and Weak Cells

We abstract the N^2 possible DRV characterizations (Fig. 4.1) into three classes[2] that are sufficient to demonstrate general observations about all DRVs:

- A *strongly 0* DRV characterization is a pair $\langle v_c^0, v_c^1 \rangle$ such that $v_c^0 = 20mV$ and $v_c^1 > 20mV$. A strongly 0 DRV indicates that no retention failure occurs at any voltage v_{test} after state 0 is written.
- A *strongly 1* DRV characterization is a pair $\langle v_c^0, v_c^1 \rangle$ such that $v_c^0 > 20mV$ and $v_c^1 = 20mV$. A strongly 1 DRV indicates that no retention failure occurs at any voltage v_{test} after state 1 is written.
- A *weak* DRV characterization is a pair $\langle v_c^0, v_c^1 \rangle$ such that $v_c^0 > 20mV$ and $v_c^1 > 20mV$. A weak DRV indicates that a failure is observed at some voltage v_{test} after each state is written.

The variation-dependent behavior of an SRAM cell occurs somewhere between 20mV and 300mV for each cell; above 300mV all cells can reliably hold either the 0 or the 1 state, and below 20mV no cells can do so. When a cell produces a strongly 0 or strongly 1 characterization, it means (per Algorithm 9) that for one written state the supply voltage is lowered all the way through the sensitive region down to 20mV and then raised back up without causing a failure. A strongly 0 or strongly 1

[2]Note that no observation of $\langle v_c^0, v_v^1 \rangle = \langle 20mV, 20mV \rangle$ is ever made, so we do not include this outcome in any of the three cases.

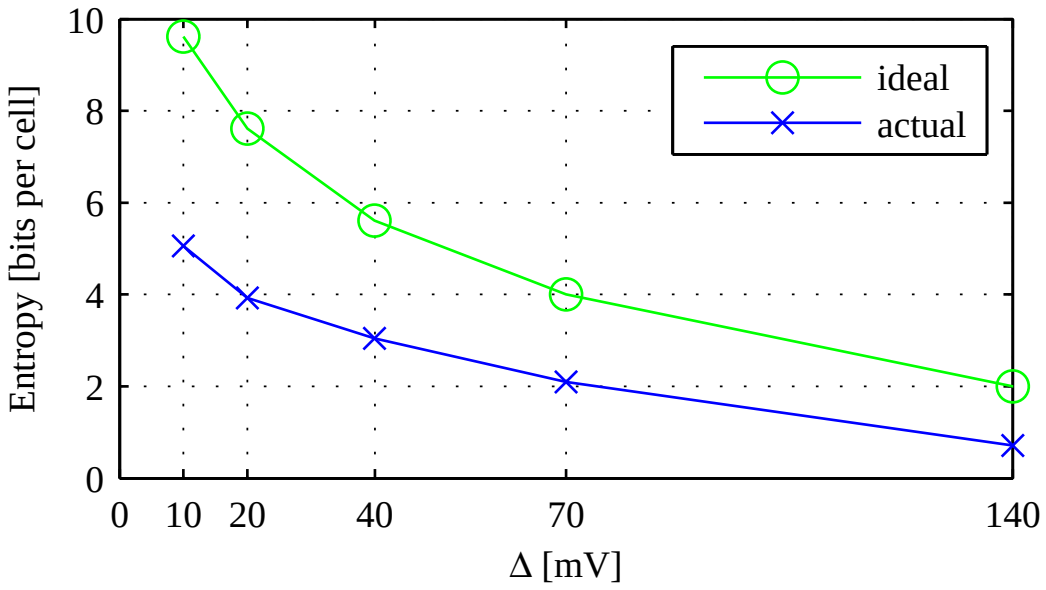


Figure 4.2: Sweeping Δ from 10mV to 140mV shows that a loss of measurement precision reduces entropy of each cell's DRV characterization.

characterization therefore indicates a strong preference for one state over the other at all supply voltages. A weak characterization is when each written state flips at some voltage within the sensitive region, and neither state can be retained down to 20mV.

Both strong and weak DRV characterizations are largely repeatable across trials. Fig. 4.3 shows the distribution of DRVs produced by randomly selected cells for which the first DRV produced is one of the 4 most commonly observed weak DRVs from Table 4.1(a); each plot shows the conditional probability distribution of a subsequent DRV characterization. Occasionally the same cells that produce a weak DRV produce a strong DRV in subsequent trials. Fig. 4.4 shows the same analysis for the 4 most commonly observed strong DRVs; none of the cells subsequently produces the opposite strong characterization.

4.3.4 Relation to Power-up State

It is known that SRAM cells consistently power-up to the same state [39, 49] in a majority of trials. Cells with highly reliable power-up states tend to be the same cells

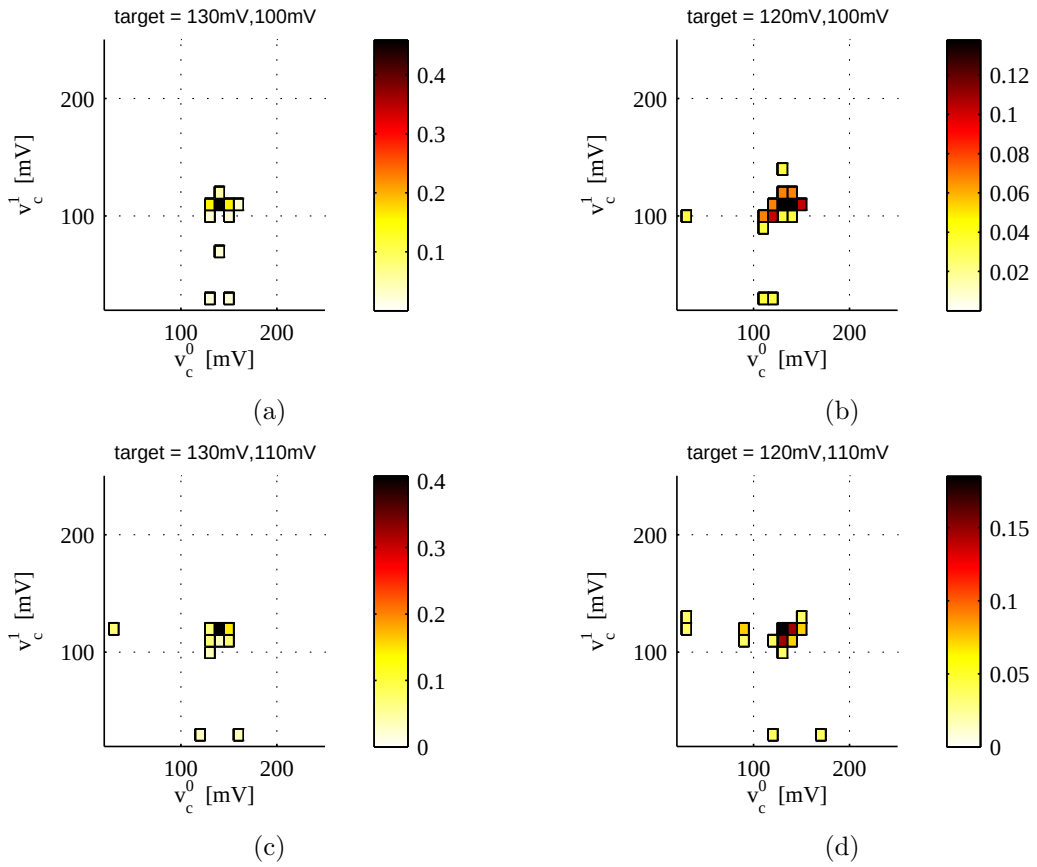


Figure 4.3: For each of the 4 most frequently observed weak DRVs (see Table 4.1(a)), the DRV in a second trial from a cell that produced the frequently observed DRV in a first trial.

with strong DRV characterizations. Fig. 4.5 shows the mean power-up state over 28 trials for cells that produced a strongly 0 or strongly 1 DRV characterization. Among cells with strongly 0 DRV, 98.6% power-up to the 0 state in all 28 power-up trials (Fig. 4.5(a)). Similarly, 95.1% of cells characterized as strongly 1 consistently power-up to the 1 state (Fig. 4.5(a)). Although a strong DRV fingerprint is correlated to power-up tendency, the DRV provides a more informative identifier than does power-up by providing information about the maximum voltage at which the unfavored state cannot be reliably stored.

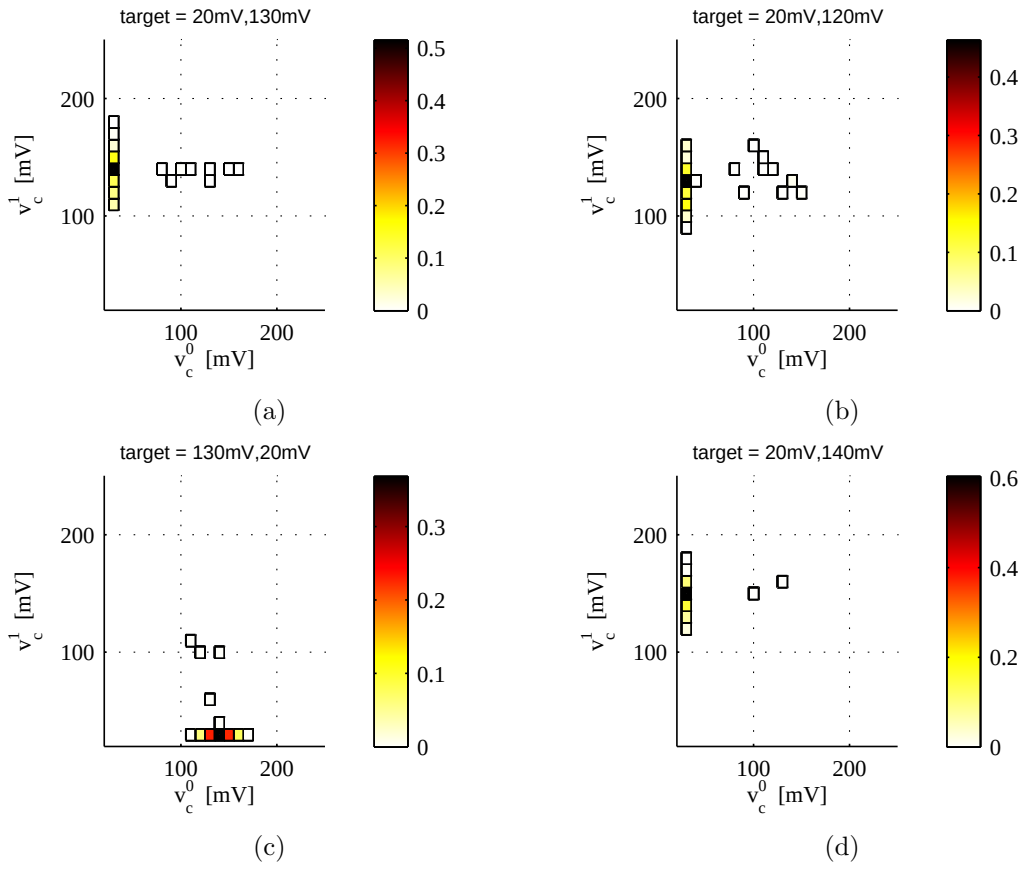


Figure 4.4: For each of the 4 most frequently observed strong DRVs (see Table. 4.1(b)), the DRV in a second trial from a cell that produced the frequently observed DRV in a first trial.

4.4 Fingerprint Matching

A DRV fingerprint is obtained from a single characterization of a set of adjacent cells within an SRAM. A k-bit fingerprint F_i comprises cell characterizations $\langle v_i^0, v_i^1\rangle, \langle v_{i+1}^0, v_{i+1}^1\rangle, \ldots, \langle v_{i+k-1}^0, v_{i+k-1}^1\rangle$. The difference between fingerprints is the sum of the differences between their corresponding single-cell characterizations. Recalling that each DRV is a point $\langle v_c^0, v_c^1\rangle$ in 2-dimensional space, we define the distance between two DRVs according to the square of their distance along each dimension (Eq. 4.3). For comparison, a second metric used is the Hamming distance between power-up trials; this is shown by Eq. 4.4, where p_i is the state of the i^{th} bit of SRAM after a power-up.

Table 4.2: Probability of different pairwise outcomes when 2 DRV fingerprints are taken from a randomly chosen cell. Over the 5000 samples collected, no cell ever has a DRV that is strongly 1 in one trial and strongly 0 in another, but 5.6% of outcomes have one strong and one weak DRV.

	Strongly 0	**Weak**	**Strongly 1**
Strongly 0	35.80%	3.10%	0.00%
Weak	-	24.98%	2.48%
Strongly 1	-	-	33.64%

$$d1(F_i, F_j) = \sum_{n=0}^{k-1} \left(v_{i+n}^0 - v_{j+n}^0\right)^2 + \left(v_{i+n}^1 - v_{j+n}^1\right)^2 \tag{4.3}$$

$$hd(F_i, F_j) = \sum_{n=0}^{k-1} p_{i+n} \oplus p_{j+n} \tag{4.4}$$

4.4.1 Identification at Nominal Temperature

At the nominal operating temperature of $29°C$, three experiments compare DRV fingerprints with power-up fingerprints. These experiments are explained in the following subsections; the first shows the histograms of distances between fingerprints, and the second and third evaluate the accuracy of distance-based matching.

Histogram of Distances Between Fingerprints

A first experiment shows that DRV fingerprints are repeatable and unique, as is necessary for successfully identifying chips within a population. Within-class pairings are of multiple fingerprints generated by the same set of cells on the same device. Between-class pairings are from different sets of cells on the same device, or from any sets of cells on different devices. The similarity of any two fingerprints is quantified by a distance, and this distance is the basis for determining the correct identity of a fingerprint. If within-class fingerprint pairings consistently have smaller distances than between-class pairings, then it is possible to determine identity by choosing an appropriate threshold that separates the two classes. The histograms of within-class

and between-class distances for DRV and power-up fingerprints are shown in Fig. 4.6. These histograms represent all data collected from the MSP430F2131 microcontrollers at room temperature. The distances on the x-axes are not directly comparable across metrics; of importance is only whether the two classes are clearly separable within each plot.

Accuracy of Top Match

The next experiment performed at nominal temperature evaluates how reliably a single within-class DRV fingerprint can be identified among a population. This experiment matches a single 16-bit target fingerprint against a population containing another fingerprint from the same cells and one fingerprint from each of the 239 remaining locations across 2 chips. A positive result occurs if the closest match among the 240 possibilities is from the same SRAM cells as the target. The results of the top match experiment are shown in Table 4.3; the column labelled "co-top" shows the percentage of trials where there are multiple top matches and one of them correctly matches the target. Multiple top matches are relatively common in Hamming distance matching due to the small number of possible distances between fingerprints. Compared to power-up fingerprints, matching based on DRV fingerprints is 28% more likely to have the correct match be closer to the target (i.e. separated by a smaller distance) than all incorrect matches.

Table 4.3: Over 300 trials with a population of 240 16-bit fingerprints, DRV identification returns the fingerprint that correctly matches the target more reliably than power-up state identification. Matching based on power-up state more frequently returns a misidentified fingerprint, or returns multiple fingerprints among which one is the correct match (denoted "co-top").

	Top	**Co-top**	**Misidentified**
DRV (d1)	99.7%	-	0.3%
Power-up	71.7%	24.7%	3.6%

Precision and Recall

The top match experiment is generalized to the case of identifying multiple correct matches among a larger population, and again shows DRV fingerprints to outperform power-up fingerprints. In this experiment, our goal is to find all correct matches in the population, without also finding too many incorrect matches. In doing so, the distance that is considered to be the threshold between a correct and incorrect match can be adjusted. If the threshold is too low then correct matches may not be identified, but if the threshold is too high then false positives will occur. *Recall* refers to the fraction of within-class pairings under the threshold, and *precision* refers to the fraction of pairings under the threshold that are within-class. Increasing the threshold will sacrifice precision for recall, and decreasing the threshold will sacrifice recall for precision. An ideal result is for both precision and recall to be 1; this result occurs if all correct matches are identified as within-class (perfect recall) with no incorrect ones identified as within-class (perfect precision).

The precision and recall plots of Fig. 4.7 are obtained by iterating the following procedure. One 16-bit segment of SRAM is chosen for identification. One fingerprint trial from this segment is chosen at random as the target, and it is matched against a population of 1019 fingerprints comprising 19 from the same SRAM segment (within-class pairings) and 1000 non-matching fingerprints (between-class pairings). The non-matching fingerprints are randomly selected among 20 trials from 239 other segments of SRAM[3]. The matching threshold is swept to find achievable precision-versus-recall tradeoffs, and each achievable tradeoff is a point in Fig. 4.7. The large number of tradeoff points in the plot is collected from multiple iterations of this procedure. The general trend is that DRV fingerprints produce better recall for a given precision, or better precision for a given recall compared to power-up fingerprints.

[3]The 239 eligible 16-bit segments are the 119 remaining on the target's own chip, and all 120 such locations on the other device.

4.4.2 Impact of Temperature Variations

Given that DRV fingerprints would likely be used in real-world scenarios without precisely-controlled temperatures, a final experiment explores the impact of temperature on DRV fingerprints. This experiment is similar to the experiment of subsection 4.4.1, but the pairs of fingerprint observations used to generate the within-class distances are now made at different temperatures. The results are shown in Fig. 4.8. The increase of within-class distances across temperature implies a diminished reliability. To compensate for this, larger fingerprints (comprising more bits) may be needed for identification, and more robust error correcting codes may be needed in key-generation applications. If the increased within-class distances are due to a uniform shift in the DRVs of all cells, then a promising direction for future work would be to design a matching scheme that is insensitive to this type of uniform shift.

4.5 Related Works

A wide variety of PUFs and fingerprints based on custom or pre-existing integrated circuit components have been developed. The identifying features used by custom designs include MOSFET drain-current [70], timing race conditions [34], and the digital state taken by cross-coupled logic after a reset [109]. IC identification based on pre-existing circuitry is demonstrated using SRAM power-up state [49, 39], and physical variations of flash memory [88]. Lee et al. [61] derive a secret key unique to each IC using the statistical delay variations of wires and transistors across ICs. Bhargava et al. explore circuit-level techniques for increasing the reliability of SRAM PUFs [13]. An experimental evaluation of low-temperature data remanence on a variety of SRAMs is provided by Skorobogatov [107], and SRAM remanence in RFID has been studied by Saxena and Voris as a limitation to SRAM-based true random number generation [104].

Previous works [123, 98] have used error correction to construct secret keys from noisy PUF sources; however, this is expensive in terms of gates and other resources. To give an idea of the cost of error correction, BCH codes previously used with PUFs include one to correct 21 errors among 127 raw bits in creating a 64-bit key [110], and to correct 102 errors among 1023 raw bits in creating a 278-bit key [39]. The work of Guajardo et al. [39] uses a derivative of power-up SRAM state as a secret key; however, it requires an error correction code and imposes SRAM space overhead. Maes et al. [73] introduce an SRAM helper data algorithm to mask unreliable bits using low-overhead post-processing algorithms. Recently, Yu et al. [131] proposed a method of error correction for PUFs using a new syndrome coding scheme to minimize the information leaked by the error correction codes, and Hiller et al. extend this approach for SRAM PUFs [46]. Van Herrewege et al. [124] have designed a new lightweight authentication scheme using PUFs that does not require the reader to store a large number of PUF challenge and response pairs.

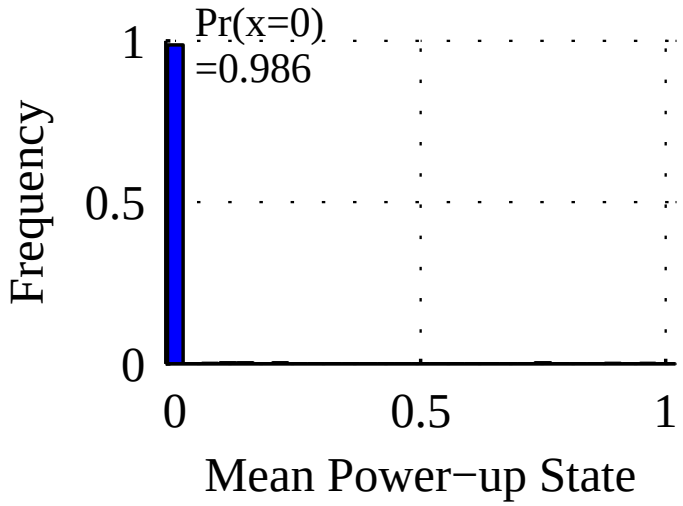
Given the low cost of the several bytes of SRAM that are used for DRV fingerprinting, a relatively significant practical cost may be associated with the generation of the test voltages for characterizing the DRVs. Emerging devices such as computational RFIDs [94] can use software routines to extract DRVs, but as contactless devices they must generate all test voltages on-chip. On-chip dynamic control of SRAM supply voltage is assumed in the low-power literature at least since work on drowsy caches [30]. Supply voltage tuning has also been applied with canary cells to detect potential SRAM failures, and as a post-silicon technique to compensate for process variation and increase manufacturing yields [79].

4.6 Conclusions and Future Works

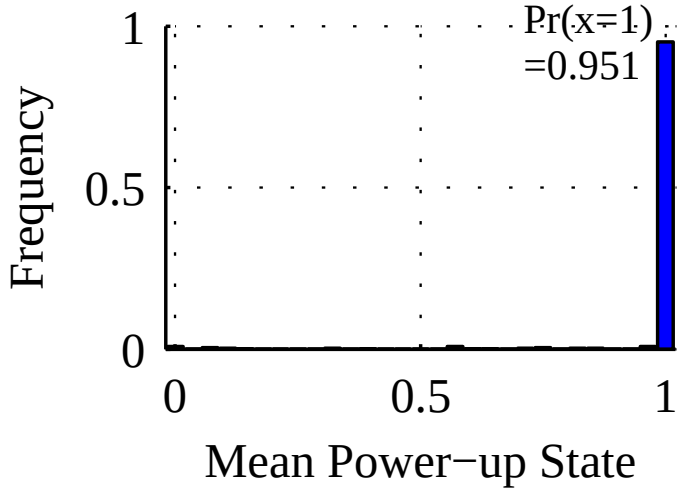
This work has demonstrated that SRAM DRV fingerprints are static identifiers of a device, and it has presented a simple characterization procedure and matching algo-

rithms to use them as such. DRV fingerprints are similar to previously demonstrated power-up fingerprints, but they provide a more informative non-binary identifier of each cell. As a result of this, DRV fingerprints are identified up to 28% more reliably than are power-up fingerprints.

The practical limits of DRV fingerprint performance and reliability should be explored further. Within the constraints of acceptable precision, the runtime of the characterization procedure can be reduced by increasing the voltage step size Δ and reducing the time t_{wait} spent at each voltage (Eq. 4.1). An expanded evaluation could investigate the reliability of DRV fingerprints across a larger variety of devices and a range of environmental conditions. A high reliability could make DRV fingerprints suitable as a basis for key-generation with lightweight error correcting codes.



(a) Strongly 0 DRV



(b) Strongly 1 DRV

Figure 4.5: The plot at left shows that 98.6% of SRAM cells that produce a strongly 0 DRV reliably power-up to state 0, as observed by a mean power-up state of 0. The plot at right shows that 95.1% of cells with strongly 1 DRVs reliably power-up to state 1. The DRV is from a single trial of the cell, and the mean power-up state is measured over 28 power-up trials.

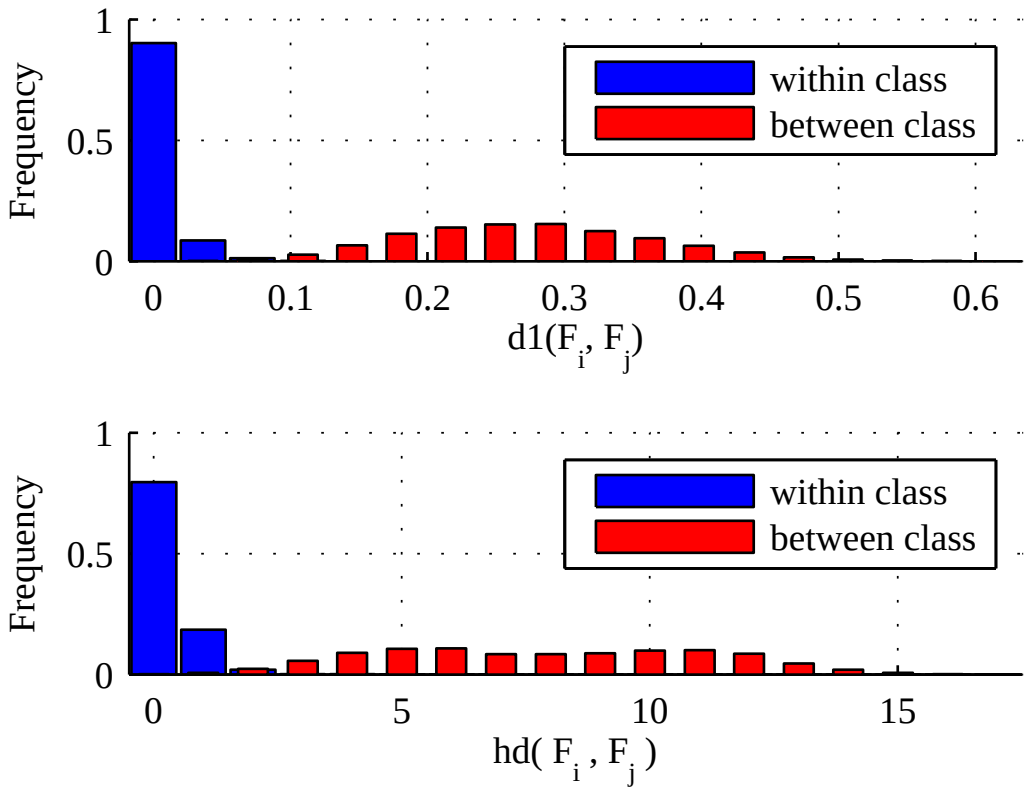


Figure 4.6: Within-class and between-class distances of 16-bit fingerprints. The upper plot uses DRV fingerprints with distance metric $d1$ from Eq. 4.3. The lower plot uses power-up fingerprints with Hamming distance as a metric.

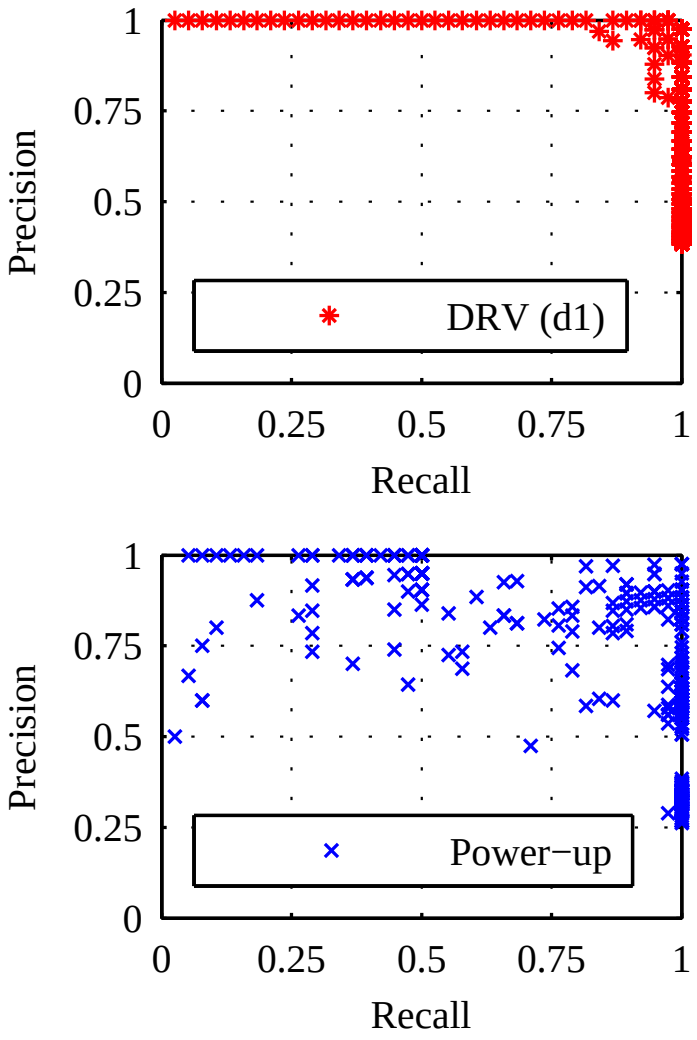


Figure 4.7: Tradeoff points of precision and recall for trials of DRV fingerprints are generally closer to the ideal result of perfect precision and recall.

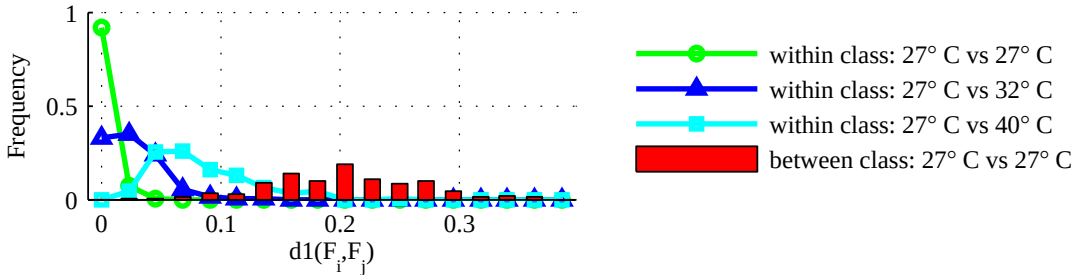


Figure 4.8: The line plots show within-class distances when one fingerprint observation is made at $27°C$ and the second at $27°C$, $32°C$, or $40°C$; within-class distances increase with temperature, implying a diminished reliability. The bar plot shows between-class distances of 16-bit fingerprints taken at $27°C$. Because there does not exist a distance threshold that can separate the two classes when temperature is varied, it may be necessary to use larger fingerprints for reliable identification.

CHAPTER V

Survey of Related Work

5.1 Physical Unclonable Functions

In TARDIS, we use the same principle used in Physical Uncloneable Functions (PUFS) to integrate a notion of time in transiently powered embedded devices. A wide variety of PUFs and fingerprints based on custom or pre-existing integrated circuit components are known. The identifying features used by custom designs include MOSFET drain-current [70], timing race conditions [34], and the digital state taken by cross-coupled logic after a reset [109]. IC identification based on pre-existing circuitry is demonstrated using SRAM power-up state [49, 39], and physical variations of flash memory [88]. Lee et al. [61] derive a secret key unique to each IC using the statistical delay variations of wires and transistors across ICs. Bhargava et al. explore circuit-level techniques for increasing the reliability of SRAM PUFs [13]. Skorobogatov [107] provide an experimental evaluation of low-temperature data remanence on a variety of SRAMs. Saxena and Voris study SRAM remanence in RFID has as a limitation to SRAM-based true random number generation [104].

Previous work [123, 98] use error correction to construct secret keys from noisy PUF sources, however, this is expensive in terms of gates and other resources. To give an idea of the cost of error correction, BoseChaudhuriHocquenghem (BCH) codes [16] used with PUFs include one to correct 21 errors among 127 raw bits in creating a

64-bit key [110], and to correct 102 errors among 1023 raw bits in creating a 278-bit key [39]. Guajardo et al. [39] use a derivative of power-up SRAM state as a secret key, but requires an error correction code and imposes SRAM space overhead. Maes et al. [73] introduce an SRAM helper data algorithm to mask unreliable bits using low-overhead post-processing algorithms. Yu et al. [131] propose a method of error correction for PUFs using a new syndrome coding scheme to minimize the information leaked by the error correction codes. Hiller et al. extend this approach for SRAM PUFs [46]. Van Herrewege et al. [124] present a new lightweight authentication scheme using PUFs that does not require the reader to store a large number of PUF challenge and response pairs.

5.2 Approximate DRAM

Approximate storage in DRAM trades precision for power by extending the refresh period beyond the point of first failure. This was first demonstrated in Flikker [69], a system that partitions DRAM into approximate and exact storage, and uses a longer refresh period for the approximate storage. In Flikker, the refresh period for the approximate data is chosen so that, across all temperatures, the bit error rate remains safely below a given threshold. Our proposal differs from this in proposing to adaptively control refresh period based on temperature, so that at each temperature the refresh period is chosen to keep bit error rate just below the given threshold. Approximate storage in multi-level phase-change memory cells is also proposed [101] as a power saving technique. Enerj [102] is a programming framework for approximate computing in which variables are labeled as exact or approximable. ISA extensions for approximate operations and storage based on dual-voltage operation are proposed by Esmaeilzadeh [28].

DRAM temperature is a function of its power consumption, ambient air temperature, and thermal capacitances and convective resistances of the chip and heat

spreader. Thermal emergencies occur when temperatures exceed safe operating conditions, and avoiding thermal emergencies is particularly challenging in heavily used high performance DRAM as found in servers. Thermal emergencies are local to a single DIMM, and measurements in servers show that temperature can vary by 10°C across DIMMs on account of the different airflow and usage of the respective DIMMs [68]. Numerous techniques exist for predicting and avoiding thermal emergencies by grouping and migrating storage [9, 68, 64] or throttling processor cores to reduce DRAM usage [65]. Techniques for resolving or avoiding thermal emergencies can be evaluated by modifying simulators to include thermal DRAM models encompassing the relevant capacitances and resistances; the model inputs are the power consumption over time, inferred from the DRAM usage [64, 9, 65]. The temperature-dependence of DRAM decay is exploited in the cold-boot attack [42] to draw secret keys from memory.

5.3 DRAM Energy Saving Techniques

Various techniques has been suggested to reduce the energy usage of DRAM. In this section we highlight some of the more prominent techniques:

RAPID [125]: In RAPID, Venkatesan et al. presented three power-saving techniques. The key idea behind their work was to prioritize pages with longer retention time in memory allocation. RAPID-1 increases the minimum required refresh time by removing the 1% worse performing pages, while RAPID-2 and RAPID-3 bin pages according to their retention times and prioritize the allocation of shorter-retaining pages. RAPID-3 also used migration to move data from lower retention pages to higher ones when space becomes available. To evaluate their method, the authors first ran a series of experiment on pages retention time at 3 temperatures using a heatgun and later used this data in a mathematical model to analyze the effect of their techniques.

RAIDR [67]: RAIDR grouped rows into different bins according to their retention time and used Bloom filters to increase the efficiency of their bins implementation. Their evaluation is simulation based and relies on the retention distribution parameters provided by Kim et al. [56].

Mosaic [4]: Mosaic builds an advanced mathematical model of embedded DRAM based on the assumption of spatial locality and retention distribution parameters of Kim et al. [56]. Using this model, it divides the eDRAM module into regions and program their refresh requirements in counters in the cache controller.

RIO & PARIS [10]: In their work, Baek et al. propose two techniques for energy saving in DRAM. In RIO, a system similar to RAPID-1 that logically deletes highly volatile page frames but their approach targets less than 0.1% of cells. They argue that most of the benefits that can be achieved by removing 1% of cells are gained in their system with much less space overhead. To evaluate their technique the authors modified the Linux OS to adjust refresh period and delete pages that have weak cells in their experimental platform. Their other technique, PARIS, works by excluding DRAM rows that contain no data from refresh similar to predecessor techniques such as ESKIMO [51]. In their system, DRAM controller has to provide a way for the OS to send information about used memory rows. Because of the overhead of hardware modification, they evaluated PARIS by implementing it as a Linux loadable kernel module. Their work also considers the effect of temperature and evaluates their system in temperature ranging from $45 - 85°C$ but also unnecessarily consider that DRAM cells respond to temperature variations differently.

Flikker [69]: Approximate storage in DRAM trades precision for power by extending the refresh period beyond the point of first failure. In Flikker, the authors partitioned DRAM into approximate and exact storage, and uses a longer refresh period for the

approximate storage. Their evaluation uses measurements done by Bhalodia [12] for simulation but ignores the effect of temperature by only considering their performance at 48°C.

DTAIL [22]: In DTAIL Cui et al. Use similar binning technique suggested in previous works but place retention data of DRAM in itself. Their key observation is that required data for each refresh decision is small and data from adjacent rows are used sequentially. Their evaluation is based on simulation and uses Baek et al. [10] and Kim et al. [56] as basis for its model. Their work also only considers 85°C for its evaluation.

We previously described how some memory systems use temperature compensated refresh to adjust refresh rate in higher temperatures. The increase in refresh rate can itself be the source of creating more heat that can move DRAM out of its operational range and into a thermal emergency. Thermal emergencies occur when temperatures exceed safe operating conditions, and avoiding thermal emergencies is particularly challenging in heavily used high performance DRAM as found in servers. Thermal emergencies are local to a single DIMM, and measurements in servers show that temperature can vary by 10°C across DIMMs on account of the different airflow and usage of the respective DIMMs [68]. Numerous techniques exist for predicting and avoiding thermal emergencies by grouping and migrating storage [9, 68, 64] or throttling processor cores to reduce DRAM usage [65]. Techniques for resolving or avoiding thermal emergencies can be evaluated by modifying simulators to include models of DRAM thermal capacitances and convective resistances; the model inputs are the power consumption over time, inferred from the DRAM usage [64, 9, 65]. Because these works focus on avoiding errors entirely, they are agnostic to spatial distributions of DRAM errors.

An alternative to traditional DRAM is embedded DRAM (eDRAM). eDRAM differs from DRAM in that it is fabricated using standard logic process technology and can therefore be implemented on the same die as logic for a memory that is higher density than SRAM, but higher bandwidth and lower latency than off-chip DRAM. The retention time of eDRAM, on the order of tens of microseconds [11] nominally or milliseconds in a low-power process [62], is considerably lower than that of off-chip DRAM. Wilkerson et al. [128] propose using a less frequent refresh and a stronger error correcting code to correct and avoid errors from bad cells in eDRAM.

CHAPTER VI

Future Work & Conclusion

In this chapter, we look at future research directions based on this research and conclude by highlighting the contributions presented in this dissertation.

6.1 Future Work

6.1.1 Security of Emerging IoT Platforms

The proliferation of computing capabilities in everyday devices have given rise to a phenomenon known as the Internet of Things (IoT). There are currently an estimated 6.4 billion IoT devices worldwide. This figure is expected to rise to more than 20 billion by the year 2020 [33]. IoT allows for integration and collaboration of individual embedded devices with each other and the cloud. This connection creates opportunity to provide useful functions and services to the user, but also create new security and privacy risks. There are many challenges with securing the IoT environment. First and foremost, devices in an IoT environment usually suffer from limited computational capability, energy availability, or communication ability. These limitations hinder implementation of many security practices and protocols in these devices. Second, the ubiquity of these systems has dramatically increased the amount of data and functions accessible to malicious parties if they infiltrate these systems. Moving from

software-based security solutions to hardware-based approaches can play an important role in improving the security and reliability of these systems.

6.1.2 Security of Approximate Computing Systems

Approximate computing systems continue to draw the attention of researchers especially in the computer architecture community. While our work studied the privacy implications of using approximate memory, similar studies can be conducted on other approximate computing components such as processors. Furthermore, one of approximate computing's main goal is to reduce the power consumption of the system by accepting certain level of error. As such, these systems may be more vulnerable to physical attacks such as side-channel or error injection.

6.1.3 Trusted Execution Environments

In recent years, hardware-supported Trusted Execution Environments such as Intel SGX [50] and ARM TrustZone [6] have become main-stream in commodity systems, mobiles, and cloud platforms. Trusted Execution Environments provide developers with features such as isolated execution, secure storage, remote attestation, secure provisioning, and trusted execution path. These platforms provide a great opportunity for researchers and application developers to build systems with higher security guarantees that do not rely upon a trusted operating system. Furthermore, improving performance of these systems, their vulnerability to side-channel attacks, and building a resilient and flexible root of trust for them remain as open problems.

6.2 Concluding Remarks

Computer systems security has traditionally focused on software mechanisms to implement features and provide security, privacy, and confidentiality guarantees in computer systems. In recent years, however, the growing availability of specialized

hardware support (*e.g.,* Trusted Platform Modules) has increased attention to hardware side-channels. Moreover, new computing paradigms such as Internet of Things and approximate computing that are more closely intertwined with the hardware has shifted attention to use of underlying hardware as both source of attack and defenses in computer systems.

This dissertation explored how memory remanence, an often ignored hardware effect, affects system security, forming the basis for both attack and defense mechanisms. To this end, we developed experimental platforms and testing methodology that enabled us to study the memory remanence effect in SRAM, DRAM, and Flash memory. Using these platforms, we developed three systems that showcased the potential of memory remanence as an attack and defense vector. First, in Probable Cause, we examined the effect of approximate computing on data processing. We showed how data passing through an approximate memory is watermarked with a unique, device specific, error pattern that compromises data anonymity by uniquely matching the data with its creator. Second, we looked at the problem of time-keeping in transiently powered systems. Because these devices lack a consistent source of energy, they are unable to run a clock that comprehends the passage of time in their unpowered state. TARDIS enables these devices to maintain a coarse notion of time by using the gradual decay of data in volatile memory as an estimator for the passing time. Finally, we looked at data-retention voltage of memory cells as an identifier that enables the creation of unforgeable fingerprints and identifiers in DRV-Fingerprint. We show that use of data-retention voltage improves identification by 28% over traditional techniques. These results showcase the potential effect of ignored hardware effect on security and privacy of computer systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] The TARDIS, British Broadcasting Channel. `http://www.bbc.co.uk/doctorwho/characters/tardis.shtml`, November 1963.

[2] Hpc0402b/c - high performance, high precision wire-bondable 0402 capacitor for smartcard, high-frequency and substrate-embedded applications. `http://www.vishay.com/docs/10120/hpc0402b.pdf`, December 2008.

[3] An introduction to the architecture of Moo 1.0. `https://spqr.cs.umass.edu/moo/Documents/Moo_01242011.pdf`, May 2011.

[4] Aditya Agrawal, Amin Ansari, and Josep Torrellas. Mosaic: Exploiting the spatial locality of process variation to reduce refresh energy in on-chip edram modules. In *International Symposium on High Performance Computer Architecture*, 2014.

[5] R Anderson and M Kuhn. Tamper resistance: a cautionary note. *Proceedings of the Second Usenix Workshop on Electronic Commerce*, 1996.

[6] ARM. Arm trustzone. `https://www.arm.com/products/security-on-arm/trustzone`.

[7] Gildas Avoine. Personal communication on French passports. 2012.

[8] Gildas Avoine, Kassem Kalach, and Jean-Jacques Quisquater. ePassport: Securing international contacts with contactless chips. In Gene Tsudik, editor, *Financial Cryptography and Data Security*, pages 141–155. Springer-Verlag, 2008.

[9] Raid Zuhair Ayoub, Krishnam Raju Indukuri, and Tajana Simunic Rosing. Energy efficient proactive thermal management in memory subsystem. In *ISLPED '10: Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, August 2010.

[10] Seungjae Baek, Sangyeun Cho, and Rami Melhem. Refresh now and then. 2013.

[11] J Barth, W R Reohr, P Parries, G Fredeman, J Golz, S E Schuster, Richard E Matick, H Hunter, C C Tanner, J Harig, H Kim, B A Khan, J Griesemer, R P Havreluk, K Yanagisawa, T Kirihata, and S S Iyer. A 500 MHz Random Cycle, 1.5 ns Latency, SOI Embedded DRAM Macro Featuring a Three-Transistor Micro Sense Amplifier. *Solid-State Circuits, IEEE Journal of*, 43(1):86–95, 2008.

[12] Vimal Bhalodia. SCALE DRAM subsystem power analysis. Master's thesis, Massachusetts Institute of Technology, 2005.

[13] Mudit Bhargava, Cagla Cakir, and Ken Mai. Reliability enhancement of bi-stable PUFs in 65nm bulk CMOS. *International Symposium on Hardware-Oriented Security and Trust*, 2012.

[14] Stephen C. Bono, Matthew Green, Adam Stubblefield, Ari Juels, Aviel D. Rubin, and Michael Szydlo. Security analysis of a cryptographically-enabled RFID device. In *Proceedings of the 14th USENIX Security Symposium*, 2005.

[15] Steve Bono, February 2012. Personal communication.

[16] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[17] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. Wireless device identification with radiometric signatures. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, September 2008.

[18] Michael Buettner, Ben Greenstein, David Wetherall, and Joshua R. Smith. Revisiting smart dust with RFID sensor networks, 2008.

[19] Adam C Cabe, Zhenyu Qi, and Mircea R Stan. Stacking SRAM banks for ultra low power standby mode operation. In *Design Automation Conference*, June 2010.

[20] Cantherm. Thermal cut-offs. `http://www.cantherm.com/products/thermal_fuses/sdf.html`, 2011. Last Viewed May 14, 2012.

[21] Tom Chothia and Vitaliy Smirnov. A traceability attack against e-Passports. In *14th International Conference on Financial Cryptography and Data Security*. Springer, 2010.

[22] Zehan Cui, Sally A McKee, Zhongbin Zha, Yungang Bao, and Mingyu Chen. DTail: A Flexible Approach to DRAM Refresh Management . In *Proceedings of the 28th ACM international conference on Supercomputing*, pages 43–52, New York, New York, USA, 2014. ACM Press.

[23] Kate Cumming. Purposeful data: the roles and purposes of recordkeeping metadata. *Records Management Journal*, 17(3):186–200, 2007.

[24] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. Memory power management via dynamic voltage/frequency scaling. In *International Conference on Autonomic Computing*, ICAC, pages 31–40, New York, NY, USA, 2011. ACM.

[25] Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F Wenisch, and Ricardo Bianchini. Memscale: active low-power modes for main memory. *ACM SIGPLAN Notices*, 2011.

[26] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[27] EPCglobal. *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHZ–960 MHZ, Version 1.2.0.*

[28] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Architecture support for disciplined approximate programming. In *ACM SIGARCH Computer Architecture News*. ACM, 2012.

[29] K. Flautner, Nam Sung Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Proc. 29th IEEE/ACM International Symposium on Computer Architecture*, pages 148–157, 2002.

[30] K Flautner, NS Kim, and S Martin. Drowsy caches: simple techniques for reducing leakage power. *International Symposium on Computer Architecture*, 2002.

[31] Saurabh Ganeriwal, Srdjan Čapkun, Chih-Chieh Han, and Mani B. Srivastava. Secure time synchronization service for sensor networks. In *Proceedings of the 4th ACM Workshop on Wireless Security*, WiSe '05, pages 97–106, 2005.

[32] Flavio D. Garcia, P. van Rossum, R. Verdult, and R.W. Schreur. Wirelessly pickpocketing a MIFARE Classic card. In *IEEE Symposium on Security and Privacy*, pages 3–15, May 2009.

[33] Gartner, Inc. Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015. `https://www.gartner.com/newsroom/id/3165317`.

[34] B Gassend, D Clarke, and M Van Dijk. Silicon physical random functions. In *Proceedings of the IEEE Computer and Communications Society*, 2002.

[35] Blaise Gassend. Physical Random Functions. Master's thesis, MIT, USA, 2003.

[36] Zeno J Geradts, Jurrien Bijhold, Martijn Kieft, Kenji Kurosawa, Kenro Kuroki, and Naoki Saitoh. Methods for identification of images acquired with digital cameras. In *Enabling Technologies for Law Enforcement*, pages 505–512. International Society for Optics and Photonics, 2001.

[37] Ian Goldberg and Marc Bricenco. GSM cloning. `http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html`, 1999. Last Viewed February 19, 2012.

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

[39] J Guajardo, S Kumar, GJ Schrijen, and P Tuyls. FPGA intrinsic PUFs and their use for IP protection. *Cryptographic Hardware and Embedded Systems*, 2007.

[40] P Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Proceedings of the 6th USENIX Security Symposium*, Jan 1996.

[41] J Halderman, S Schoen, N Heninger, W Clarkson, W Paul, J Calandrino, A Feldman, J Appelbaum, and E Felten. Lest we remember: Cold boot attacks on encryption keys. In *Proceedings of the 17th USENIX Security Symposium*, 2008.

[42] J Alex Halderman, Seth D Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A Calandrino, Ariel J Feldman, Jacob Appelbaum, and Edward W Felten. Lest we remember: cold-boot attacks on encryption keys. *Communications of the ACM*, 2009.

[43] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the 29th Annual IEEE Symposium on Security and Privacy*, pages 129–142, May 2008.

[44] Takeshi Hamamoto, Soichi Sugiura, and Shizuo Sawada. On the retention time distribution of dynamic random access memory (DRAM). *IEEE Transactions on Electron Devices*, 1998.

[45] Thomas S. Heydt-Benjamin, Dan V. Bailey, Kevin Fu, Ari Juels, and Tom OHare. Vulnerabilities in first-generation RFID-enabled credit cards. In *Proceedings of Eleventh International Conference on Financial Cryptography and Data Security, Lecture Notes in Computer Science, Vol. 4886*, pages 2–14, February 2007.

[46] Matthias Hiller, Dominik Merli, Frederic Stumpf, and Georg Sigl. Complementary IBS: Application specific error correction for PUFs. *International Symposium on Hardware-Oriented Security and Trust*, 2012.

[47] H. Ho, E. Saeedi, S.S. Kim, T.T. Shen, and B.A. Parviz. Contact lens with integrated inorganic semiconductor devices. In *Micro Electro Mechanical Systems, 2008. MEMS 2008. IEEE 21st International Conference on*, pages 403–406, January 2008.

[48] D Holcomb, WP 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.

[49] Daniel E Holcomb, Wayne P Burleson, and K Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers*, 2009.

[50] Intel. Intel software guard extensions. `https://software.intel.com/en-us/sgx`.

[51] Ciji Isen and Lizy John. Eskimo-energy savings using semantic knowledge of inconsequential memory occupancy for dram subsystem. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 337–346. IEEE, 2009.

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

[53] JEDEC. LPDDR2 SDRAM Specification, 2010.

[54] Ari Juels. Minimalist cryptography for low-cost RFID tags (extended abstract). In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2005.

[55] Ari Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, February 2006.

[56] Kinam Kim and Jooyoung Lee. A new investigation of data retention time in truly nanoscaled drams. *Electron Device Letters, IEEE*, 30(8):846–848, 2009.

[57] T Kohno, A Broido, and K Claffy. Remote physical device fingerprinting. *Security and Privacy, 2005 IEEE Symposium on*, pages 211–225, 2005.

[58] Kenji Kurosawa, Kenro Kuroki, and Naoki Saitoh. Ccd fingerprint method-identification of a video camera from videotaped images. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 3, pages 537–540. IEEE, 1999.

[59] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[60] Thanh-Ha Le, Jessy Clédière, Christine Servière, and J-L Lacoume. Noise reduction in side channel attack using fourth-order cumulant. *Information Forensics and Security, IEEE Transactions on*, 2(4):710–720, 2007.

[61] 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.*, pages 176 – 179, June 2004.

[62] Yoonmyung Lee, Mao-Ter Chen, Junsun Park, D Sylvester, and David T Blaauw. A 5.42nW/kB retention power logic-compatible embedded DRAM with 2T dual-Vt gain cell for low power sensing applications. In *Solid State Circuits Conference (A-SSCC), 2010 IEEE Asian*, pages 1–4, 2010.

[63] Peter H. Lewis. Of privacy and security: The clipper chip debate. The New York Times, April 24, 1994.

[64] Chung-Hsiang Lin, Chia-Lin Yang, and Ku-Jei King. PPT: joint performance/power/thermal management of DRAM memory for multi-core systems. In *ISLPED '09: Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, August 2009.

[65] Jiang Lin, Hongzhong Zheng, Zhichun Zhu, Howard David, and Zhao Zhang. Thermal modeling and management of DRAM memory systems. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, June 2007.

[66] Y Lin, Dennis M Sylvester, and David T Blaauw. A sub-pW timer using gate leakage for ultra low-power sub-Hz monitoring systems. *Custom Integrated Circuits Conference*, 2007.

[67] Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu. Raidr: Retention-aware intelligent dram refresh. In *International Symposium on Computer Architecture*, ISCA, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society.

[68] Song 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.

[69] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G Zorn. Flikker: saving DRAM refresh-power through critical data partitioning. In *ASPLOS XVI: Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, June 2012.

[70] K. Lofstrom, W.R. Daasch, and D. Taylor. IC identification circuit using device mismatch. In *IEEE International Solid-State Circuits Conference. Digest of Technical Papers.*, pages 372 –373, 2000.

[71] J Lukas, J Fridrich, and M Goljan. Digital camera identification from sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, 1(2):205–214, 2006.

[72] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

[73] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. *Cryptographic Hardware and Embedded Security*, 2009.

[74] Wenbo Mao. Timed-release cryptography. In *Selected Areas in Cryptography VIII (SAC'01*, pages 342–357. Prentice Hall, 2001.

[75] Gary McGraw. Silver bullet podcast: Interview with Ross Anderson. http://www.cigital.com/silver-bullet/show-070/. Show #70, January 31, 2012.

[76] Micron. *DDR2 SDRAM SODIMM: MT8KTF51264HZ 4GB*, 2011.

[77] Micron Technology Inc. Mobile DRAM Power-Saving Features and Power Calculations. 2005.

[78] Nicholas Nethercote and Julian Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. In *ACM Sigplan Notices*, 2007.

[79] Afshin Nourivand, Asim J Al-Khalili, and Yvon Savaria. Postsilicon Tuning of Standby Supply Voltage in SRAMs to Reduce Yield Losses Due to Parametric Data-Retention Failures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, (1):29–41, 2011.

[80] NXP Semiconductors MIFARE classic. http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/mifare_classic/. Last Viewed February 18, 2012.

[81] NXP Semiconductors SPI real time clock/calendar. http://www.nxp.com/documents/data_sheet/PCF2123.pdf. Last Viewed February 18, 2012.

[82] Inc. Omega Engineering. *OSXL450 Infrared Non-Contact Thermometer Manual*.

[83] OpenCores.org. Openrisc or1200 processor. http://opencores.org/or1k/OR1200_OpenRISC_Processor.

[84] Y. Oren and A. Shamir. Remote password extraction from RFID tags. *Computers, IEEE Transactions on*, 56(9):1292–1296, September 2007.

[85] David Oswald and Christof Paar. Breaking MIFARE DESFire MF3ICD40: Power analysis and templates in the real world. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 207–222, 2011.

[86] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymitya proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.

[87] A C Polak, S Dolatshahi, and D L Goeckel. Identifying Wireless Users via Transmitter Imperfections. *Selected Areas in Communications, IEEE Journal on*, 29(7):1469–1479, 2011.

[88] P Prabhu, A Akel, L Grupp, WK Yu, G Suh, Edwin Kan, and Steven Swanson. Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations. *Proceedings of the 4th International Conference on Trust and Trustworthy Computing*, 2011.

[89] Hulfang Qin, Yu Cao, D Markovic, A Vladimirescu, and J Rabaey. SRAM leakage suppression by minimizing standby supply voltage. In *5th International Symposium on Quality Electronic Design.*, pages 55–60, 2004.

[90] Amir Rahmati, Matthew Hicks, Daniel E. Holcomb, and Kevin Fu. Refreshing thoughts on DRAM: Power saving vs. data integrity. In *Workshop on Approximate Computing Across the System Stack (WACAS)*, Salt Lake City, UT, March 2014. To appear.

[91] Amir Rahmati, Mastooreh Salajegheh, Dan Holcomb, Jacob Sorber, Wayne P. Burleson, and Kevin 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*, Security '12, Bellevue, WA, August 2012.

[92] Murugavel Raju. UltraLow Power RC Timer Implementation using MSP430. In *Texas Instruments Application Report SLAA119*, 2000.

[93] Benjamin Ransford, Shane Clark, Mastooreh Salajegheh, and Kevin Fu. Getting things done on computational RFIDs with energy-aware checkpointing and voltage-aware scheduling. In *USENIX Workshop on Power Aware Computing and Systems (HotPower '08)*, December 2008.

[94] Benjamin Ransford, Shane Clark, Mastooreh Salajegheh, and Kevin Fu. Getting things done on computational RFIDs with energy-aware checkpointing and voltage-aware scheduling. In *USENIX Workshop on Power Aware Computing and Systems (HotPower)*, December 2008.

[95] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA, 1996.

[96] 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. *Solid-State Circuits, IEEE Journal of*, (99):1–10, 2013.

[97] Ludovic Rousseau. Secure time in a portable device. In *Gemplus Developer Conference*, 2001.

[98] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. *Enhancing RFID Security and Privacy by Physically Unclonable Functions*, pages 281–307. Information Security and Cryptography. Springer, Sep 2010.

[99] Mastooreh Salajegheh, Yue Wang, Kevin Fu, Anxiao Jiang, and Erik G Learned-Miller. Exploiting half-wits: smarter storage for low-power devices. In *9th USENIX Conference on File and Storage Technologies*, 2011.

[100] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. Design of an RFID-based battery-free programmable

sensing platform. *IEEE Transactions on Instrumentation and Measurement*, 57(11):2608–2615, November 2008.

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

[102] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. Enerj: Approximate data types for safe and general low-power computation. In *PLDI '11: Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*, 2011.

[103] Samsung Electronics. KM41464A NMOS DRAM.

[104] Nitesh Saxena and Jonathan Voris. We can remember it for you wholesale: Implications of data remanence on the use of RAM for true random number generation on RFID tags. *Proceedings of the Conference on RFID Security*, 2009.

[105] Bruce Schneier. *Applied cryptography (2nd ed.): Protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., 1995.

[106] Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, and Dan Boneh. On the effectiveness of address-space randomization. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS)*, 2004.

[107] S Skorobogatov. Low temperature data remanence in static RAM. Technical Report UCAM-CL-TR-536, University of Cambridge Computer Laboratory, 2002.

[108] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In Bruce Christianson, Bruno Crispo, James Malcolm, and Michael Roe, editors, *Security Protocols*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–182. Springer, 2000.

[109] Ying Su, J. Holleman, and B.P. Otis. A digital 1.6 pj/bit chip identification circuit using process variations. *IEEE Journal of Solid-State Circuits*, 43(1):69 –77, Jan. 2008.

[110] G.E. Suh, C.W. O'Donnell, and S. Devadas. AEGIS: a single-chip secure processor. *IEEE Design & Test of Computers*, 24(6):570 –580, Nov.-Dec. 2007.

[111] Kun Sun, Peng Ning, and Cliff Wang. TinySeRSync: secure and resilient time synchronization in wireless sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 264–277, 2006.

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

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

[114] RM Swanson and James D Meindl. Ion-implanted complementary MOS transistors in low-voltage circuits. *International Solid-State Circuits Conference*, May 1972.

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

[116] R Tessier, D Jasinski, A Maheshwari, Aiyappan Natarajan, Weifeng Xu, and Wayne Burleson. An energy-aware active smart card. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, 2005.

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

[118] Texas Instruments Inc. MSP430 hardware tools. In *Texas Instruments User's Guide*, May. 2009, revised Feb. 2014.

[119] Texas Instruments Inc. MSP430F21x1 Mixed Signal Microcontroller. In *Texas Instruments Application Report SLAS439F*, Sep. 2004, revised Aug. 2011.

[120] ThingMagic Inc. *Mercury 4/ MERCURY 5 User Guide*, February 2007.

[121] David Tschumperlé. The CImg library. In *IPOL Meeting on Image Processing Libraries*, 2012.

[122] T Tuan, T Strader, and S Trimberger. Analysis of data remanence in a 90nm FPGA. *Custom Integrated Circuits Conference*, 2007.

[123] Pim Tuyls and Lejla Batina. RFID-tags for anti-counterfeiting. In *Topics in Cryptology - CT-RSA 2006, volume 3860 of LNCS*, pages 115–131. Springer Verlag, 2006.

[124] Anthony van Herrewege, Stefan Katzenbeisser, Roel Maes, Roel Peeters, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, , and Christian Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs. In *Financial Cryptography (FC) 2012*, LNCS. Springer, Feb 2012.

[125] Ravi K Venkatesan, Stephen Herr, and Eric Rotenberg. Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM. In *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*. IEEE, 2006.

[126] E Vittoz. Low-power design: Ways to approach the limits. *International Solid-State Circuits Conference*, May 1994.

[127] Jiajing Wang and Benton Highsmith Calhoun. Techniques to Extend Canary-Based Standby VDD Scaling for SRAMs to 45 nm and Beyond. *IEEE Journal of Solid-State Circuits*, 43(11):2514–2523, 2008.

[128] Chris Wilkerson, Alaa R Alameldeen, Zeshan Chishti, Wei Wu, Dinesh Somasekhar, Shih-lien Lu, Chris Wilkerson, Alaa R Alameldeen, Zeshan Chishti, Wei Wu, Dinesh Somasekhar, and Shih-lien Lu. Reducing cache power with low-cost, multi-bit error-correcting codes. *ACM SIGARCH Computer Architecture News*, 38(3):83–93, June 2010.

[129] Xunteng Xu, Lin Gu, Jianping Wang, and Guoliang Xing. Negotiate power and performance in the reality of RFID systems. In *PerCom*, pages 88–97. IEEE Computer Society, 2010.

[130] D. Yeager, Fan Zhang, A. Zarrasvand, N.T. George, T. Daniel, and B.P. Otis. A 9 μa, addressable Gen2 sensor tag for biosignal acquisition. *IEEE Journal of Solid-State Circuits*, 45(10):2198–2209, October 2010.

[131] Meng-Day Yu and S Devadas. Secure and Robust Error Correction for Physical Unclonable Functions. *IEEE Design & Test of Computers*, 27(1):48–65, 2010.

[132] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. Moo: A batteryless computational RFID and sensing platform. Technical Report UM-CS-2011-020, Department of Computer Science, University of Massachusetts Amherst, Amherst, MA, June 2011.