SOFTWARE TECHNIQUES TO REDUCE THE ENERGY CONSUMPTION OF LOW-POWER DEVICES AT THE LIMITS OF DIGITAL ABSTRACTIONS

A Dissertation Presented

by

MASTOOREH (NEGIN) SALAJEGHEH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2012

Department of Computer Science

SOFTWARE TECHNIQUES TO REDUCE THE ENERGY CONSUMPTION OF LOW-POWER DEVICES AT THE LIMITS OF DIGITAL ABSTRACTIONS

A Dissertation Presented

by

MASTOOREH (NEGIN) SALAJEGHEH

Approved as to style and content by:

Kevin Fu, Chair

Wayne Burleson, Member

Deepak Ganesan, Member

Erik Learned-Miller, Member

Lori Clarke, Department Chair
Department of Computer Science

This page is intentionally left blank.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Kevin Fu, for inspiring me and encouraging me for the past five years. I thank him for teaching me how to think, write, and present and I still have a lot to learn from him. I thank him for making me aim high and work hard.

I thank members of my committee, Dr. Wayne Burleson, Dr. Erik Learned-Miller, and Dr. Deepak Ganesan whose advice, feedback, and encouragement have been invaluable.

This thesis would not have been possible without the collaborative work, time, advice, and support of my colleagues in the SPQR lab: Ben Ransford, Shane Clark, Andres Molina, Hong Zhang, Amir Rahmati, Shane Guineau, and Benessa Defend as well as my fantastic collaborators outside of SPQR Lab: Jacob Sorber and Dan Holcomb. I also thank all the students in "Lab" lab for making my life more fun in the lab.

I thank Wendy Cooper for making Halloween scary for me for the first time by setting Halloween the deadline for my dissertation proposal. I also thank her for all of her support and encouragement.

I would like to thank Quinn Stewart for her tremendous help on editing my writings including this thesis.

I would like to acknowledge NSF and UMass CVIP group for funding my graduate studies—of course through my advisor's efforts.

During the past five years in Amherst, I have found many friends who made my life fun and exciting. I thank them all for letting me feel part of a large family.

I greatly thank my parents and my sister who sent me their support and love through phone calls, emails, and Skype screens for the past five years.

Most of all, I would like to thank my husband, Hamed Soroush, who has been my classmate, co-author, and colleague for the past 10 years. Without his non-stop support and encouragement, I would not have been here. He had more confidence in me than I did in myself.

ABSTRACT

SOFTWARE TECHNIQUES TO REDUCE THE ENERGY CONSUMPTION OF LOW-POWER DEVICES AT THE LIMITS OF DIGITAL ABSTRACTIONS

SEPTEMBER 2012

MASTOOREH (NEGIN) SALAJEGHEH

B.Sc., SHARIF UNIVERSITY OF TECHNOLOGY

M.Sc., CARNEGIE MELLON UNIVERSITY, INI

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Kevin Fu

My thesis explores the effectiveness of software techniques that bend digital abstractions in order to allow embedded systems to do more with less energy. Recent years have witnessed a proliferation of low-power embedded devices with power ranges of few milliwatts to microwatts. The capabilities and size of the embedded systems continue to improve dramatically; however, improvements in battery density and energy harvesting have failed to mimic a Moore's law. Thus, energy remains a formidable bottleneck for low-power embedded systems.

Instead of trying to create hardware with ideal energy proportionality, my dissertation evaluates how to use unconventional and probabilistic computing that bends traditional abstractions and interfaces in order to reduce energy consumption while protecting program semantics. My thesis considers four methods that unleash energy

otherwise squandered on communication, storage, time keeping, or sensing: 1) CCCP, which provides an energy-efficient storage alternative to local non-volatile storage by relying on cryptographic backscatter radio communication, 2) Half-Wits, which reduces energy consumption by 30% by allowing operation of embedded systems at below-spec supply voltages and implementing NOR flash memory error recovery in firmware rather than strictly in hardware, 3) TARDIS, which exploits the decay properties of SRAM to estimate the duration of a power failure ranging from seconds to several hours depending on hardware parameters, and 4) Nonsensors, which allow operation of analog to digital converters at low voltages without any hardware modifications to the existing circuitry.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

This page is intentionally left blank.

CHAPTER 1

INTRODUCTION

Recent years have witnessed a proliferation of low-power embedded devices [6, 22, 63, 75] with power ranges of few milliwatts (battery-powered) to microwatts (batteryless) [66]. The capabilities and size of the embedded systems continue to improve dramatically; however, improvements in battery density and energy harvesting have failed to mimic a Moore's law. The battery energy density is the slowest trend in mobile computing and it does not scale exponentially [93]. Thus, energy remains a formidable bottleneck for low-power embedded systems. For example, circuits of a smart lens can be miniaturized enough to be implanted inside an eye [56]; however, its corresponding batteries are not made small enough for such implantation.

1.1 A Vision for a Cleaner World

Every year in United States alone, more than 3 billion batteries are purchased [4], a fraction of which is utilized to power embedded systems ranging from smoke detectors and utility meters to tollway payment transponders and pacemakers. I envision a future in which fewer chemical batteries are purchased and have landed on the earth even though the battery-operated devices continue to become computationally more powerful. Furthermore, batteryless devices are developed and utilized in places that have not been possible before.

Achieving this vision will require advances on many fronts: improvements in battery technologies in terms of energy per density, more productive energy harvesting schemes, better energy-aware programming languages, and new methods to reduce

the energy consumption of existing devices. More fundamentally, we would like to redefine and reexamine design goals and requirements for low-power systems that are not as well equipped as their more powerful descendants, desktop computers.

1.2 Thesis Overview

Rather than trying to create hardware with ideal energy proportionality, the thesis of this work is that we can use unconventional and probabilistic computing to bend traditional abstractions and interfaces in order to reduce energy consumption while protecting program semantics. In support of my thesis, I have designed and evaluated four software techniques that allow both battery-powered and batteryless devices to better exploit their power supplies: The first contribution, CCCP, exploits a passive radio to provide a less energy-hungry alternative to non-volatile storage. The primary challenge to this approach was to preserve data privacy and security. My other two contributions, Half-Wits and nonsensors, make energy consumption proportional to workload by operating the chip below safety margins while preserving data reliability. Finally, the TARDIS provides an hourglass-like timer which utilizes decay properties of volatile memory in an unconventional way to provide a secure notion of time. This timer is not as accurate as a clock but it requires no power to keep time. Table 1.1 summarizes the techniques and challenges to save energy for each of these four contributions.

Contribution	**Technique to Save Energy**	**Research Challenge**
CCCP	Unconventional Use of Radio	Security & Privacy
Half-Wits	Probabilistic Storage	Reliability
TARDIS	Unconventional Use of SRAM	Security and Reliability
Nonsensors	Probabilistic Sensing	Reliability

Table 1.1. My thesis uses unconventional and probabilistic computing that bends traditional abstractions and interfaces in order to reduce energy consumption while protecting program semantics.

1.3 CCCP, Secure Remote Storage for RFID Tags

Research involving low-energy computing systems has long treated radio as an energy-hungry resource to be used sparingly. This work, CCCP, exploits radio as a resource to amplify the storage capabilities of batteryless, programmable RFID tags. The main idea of this work is that a tag can securely use radio communication as a less energy-intensive alternative to local, flash-based storage. The smaller energy requirements of radio (Figure 1.1) allow the RFID tags either to devote more energy to computation or to accomplish the same tasks using less energy, which may translate into a longer operating range. CCCP uses established cryptographic mechanisms to protect against untrustworthy RFID readers that could attempt to violate the confidentiality, authenticity, integrity, and freshness of the data on a programable RFID tag. The main challenge in this work is to design an energy-saving remote storage system that provides security under the constraints of passive RFID systems. Our experiments show that—despite cryptographic overhead—remote checkpointing consumes less energy than checkpointing to flash for data sizes above roughly 64 bytes. CCCP enables secure and flexible remote storage that would otherwise outstrip batteryless RFID tags' resources. This work has been published in USENIX Security'09 [109].

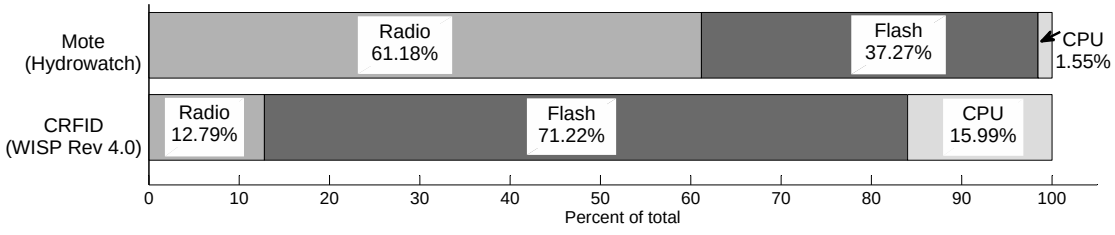


Figure 1.1. Per-component maximum power consumption of two embedded devices. Radio communication on the WISP requires less power than writes to flash memory. The relative magnitudes of the power requirements means that a sensor mote favors shifting storage workloads to local flash memory instead of remote storage via radio, while a Computational RFID (CRFID) favors radio over flash.

1.4 Half-Wits, Non-Volatile Storage at Low Voltage

While perfect data integrity is always an appealing target, system design constraints (e.g., energy scarcity) are likely to push systems towards execution modes in which less-than-perfect data integrity is sometimes tolerated, in exchange for improved power, density [86], or speed [15]. To enable energy-proportional computing for application-specific data, embedded storage systems ought to consider unconventional hardware-software interfaces that allow applications to provide reliability "hints" to hardware. This would allow the hardware to perform in a probabilistic manner that meets the application-level specifications without over-provisioning. The Half-Wits work shows how to reduce energy consumption by 30% by implementing NOR flash memory error recovery in firmware rather than strictly in hardware. The energy savings result from the ability to lower the supply voltage for components that share the same power rail with the flash memory. This research also indicates that significant energy savings are possible if application-specific data can tolerate bounded errors. This work has been published in USENIX FAST [110] and ACM TECS [111].

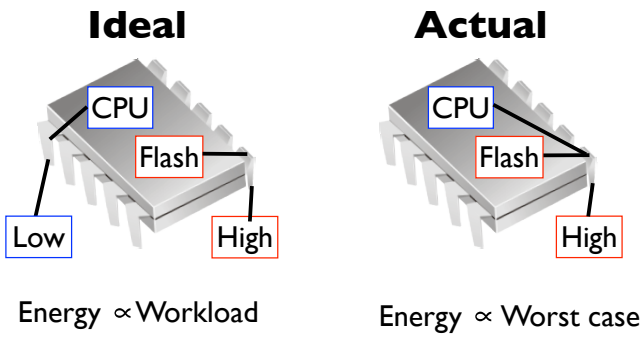


Figure 1.2. On a chip, different components have different minimum voltage requirements. Ideally, each component would have separate voltage sources and therefore the energy consumption of the device will be proportional to the workload. In practice, however, a chip has a single power source due to cost constraints. As a result, the energy consumption will be proportional to the worst case voltage requirement regardless of the workload.

1.5 TARDIS, Secure Notion of Time for Batteryless Devices

Unawareness of time has left smart cards and RFID tags vulnerable to a number of successful attacks [91, 90, 20, 55]. A batteryless tag does not know about 1) the amount of time spent by the reader to answer the challenge and 2) the time between two different queries (Figure 1.3). 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. The TARDIS keeps track of time by exploiting SRAM decay properties. 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. Our experiments demonstrate that the TARDIS can measure time ranging from seconds to several hours depending on hardware parameters Key challenges to implementing a practical TARDIS include compensating for temperature and handling variation across hardware. This work has been published in USENIX Security [99].

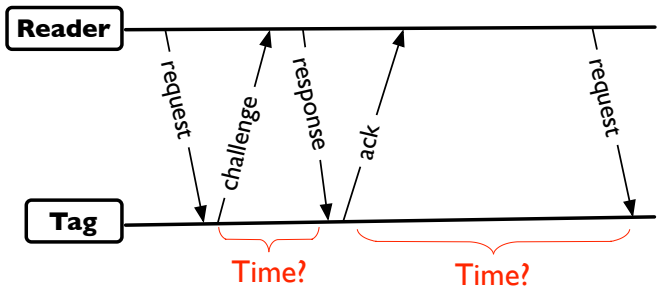


Figure 1.3. The tag cannot determine the time between a challenge and a response or the time between two sessions. The reader could respond to the tag as tardily as it likes or query the tag as quickly as it wants.

1.6 Nonsensors, Sensing at Low Voltage

Similar to the non-volatile storage, Analog to Digital Converters (ADCs) require relatively high voltage to guarantee functionality. For example, Texas Instruments MSP430F1232 requires the user to provide the ADC with at least 2.2 V to guarantee functionality while the CPU portion of the chip demands only 1.8 V. Table 1.2 shows more examples of popular microcontrollers with ADCs that restrict choices for the CPU voltage supply. This mismatch of voltage requirements causes energy disproportionality when the ADC and other on-board components share the same power rail. Nonsensors proposes to operate ADCs at the same voltage level as other embedded components to reduce the energy consumption of the whole system. To evaluate the nonsensors, we have designed a logging system that monitors and logs the output of ADC at low as well as high voltages. We define error as the measurement different of ADC at low and high voltage. Our preliminary results show that the ADC could work at voltages lower than specified by the chip manufacturers with introducing negligible errors (1.30%).

Microcontroller	CPU Minimum Voltage	ADC Minimum Voltage	Supply Voltage Gap
TI MSP430F1232 [128]	1.8 V	2.2 V	22%
PIC32M [82]	2.5 V	3.0 V	20%
ATmega128L [9]	2.7 V	4.5 V	66%
STM32F103C6 [120]	2.0 V	2.4 V	20%

Table 1.2. The ADC restricts choices for the CPU voltage supply on microcontrollers because the CPU shares the same power rail as the on-chip ADC.

1.7 Summary of Contributions

To unleash energy otherwise squandered on storage, reliability, time keeping, or sensing, this thesis makes the following contributions:

1. CCCP: A secure remote storage protocol that suits the characteristics and constraints of batteryless RFID tags, which can be used in the contexts of execution checkpointing and external data storage on an untrusted RFID reader infrastructure.

2. Half-Wits: A set of algorithms that enable reliable writes to flash memory while coping with low voltage, and an empirical evaluation that characterizes the behavior of on-chip flash memory at voltages below minimum levels specified by manufacturers.

3. TARDIS: Algorithmic building blocks to demonstrate the feasibility of using SRAM for a trustworthy source of time without power.

4. Nonsensors: Evaluation of operating analog to digital converters at low voltages without any hardware modifications to the existing circuitry.

This page is intentionally left blank.

CHAPTER 2

LOW-POWER EMBEDDED SYSTEMS

The focus of this thesis is on low-power devices with scarce energy resources; whether it is battery or it is harvested energy stored in capacitors. This chapter will give a brief background on battery-powered embedded systems and intermittently powered devices—batteryless devices that lose power regularly (e.g., passive RFID tags). Table 2.1 shows the abbreviations used in this thesis.

2.1 Battery-powered Embedded Systems

Billions of embedded systems ranging from thermostats and utility meters to more recently-developed sensor motes run on batteries and having a small energy budget makes energy efficiency a top priority for them. There are many new energy-saving techniques developed for low-power embedded systems recently. For example, Blaauw et al. [15] reduce power consumption by lowering the operating voltage of a pipelined CPU. Certain pipeline stages may produce incorrect computation that require re-computation, but the errors can be made rare to allow better scalability of power consumption. In another case, Sorber et al. [118] introduced Eon, an energy-aware language and runtime system that measured energy harvesting and consumption. Eon automatically adjusts the device behavior as the energy conditions change.

However, not all embedded systems are taking full advantage of their energy resources. Most systems inherit algorithms from their powerful ancestors without much tailoring or tuning based on their limited resources. For example, most cryptography algorithms are designed and developed to perfection based on the assumption that

Abbreviation	Full Name
ADC	Analog to Digital Conversion
CCCP	Cryptographic Computational Continuation Passing
CPU	Central Processing Unit
CRFID	Computational RFID
DRV	Data Retention Voltage
ECC	Error Correction Code
ECG	Electrocardiography
EPC	Electronic Product Code
GPIO	General Purpose Input/Output
RFID	Radio Frequency IDentification
SRAM	Static Random Access Memory
TARDIS	Time And Remanence Decay In SRAM
TI	Texas Instruments

Table 2.1. Terminology

resources (memory, power, or time) available to the device are plenty. The choice of using RC5 [105] as is in a sensor mote [67] could be replaced with encryption algorithms that are designed with limited resources in mind [18].

Matching energy resources and energy demands remains a challenge for many battery-powered systems. Pacemakers and Implantable Cardiac Defibrillators often contain high-capacity lithium-based batteries that last five to seven years [133] which requires surgery for battery (device) replacement. Many home electronic devices such as smoke detectors and other sensing devices that use microcontrollers need battery maintenance very often. Wireless peripherals and game controllers have battery lifetime of months and hours respectively. The short lifetime of everyday electronic devices states the importance of improving energy efficiency of embedded system.

2.2 Intermittently Powered Systems

Unlike battery-powered devices that assume their energy resource will be available on orders of months, batteryless system that live on harvested energy expect energy interruptions every few seconds. We define intermittently powered devices as

systems that do not rely on constant source of power and faces reboots very regularly. Examples of devices targeted and experimented within this thesis are RFID tags and smart cards, both of which rely on Radio Frequency (RF) power.

2.2.1 Computational RFID Tags

Consistent with the usage of RFID terminology, the term *Computational RFID* (CRFID) has two meanings: the *model* under which passively powered computers operate in concert with an RFID reader infrastructure, and the passively powered computers themselves. CRFIDs represent a class of programmable, batteryless computers [22, 101]. The small size and low maintenance requirements of CRFIDs make them especially appealing for adding computational capabilities to contexts in which placing or maintaining a conventional computer would be infeasible or impossible. However, CRFID systems require that nearby, actively powered RFID readers provide energy whenever computation is to occur, a requirement that may not suit all applications.

The components of a CRFID are: a low-power microcontroller; onboard RAM; flash memory (on or off the microcontroller); energy harvesting circuitry tuned to a certain frequency (e.g., 913 MHz for EPC Gen 2 RFID); an antenna; a transistor between the antenna and the microcontroller to modulate the antenna's impedance; a capacitor for storage of harvested energy; one or more analog-to-digital converters; and optional sensors for physical phenomena such as acceleration, heat, or light. The first working example of a CRFID is the Wireless Identification and Sensing Platform, or WISP [112], a prototype device slightly smaller than a postage stamp (discounting its inches-long antenna). The WISP is built around an off-the-shelf TI MSP430 microcontroller. Another example of a CRFID is UMass Moo [140], with a more powerful microcontroller, with additional memory, more pins for controlling sensors and actuators, and redesigned voltage regulation circuitry.

2.2.2 Smart Cards

Smart cards which are also known as Integrated Circuit Cards (ICCs) refer to card-sized devices that contain an integrated circuit. Based on the way smart cards are powered up, they are considered contact or contactless. Contact smart cards which are more common than contactless ones, use six pins to communicate the a card reader. One of the pins is the VCC which should be in contact with the power pin of the reader. The contactless cards, harvest RF energy using their antenna and do not need to directly be in contact with the reader.

Although some smart cards are battery-powered, here we are talking about the ones that rely only on energy received from the reader. For example, contactless credit cards or transit cards are batteryless. The read time of card which is the equal to the power-up time is required to be less than 300ms–500ms (acceptable delay by customers to use a card). The short periods of power-up time makes smart cards count as intermittently-powered devices.

In addition to energy issues, the security and privacy of smart cards are usually challenging. Some RFID credit cards are vulnerable to replay attacks because they lack a notion of time [55]. Oswald et al. [91] recently demonstrated how to extract the 112-bit key from a MIFARE DESFire contactless smartcard (used by the Clipper all-in-one transit payment card). The side channel attack required approximately 10 queries/s for 7 hours. An earlier MIFARE classic chip was broken in 2008 by Nohl et a. [87]. Protecting the security and privacy of batteryless smart cards remains a challenging problem as long as there is no secure notion of time provided to smart cards.

CHAPTER 3

CCCP: SECURE REMOTE STORAGE FOR COMPUTATIONAL RFID TAGS

Passive RFID tags harvest their operating energy from an interrogating reader, but constant energy shortfalls severely limit their computational and storage capabilities. In this chapter, I propose *Cryptographic Computational Continuation Passing* (CCCP), a mechanism that amplifies programmable passive RFID tags' capabilities by exploiting an often overlooked, plentiful resource: low-power radio communication. While radio communication is more energy intensive than flash memory writes in many embedded devices, we show that the reverse is true for passive RFID tags.

The main idea of this work is that a CRFID can securely use radio communication as a less energy-intensive alternative to local, flash-based storage. The smaller energy requirements of radio allow the CRFID either to devote more energy to computation or to accomplish the same tasks using less energy, which may translate into a longer operating range.

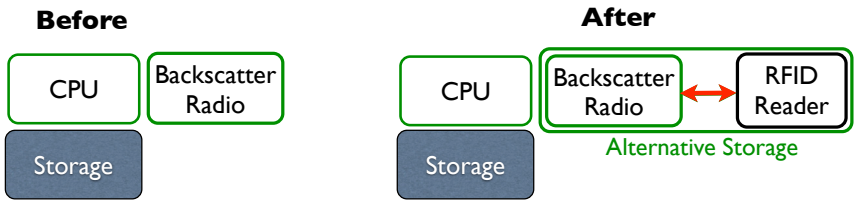


Figure 3.1. CCCP exploits radio to offer a less energy-expensive alternative to local non-volatile storage. The main challenge of this work is to preserve the security and privacy of the tag-to-reader communication.

Security is the major challenge in such remote storage system. Using scant and fleeting energy, a tag must enforce confidentiality, authenticity, integrity, and data

freshness while communicating with potentially untrustworthy infrastructure. This work contribution synthesizes well-known cryptographic and low-power techniques with a novel flash memory storage strategy, resulting in a secure remote storage facility for an emerging class of devices.

Evaluation of CCCP consists of energy measurements of a prototype implementation on the batteryless, MSP430-based Intel WISP platform. Experiments show that—despite cryptographic overhead—remote checkpointing consumes less energy than checkpointing to flash for data sizes above roughly 64 bytes. CCCP enables secure and flexible remote storage that would otherwise outstrip batteryless RFID tags' resources.

3.1 Frequent Power Loss on Tags, but Plentiful External Resources

Several key observations motivate the development of secure remote storage for computational RFIDs.

Frequent loss of power may interrupt computation. The CRFID model posits computing devices that are primarily powered by RF energy harvesting, a mechanism that is naturally finicky because of its dependence on physical conditions. Any change to a CRFID's physical situation—such as its position or the introduction of an occluding body—may affect its ability to harvest energy. Existing systems that use RF harvesting typically counteract the effect of physical conditions by placing stringent requirements on use. For example, an RFID transit card reader presented with a card may behave in an undefined way unless the card is within 1 cm for at least 300 ms, parameters designed to ensure that the card's computation finishes while it is still near the reader. CRFID applications may preclude such a strategy: programs on general-purpose CRFIDs may not offer convenient execution time horizons, and communication distances may not be easily controlled. Without any guarantees of

energy availability, it may be unreasonable to mandate that programs running on CRFIDs complete within a single energy lifecycle. As an extension of the Mementos system [101], CCCP aims to address the problem of suspending and resuming computations to facilitate spreading work across multiple energy lifecycles.

Storing remotely may require less energy than storing locally. Some amount of onboard nonvolatile memory exists on a CRFID, so an obvious approach to suspension and resumption is simply to use this local memory for state storage. However, to implement nonvolatile storage, current microcontrollers use flash memory, which imports several undesirable properties. While reading from flash consumes energy comparable to reading from volatile RAM, the other two flash operations—writing and erasing—require orders of magnitude more energy per datum (Table 3.3). Our measurements of a CRFID prototype reveal that the energy consumption of storing a datum locally in flash can in fact exceed the energy consumption of transmitting the same datum via backscatter communication.

To illustrate the difference between flash and radio storage on a CRFID and to show how the relationship is different on a sensor mote, we offer Figure 1.1. The figure helps explain why designers of mote-based systems choose to minimize radio communication; similarly, it justifies our exploration of radio-based storage as an alternative to flash-based storage on CRFIDs.

It should be noted that CCCP, although its primary data storage mechanism is the communication link between CRFIDs and readers, still requires *some* flash writes during storage operations: CCCP maintains a counter in flash to ensure that key material is not reused. However, because CCCP employs hole punching (Section 3.2.4.4) to maintain the counter, the amount of data written for counter updates is small compared to the amount of data that can be stored at once—small enough not to obviate the energy advantage of radio-based storage—and counter updates do not frequently necessitate erasures.

EPC Gen 2 RFID readers are typically not standalone devices. Rather, they are connected to networks or other systems (for, e.g., control or logging) that can offer computing resources such as storage. The benefit to CRFIDs that communicate with such a reader infrastructure is access to effectively limitless storage. Several kilobytes of onboard flash memory is minuscule compared to the potentially vast amount of storage available to networked RFID readers. While unlimited external storage is not obviously helpful for saving computational state—a CRFID cannot save or restore more state than it can hold locally—its usefulness as general-purpose long-term storage is analogous to the usefulness of networked storage for PCs.

RFID protocols allow arbitrary payloads. While the EPC Gen 2 protocol imposes constraints on the transmissions between RFID tags and RFID readers—for example, the maximum upstream data rate from tag to reader is 640 Kbps [37]—it also offers sufficient flexibility that CCCP can be implemented on top. In particular, the Gen 2 protocol permits a reader to issue a *Read* command to which a tag can respond with an arbitrary amount of data. Previous versions of the EPC RFID standard mandated a small response size that would have imposed severe communication overhead on large upstream transmissions.

CRFID is not married to EPC Gen 2 as an underlying protocol, but the existence of a widespread RFID reader infrastructure and the availability of commodity reader hardware makes for easy prototyping.

3.1.1 Challenges Due to Energy Scarcity

Several energy-related considerations limit the resources available for computation on CRFIDs, limiting the utility of CRFIDs as a general-purpose computing platform. Ransford et al. [101] discuss the difficulty of effectively utilizing a storage capacitor and enumerate the drawbacks of using capacitors for energy storage; Buettner et al. [22] discuss how energy limitations bear on the deployment of a CRFID-based

system. Two key design features of CRFIDs pose energy challenges to a system like CCCP: first, the voltage and current requirements of flash memory constrain the design of flash-bearing CRFIDs and limit the portion of a CRFID's energy lifecycle that is usable for computation. Second, a CRFID's reliance on energy harvesting and backscatter communication means that a CRFID cannot compute or communicate without reader contact.

Flash memory limitations. Microcontrollers that incorporate flash typically have separate threshold voltages: one threshold for computation, and a higher threshold for flash writes and erases. Because of this difference, flash writes cannot be executed at arbitrary times during computation on a CRFID; they require sufficient voltage on the storage capacitor. Without a constant supply of energy, capacitor voltage declines with time and computation, so waiting until the end of a computation to record its output to nonvolatile memory may be risky.

The size of a CRFID's storage capacitor imposes another basic limitation. Flash writes, which owe their durability to a process that effects significant physical changes, require more current and time (and therefore energy) than much simpler RAM or register writes. Per-datum measurements show that, on a WISP's microcontroller, writing to flash consumes roughly 400 times as much energy as writing to a register [101]. Such outflow from the storage capacitor can dramatically shorten the device's energy lifecycles.

Non-autonomous operation. Backscatter communication involves modulating an antenna's impedance to reflect radio waves—an operation that, for the sender, involves merely toggling a transistor to transmit binary data. Such communication cannot occur without a signal to reflect; CRFIDs, like other passive RFIDs, are therefore constrained to communicate only when a reader within range is transmitting. Computation may occur during times of radio silence, but only if sufficient energy remains in the CRFID's storage capacitor. Unlike battery-powered platforms that

can operate autonomously between beacon messages from other entities, a CRFID may completely lose power between interactions with an RFID reader. Our experience shows that, lacking a source of harvestable energy, the storage capacitor on a WISP (Revision 4.0) can support roughly one second of steady computation before its voltage falls below the microcontroller's operating threshold. Such limitations constrain the design space of applications that can run on CRFIDs. For example, without autonomy, an application cannot plan to perform an action at a specific time in the future.

Unsteady energy supply. A key challenge CRFIDs face is that their supply of energy can be unsteady and unpredictable, especially under changing physical conditions. RFID readers may not broadcast continuously or even at regular intervals, and they do not promise any particular energy delivery schedule to tags. In our experiments, even within inches of an RFID reader that emitted RF energy at a steady known rate, the voltage on a CRFID's storage capacitor did not appear qualitatively easy to predict despite the fixed conditions. A CRFID's storage capacitor must buffer a potentially unsteady supply of RF energy without the ability to predict future energy availability.

3.2 Design of CCCP

CCCP's primary design goal is to furnish computational RFIDs with a mechanism for secure outsourced storage that facilitates the suspension and resumption of programs. This section describes how CCCP is designed to meet that goal and several others. Refer to Section 3.3 for a discussion of CCCP's implementation, and refer to Section 3.4 for an evaluation of CCCP's design choices and security; in particular, Section 3.4.3.1 discusses the overhead imposed by cryptographic operations.

Given a chunk of serialized computational state on a CRFID, CCCP sends the state to the reader infrastructure for storage. (CCCP is designed to work indepen-

Design goal	Approach
Computational progress	Checkpointing via radio to untrusted RFID readers
Security: authentication, integrity	UHF-based MAC
Security: data freshness	Key non-reuse; counter stored by hole punching in nonvolatile memory
Security: confidentiality	Symmetric encryption with keystream precomputation

Table 3.1. CCCP's design goals and techniques for accomplishing each of them.

dently of the state serialization method, and does not prescribe a specific method.) In a subsequent energy lifecycle, an RFID reader that establishes communication with the tag sends back the state, CCCP performs appropriate checks, and the CRFID resumes computation where it left off. CCCP provides several operating modes that allow an application designer to increase security—by adding authentication alone, or authentication and encryption—at the cost of additional per-checkpoint energy consumption. Table 3.1 describes how CCCP meets each of the goals discussed in this section.

3.2.1 Design Goal: Computational Progress on CRFIDs

CCCP remotely checkpoints computational state to make long-running operations robust against power loss—i.e., to enable their *computational progress*. We define computational progress as change of computational state toward a goal (e.g., the completion of a loop). While CRFIDs are able to finish short computations in a small number of energy lifecycles (e.g., symmetric-key challenge-response protocols [26, 54]), the challenges described in Section 2.2.1 make it difficult for a CRFID to guarantee the computational progress of longer-running computations.

If a CRFID loses power before it completes a computation, all volatile state involved in the computation is lost and must be recomputed in the next cycle. If energy availability is similarly inadequate in subsequent cycles, the CRFID may never ob-

tain enough energy to finish its computation or even to checkpoint its state to flash memory. We refer to such vexatious computations as *Sisyphean tasks*. (Sisyphus was condemned to roll a large stone up a hill, but was doomed to drop the stone and repeat hopelessly forever [58].) A major goal of CCCP is to prevent tasks from becoming Sisyphean by shifting energy use away from flash operations and toward less energy-intensive radio communication.

3.2.2 Checkpointing Strategies: Local vs. Remote

We consider two strategies for the nonvolatile storage of serialized checkpointed state. The first, writing the state to flash memory, involves finding an appropriately sized region of erased flash memory or creating one via erase operations. The second strategy, using CCCP, requires a CRFID to perform zero or more cryptographic operations (depending on the operating mode) and then transmit the result via backscatter communication.

The obvious advantage of flash memory is that its proximity to the CRFID makes it readily accessible. On-chip flash has the further advantage that it may be inaccessible to an attacker. However, the operating requirements of flash are onerous in many situations. With unlimited energy, a CRFID could use flash freely and avoid the complexity of a radio protocol such as CCCP. Unfortunately, energy is limited in ways described elsewhere in this chapter, and several disadvantages of flash memory diminish its appeal as a store for checkpointed state. The most obvious disadvantage is an imbalance between the requirements for reading and writing. Write and erase operations require more time and energy per bit than reading (Table 3.3); additionally, the minimum voltage and current requirements are higher. For example, in the case of the MSP430F2274, read operations are supported at the microcontroller's minimum operating voltage of 1.8 V, but write and erase operations require 2.2 V. Finally, flash memory (both NOR and NAND types) generally imposes the

requirement that memory segments be erased before they are written: if a bit acquires a zero value, the entire segment that contains it must be erased for that bit to return to its default value of 1. Aside from burdening the application programmer with inconvenience, erase-before-write semantics complicate considerations of energy requirements. These disadvantages are minor afflictions for higher-powered systems, but they pose serious challengess to the utility of flash memory on CRFIDs.

Backscatter transmission, since it involves modulating only a single transistor to encode data, requires significantly less energy than transmission via active radio. In fact, our measurements (Figure 3.4) show that backscatter transmission of an authenticated, encrypted state checkpoint (plus a small amount of bookkeeping in flash) can require less energy than exclusively writing to flash memory, even after including the energy cost of encrypting and hashing the checkpointed state. Because of its consistent behavior throughout the microcontroller's operating voltage range, backscatter transmission is an especially attractive option when the CRFID receives radio contact frequently but cannot harvest energy efficiently, in which case writing to flash may be infeasible because of insufficient energy in the storage capacitor. These circumstances may occur far from the reader, or in the presence of radio occlusions, or when a computation uses energy quickly as soon as the CRFID wakes up.

Despite its advantages over flash storage and active radio, CCCP's reliance on backscatter transmission has drawbacks. Bitrate limitations in the EPC Gen 2 protocol cause CCCP's transmissions to require up to twice as much time per datum as flash storage on some workloads. The best choice of storage strategy depends on an application's ability to tolerate delay and the necessity of saving energy.

3.2.3 Threat Model

We define CCCP's threat model as a superset of the attacks that typically threaten RFID systems [62]. The most obvious way an attacker can disrupt the operation of a

CRFID is to starve it of energy by jamming, interrupting, or simply never providing RF energy for the CRFID to harvest. Because they depend entirely on harvestable energy, CRFIDs cannot defend against such denial-of-service (DoS) attacks, so we consider these attacks as a problem to be dealt with at a higher system level. We instead focus on two types of attacks that a CRFID can use its resources to address: (1) active and passive radio attacks and (2) attacks by an untrusted storage facility.

An adversary may attempt to:

- Eavesdrop on radio communication in both directions between a CRFID and reader.
- Masquerade as a legitimate RFID reader in order to collect checkpointed state from CRFIDs. Because CRFIDs do not trust reader infrastructure, such an attack should allow an attacker to collect only ciphertext.
- Masquerade as a legitimate RFID reader in order to send corrupted data or old data (e.g., a previous computational state) to the CRFID. Such invalid data should not trick the CRFID into executing arbitrary or inappropriate code.
- Masquerade as a specific legitimate CRFID in order to retrieve that CRFID's stored state from the reader. This state should be useless without access to the keystream material that encrypted it—keystream material that is stored in the legitimate owner's nonvolatile memory and never transmitted.

We additionally assume that an adversary cannot physically inspect the contents of a CRFID's memory.

3.2.4 Secure Storage in CCCP

Because computational RFIDs depend on RFID readers for energy—if a CRFID is awake, there is probably a reader nearby—readers are a natural choice for storing information. But a reader trusted to provide energy should not necessarily be trusted with sensitive information such as checkpointed state.

CCCP involves communication with untrusted reader infrastructure, so we establish several security goals:

- Authenticity: a CRFID that stores information on external infrastructure will eventually attempt to retrieve that information, and the authenticity of that information must be cryptographically guaranteed. Under CCCP, the only party that ever needs to verify the authenticity of a CRFID's stored information is the CRFID itself.
- Integrity: an untrusted reader may attempt to impede a CRFID's computational progress by providing data from which the CRFID cannot resume computation (e.g., random junk). While CCCP cannot prevent a denial of service attack in which a reader provides only junk, it guarantees that CRFIDs will compute only on data they recognize.
- Data freshness: just as a reader can provide corrupted data instead of usable data, it can replay old state in an attempt to hinder the computational progress of a computation. Under CCCP, a CRFID recognizes and rejects old state.
- Confidentiality: in certain applications, the leaking of intermediate computational state might be a critical security flaw. For other applications, confidentiality may not be necessary.

3.2.4.1 Keystream Precomputation

Because CCCP's threat model assumes a powerful adversary that can intercept all transmissions, CCCP never reuses keystream material when encrypting data or computing MACs. We use CCCP's refreshable pool of pseudorandom bits (a circular buffer in the CRFID's nonvolatile memory) as a cryptographic keystream to provide confidentiality and authentication.

CCCP stores keystream material on the CRFID because we assume that the CRFID trusts only itself; a CRFID cannot extract trustworthy keystream material from a reader it does not trust, nor from any observable external phenomenon (which, in our

Variable	Description
chkpt_counter	Counter representing the number of checkpoints completed; used to calculate the location of the first unused keystream material; updated each time keystream material is consumed; unary representation
kstr_end	Pointer to the end of the last chunk of unused keystream bits in keystream memory; updated during key refreshment
rc5counter	Incrementing counter used as an input to RC5 while filling keystream memory with pseudorandom data; updated during key refreshment

Table 3.2. Variables CCCP stores in nonvolatile memory.

threat model, an attacker would be able to observe equally well). Because a CRFID can reserve only finite storage for storage of keystream material, the material must be periodically refreshed. CCCP opportunistically refreshes the keystream material with pseudorandom bits, following Algorithm 3.

To provide unique keystream bits to cryptographic operations (encryption and MAC), CCCP uses an existing implementation [26] of the RC5 block cipher [105] in counter mode to generate pseudorandom bits and store them to flash. The choice of a block cipher in counter mode means that the resulting MAC and ciphertext are secure against a computationally bounded adversary [21]. A stream cipher would work equally well in principle, but in implementing CCCP, we found that those under consideration required a large amount of internal read-write state. For example, the stream cipher ARC4 requires at least 256 bytes of RAM [114], whereas RC5 requires only an 8-byte counter. The RC5 key schedule is preloaded into flash memory the first time the device is programmed, and the keystream materials are generated during periods of excess energy (or *power seasons*; see § 3.2.5). One such period of excess energy is the CRFID's initial programming, at which time the entire keystream buffer is filled with keystream bits. To avoid reusing keystream bits, CCCP maintains several variables in nonvolatile memory. Table 3.2 summarizes the variables CCCP stores in nonvolatile memory.

3.2.4.2 UHF-based MAC for Authentication and Integrity

CCCP uses a MAC scheme based on universal hash functions (UHF) [25] to provide authentication and integrity. CCCP constructs the MAC by first hashing the message and then XORing the 80-bit hash with a precomputed cryptographic keystream. Because of the resource constraints of CRFIDs, it is critical to use a scheme that consumes minimal energy, and according to recent literature [16, 38], UHF-based MACs are potentially an order of magnitude faster than MACs based on cryptographic hash functions. We chose UMAC [16] as the MAC function after evaluating several alternatives. Our experiments on WISP (Revision 4.0) CRFIDs determined that UMAC takes on average 18.38 ms and requires 28.79 μJ of energy given a 64-byte input.

3.2.4.3 Stream Cipher for Confidentiality

To provide confidentiality, a CRFID simply XORs its computational state with a precomputed cryptographic keystream. This encryption scheme is low-cost in terms of computation and energy, but it relies on using each keystream bit at most once. CCCP ensures that the encryption and MAC functions never reuse keystream bits by keeping track of the beginning and end of fresh keystream material in flash memory. The keystream pool is represented as a circular buffer. The address of the first unused keystream material is derived from the value of *chkpt_counter* (Table 3.2), and the last unused keystream material ends just before the address pointed to by *kstr_end*.

If the application using CCCP demands confidentiality at all times, then if CCCP cannot satisfy a request for unused keystream bits, it pauses its work to generate more keystream bits. This behavior is inspired by that of the blocking `random` device in Linux [51].

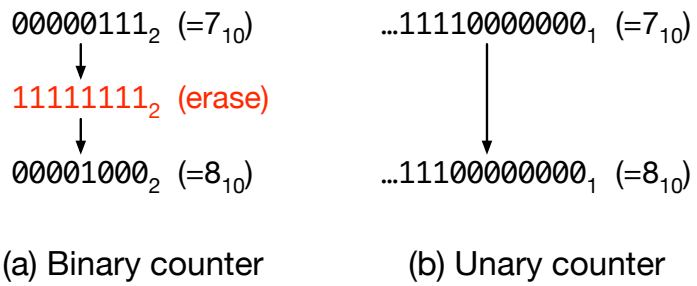


Figure 3.2. Illustration of hole punching. While incrementing a binary counter (a) in flash memory may require an energy-intensive erase operation, complemented unary representation ((b), with the number of zeros, or "holes," representing the counter value) allows for incrementing without erasure at a cost of space efficiency.

3.2.4.4 Hole Punching for Counters Stored in Flash

To avoid reusing keystream material, CCCP maintains a counter (*chkpt_counter*) from which the address of the first unused keystream bits can be derived. The counter is stored in flash memory because it is used for state restoration after power loss. However, incrementing a counter stored in binary representation always requires changing a 0 bit into a 1 bit (Figure 3.2(a)). On segmented flash memories, changing a single bit to 1 requires the erasure (setting to 1) of the entire segment that contains it—at least 128 bytes on the MSP430F2274—before the new value can be written. An additional cost that varies among flash cells is that they wear out with repeated erasure and writing [52].

To avoid energy-intensive erasures and minimize the energy cost of writing counter updates, CCCP represents *chkpt_counter* in complemented unary instead of binary. CCCP interprets the value of such a counter as the number of 0 bits therein. Because 1 bits can be changed to 0 bits without erasure, incrementing a counter requires a relatively small write, with erasures necessary only if the unary counter must be extended into unerased memory. We call this technique *hole punching* after the visual effect of turning 1 bits into 0 bits. Since *chkpt_counter* is simply incremented at each remote checkpoint, updating the counter generally requires writing only a single word.

Operation	Seg. erase	Write	Read	Write
Size (bytes)	128	128	128	2
Energy (μJ)	46.81	56.97	0.64	0.96

Table 3.3. Comparison of energy required for flash operations on an MSP430F2274. Hole punching often allows CCCP to use a single-word write (2 bytes on the MSP430) instead of a segment erase when incrementing a complemented unary counter.

Table 3.3 illustrates the energy cost of erasing an entire segment and the energy cost of writing a single word.

To minimize the length of the unary *chkpt_counter*'s representation and to facilitate simple computation of offsets, CCCP assumes a fixed size for checkpointed state; in practice an application designer can choose an appropriate value for the fixed checkpoint size.

3.2.4.5 Extension for Long-Term Storage

Under CCCP, readers can act not only as outsourced storage for computational state, but also as long-term external storage. Because of their ultra-low-power microcontrollers, CRFIDs are likely to have only a small amount of flash available for data storage. Moreover, since flash operations are energy intensive, depending exclusively on flash memory as a storage medium is undesirable. CCCP could enable a CRFID to instead use the reader infrastructure as an external storage facility with effectively limitless space.

Long-term storage requires a different key management strategy than checkpointing data. With a temporary checkpointing system, the CRFID needs access only to the keystream material used to prepare the last checkpoint sent to a reader. However, in the case of long-term storage, the CRFID may require access to all of the data it has ever stored on the reader and therefore must remember all of the cryptographic keys from those stores. To avoid this unrealistic requirement, a potential extension

to CCCP allows the CRFID to generate keys on demand when long-term storage is required.

There are two operations that CCCP can provide to a CRFID application for this purpose:

- To satisfy a STORE(*data*) request, CCCP provides a keystream generator in the form of a block cipher in counter mode; this requires a monotonically increasing counter in addition to CCCP's *chkpt_counter*. CCCP XORs the given data with the generated keystream and then constructs a MAC, then sends the ciphertext and MAC to the reader for storage. CCCP then sends the counter value back to the application.
- To satisfy a RETRIEVE(*index*) request, CCCP asks the RFID reader for the data at the given index. CCCP then generates the same keystream it used to encrypt the data by passing the index to the block cipher. Finally, CCCP verifies the MAC provided by the reader and returns the decrypted data to the application.

3.2.5 Power Seasons

If a CRFID could predict future energy availability, then it would be able to schedule its generation of keystream bits and ensure that it never exhausted its supply of pseudorandomness during normal operation. However, because CRFIDs lack autonomy and cannot depend on RFID reader infrastructure to provide a steady energy supply, we roughly classify the energy availability scenarios a CRFID faces into two *seasons*. We assume that the general case is a *winter* season, in which a CRFID cannot consistently harvest enough energy to perform all of its tasks. During winter, the CRFID must focus on minimizing checkpoints and wasted energy. The other season is *summer*, during which harvested energy is plentiful and the CRFID can

afford to perform energy-intensive operations such as precomputation and storage of keystream material for later use.

CCCP can identify a summer season if one of two conditions is true. First, the CRFID may find itself awake with no computations left to complete, for example after it has finished a sensor reading. Second, the CRFID may find itself communicating with a reader that does not understand CCCP and simply provides harvestable energy.

3.3 Implementation

The components of CCCP span two environments: CRFIDs and RFID readers. On a CRFID, CCCP accepts data from an application and uses the CRFID's backscatter mechanism to ship the data to a reader. The reader (which we consider as an RFID reader plus a controlling computer) is programmed to participate in the CCCP protocol and return computational state where necessary. This section describes the CRFID-side components, the reader-side components, and the protocol that ties them together.

The CRFID side of CCCP is implemented in the C programming language on WISP (Revision 4.0) prototypes. At its core are three primary routines, which we present in pseudocode: Checkpoint (Algorithm 1), Resume (Algorithm 2), and Key-Refresh (Algorithm 3). Checkpoint and Restore refer to a counter called *chkpt_counter* from which CCCP derives the address of the first unused keystream material. For routines that require radio communication, we borrow radio code from Intel's WISP firmware version 1.4. Note that, since a CRFID cannot assume that a reader is listening at an arbitrary time, the Transmit subroutine waits for an interrupt indicating that the CRFID has received a go-ahead message from the reader.

The RFID reader side of CCCP consists only of code to drive the reader appropriately for communication events. Because of the Gen 2 protocol's complexity, we have not completely implemented the reader side of the CCCP protocol. Rather

Algorithm 1 The CHECKPOINT routine encrypts, MACs, and transmits a fixed-size (*STATE_SIZE*, selected by the application designer) chunk of computational state. $\langle A, B \rangle$ means the concatenation of A and B with a delimiter in between. 80 bits is the fixed output size of NH, the hash function used by UMAC. For arithmetic simplicity, this pseudocode treats the *keystream* pool as an infinite array.

CHECKPOINT(*state*, *keystream*, *chkpt_counter*)

1 ≻ Compute the (constant) amount of keystream material that will be used in this invocation
2 $chkpt\_size = STATE\_SIZE + \text{LENGTH}(\langle state, chkpt\_counter \rangle) + 80$ bits
3
4 $k \leftarrow chkpt\_counter \times chkpt\_size$ ≻ $keystream[k]$ holds unused keystream material
5 $chkpt\_counter \leftarrow chkpt\_counter + 1$ ≻ Update *chkpt_counter* in nonvolatile memory
6
7 $C \leftarrow state \oplus keystream[k \ldots k + STATE\_SIZE - 1]$
8 ≻ Encrypt *state* by XORing with keystream material
9 $k \leftarrow k + STATE\_SIZE$ ≻ . . . and advance k
10
11 $H \leftarrow \text{NH}(\langle C, k \rangle, keystream[k \ldots k + \text{LENGTH}(\langle C, k \rangle) - 1])$ ≻ Hash the encrypted state
12 $k \leftarrow k + \text{LENGTH}(\langle C, k \rangle)$ ≻ . . . and advance k
13
14 $M \leftarrow H \oplus keystream[k \ldots k + \text{LENGTH}(H) - 1]$ ≻ Construct an 80-bit MAC
15
16 TRANSMIT(C, M) ≻ Note: TRANSMIT blocks until a reader is detected

than write a large amount of code for the reader, we chose to use simple control programs for the reader and inspect the exchanged messages manually, a strategy that allowed us to concentrate on the more resource-constrained CRFID side of the system while avoiding porting applications from one proprietary reader to another. (The WISP [Revision 4.0] is nominally compatible with only the Alien ALR-9800 and Impinj Speedway readers; we chose to use a desktop PC to program these readers for the sake of simplicity and portability.) A full implementation of the reader side would properly parse each message received from the CRFID and manage storage for checkpointed state.

Algorithm 2 The RESUME routine receives an encrypted checkpoint C and a message authentication code M from a reader, then restores the computational state of the CRFID if the received data pass an authenticity test. *chkpt_counter* is the value stored in nonvolatile memory at the beginning of CHECKPOINT (Algorithm 1). We assume that, since k and *chkpt_counter* are both numbers, their in-memory representations have the same length. As in Algorithm 1, this pseudocode treats the *keystream* pool as an infinite array for arithmetic simplicity. $\langle A, B \rangle$ means the concatenation of A and B with a delimiter in between. 80 bits is the fixed output size of NH, the hash function used by UMAC.

RESUME(C, M, *keystream*, *chkpt_counter*)

1 $\succ$ Find the first unused keystream material, then backtrack to find the keystream material CHECKPOINT used to hash and MAC the ciphertext
2 *chkpt_size* = *STATE_SIZE* +LENGTH($\langle C$, *chkpt_counter*$\rangle$) + 80 bits
3 $k \leftarrow$ *chkpt_counter* $\times$ *chkpt_size*
4 $k \leftarrow k -$ (LENGTH($\langle C, k \rangle$) + 80 bits)
5
6 $H \leftarrow$ NH($\langle C, k \rangle$, *keystream*$[k \ldots k +$ LENGTH($\langle C, k \rangle) - 1]$) $\succ$ Compute the hash
7 $k \leftarrow k +$ LENGTH($\langle C, k \rangle$) $\succ$ ... and advance k to point to the MAC
8
9 **if** $M = H \oplus$ *keystream*$[k \ldots k +$ LENGTH(H) $- 1]$ $\succ$ If the MAC is OK, then...
10 **then** $k \leftarrow k -$ (LENGTH(C) + LENGTH($\langle C, k \rangle$)) $\succ$ backtrack further ...
11 *state* $= C \oplus$ *keystream*$[k \ldots k +$ LENGTH(C) $- 1]$ $\succ$ and decrypt C to get *state*
12 RESTORE-STATE(*state*)
13 **else** $\succ$ Do nothing

3.3.1 Communication Protocol

The CRFID model places a number of restrictions on communication. The only communication hardware on a CRFID is a backscatter circuit involving an antenna and a modulating transistor; an active radio would require significantly more energy. Since backscatter simply reflects an incoming carrier signal, a prerequisite for communication is that the reader emits an appropriate carrier signal. In our experiments, we used two different EPC Gen 2-compatible RFID readers that are readily available as off-the-shelf products; we used no nonstandard reader hardware or antennas.

Algorithm 3 The KEY-REFRESH replaces used keystream material with new keystream material in nonvolatile memory. Unlike in CHECKPOINT and RESUME, this pseudocode treats the *keystream* pool as a fixed-size circular buffer. This allows us to treat keystream material between k and *kstr_end* as unused, and the rest—between *kstr_end* and k—as used. This pseudocode omits two subtleties for simplicity: first, the routine must not erase keystream material that is waiting to be used by RESUME. Second, because flash erasure affects entire segments at once, the ERASE-MEMORY-RANGE routine must sometimes restore data that should not have been erased.

KEY-REFRESH(*keystream*, *kstr_end*, *chkpt_counter*, *rc5counter*)

```
1  ≻ Find the first unused keystream in the circular keystream buffer
2  chkpt_size = STATE_SIZE +(STATE_SIZE +LENGTH(⟨null, chkpt_counter⟩)) + 80 bits
3  k ← chkpt_counter × chkpt_size (mod LENGTH(keystream)/ chkpt_size)
4
5  ≻ Erase all used keystream memory, then write pseudorandom data to it
6  ERASE-MEMORY-RANGE(keystream[kstr_end ... k])
7  i ← kstr_end
8  while (i < k)
9        do rc5counter ← rc5counter +1 ≻ Update counter in nonvolatile memory
10          keystream[i] ← RC5(rc5counter −1) ≻ Write keystream into nonvolatile memory
11          kstr_end ← i + 1          ≻ Update kstr_end in nonvolatile memory
12          i ← i + 1
```

CCCP's communication protocol is based on primitives provided by the EPC Gen 2 RFID protocol (the RFID protocol the WISP understands). Specifically, CCCP makes use of three EPC Gen 2 commands:

- A reader issues a *Query* command to a specific tag (in our case, a CRFID). The *Query* command comprises a 4-tuple: ⟨*action*, *membank*, *pointer*, *length*⟩. While a conventional RFID tag may require reasonable values for all four tuple members, a CRFID need examine only the fourth member to learn the maximum reply length the reader will accept. The reader can use the other three fields to encode meta-information such as whether the reader wants to offer checkpointed state to the CRFID.

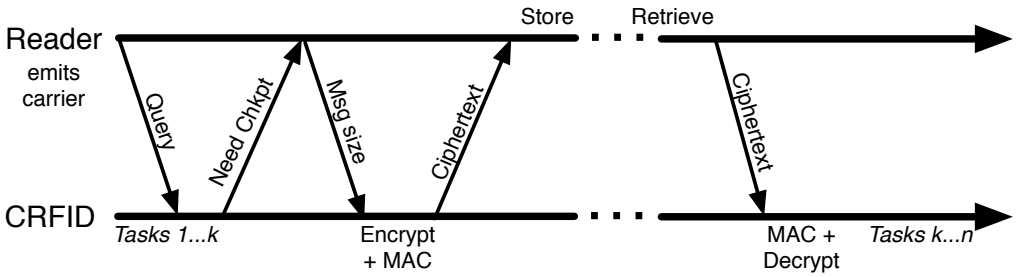


Figure 3.3. Application-level view of the CCCP protocol. The CRFID sends a request to checkpoint state while in the presence of a reader, and the reader specifies the maximum size of each message. The CRFID then prepares the checkpoint and transmits it in a series of appropriately sized messages. The reader stores the checkpoint data for later retrieval by the CRFID. All messages from the reader to the CRFID also supply power to the CRFID if the latter is within range.

- A reader issues a *Read* command to a specific tag to request an arbitrary amount of data from an RFID tag's memory. A CRFID can respond to a coordinated *Read* command with a chunk of checkpointed state.
- A reader issues a *Write* command to send data for storage in a specific tag's memory. Because RFID tags tend to have fewer resources even than CRFIDs, *Write* commands transmit only a small amount (16 bits) of data. A CRFID can request a series of *Write* commands from the reader to retrieve checkpointed state, then reassemble the results in memory and restore its state from the checkpoint.

Figure 3.3 gives an overview of CCCP's message types and their ordering. CCCP does not require protocol changes to the EPC Gen 2 standard, but it requires that an RFID reader be controlled by an application that understands CCCP. While a proprietary radio protocol for CCCP could be more efficient than one built atop an existing RFID protocol, a goal of CCCP—inherited from the design goals of the WISP CRFID—is to maintain compatibility with existing RFID readers.

3.4 System Evaluation

This section justifies our design choices and offers evidence for our previous claims. We evaluate the security properties of four distinct checkpointing strategies—three based on CCCP's radio transmission and one on local flash storage—and describe how CCCP provides data integrity with or without confidentiality. We describe our experimental setup and methods, then provide empirical evidence that CCCP's radio-based checkpointing requires less energy per checkpoint than a flash-based strategy. Finally, we characterize the overhead incurred by CCCP's cryptographic operations in terms of both energy and the keystream material that they consume.

3.4.1 Security Semantics

CCCP trades the physical security of local storage for the energy savings of remote storage, but its use of radio communications introduces different security properties. We consider CCCP's four operating modes in increasing order of cryptographic complexity. Note that the algorithm listings (Algorithms 1–3) describe the most computationally intensive operating mode; the other modes involve subsets of its operations.

- Under CCCP's threat model, storing checkpointed state only in local flash memory is the most secure option, since it involves no radio transmission at all. However, for reasons detailed elsewhere in this chapter, writing to flash memory is not always possible or desirable. We call the flash-only approach *Mementos* after the system [101] that inspired CCCP.
- In a mode called *CCCP/NoSec*, a CRFID sends computational state in plaintext. Under CCCP's threat model, CCCP/NoSec allows an attacker to intercept computational state and trivially recover the information it contains.
- In a mode called *CCCP/Auth*, the CRFID computes a message authentication code (MAC), attaches it to plaintext computational state, and transmits both. To trick a CRFID into accepting illegitimate state, an attacker must craft a

message that incorporates a MAC that the CRFID can verify. However, since CCCP's MAC routine incorporates keystream material that is local to the CRFID, the attacker must guess the contents of a chunk of the CRFID's keystream memory, which requires brute force under our threat model.

- In a mode called *CCCP/AuthConf*, CCCP encrypts computational state, computes a MAC, and transmits both (Algorithm 1). As with CCCP/Auth, an attacker who wants to trick a CRFID into accepting illegitimate state must find a hash collision; however, part of her colliding input must be a valid *encrypted* computational state from which the CRFID would be able to resume. Since CCCP does not reuse keystream material, the attacker is limited to brute-force search to find such an encrypted state.

3.4.2 Experimental Setup & Methods

We used a consistent experimental setup to obtain timing and energy measurements for a prototype CRFID. We programmed a WISP with a task (e.g., a flash write) and set a GPIO pin to toggle immediately before and after the task. We then charged the WISP's capacitor to 4.5 V using a DC power supply, disconnected the power supply so that the storage capacitor was the only source of energy for the WISP, and observed the task's execution and storage capacitor's voltage on an oscilloscope. We delivered energy directly from a DC power supply when taking measurements because the alternative, providing an RF energy supply, results in unpredictable and unsteady charge accumulation, making it difficult to shut off the energy supply at a precise capacitor voltage.

After watching the GPIO pin signal the beginning and end of the task, we calculated the task's duration and the corresponding change in the storage capacitor's voltage. When an operation completed too quickly to observe clearly on the oscilloscope, we repeated it in an unrolled loop and divided our measurements by the number of repetitions. Finally, we calculated per-bit energy values by subtracting

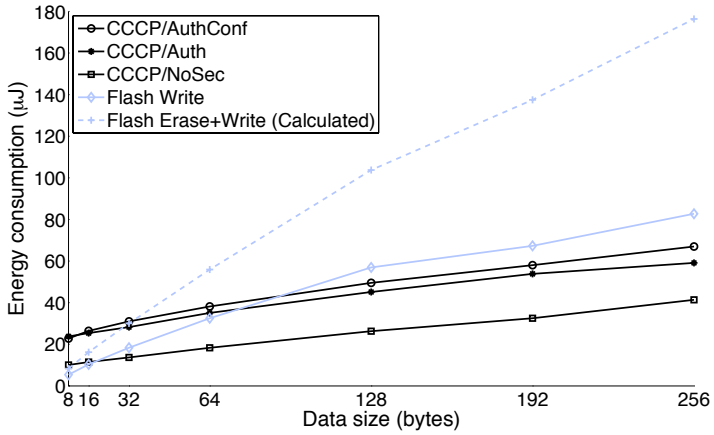


Figure 3.4. Energy consumption measurements from a WISP (Revision 4.0) prototype for all considered checkpointing strategies. Under our experimental method, we are unable to execute flash writes larger than 256 bytes on current hardware because larger data sizes exhaust the maximum amount of energy available in a single energy lifecycle. The average and maximum percent error of the measurements are 5.85% and 14.08% respectively.

the baseline energy consumption of the WISP with its MSP430 microcontroller in the LPM3 low-power (sleep) mode. We subtract the WISP's baseline energy consumption in order to discount the effects of omnipresent consumers such as RAM and CPU clocks. For all measurements that we present, we give the average of five trials.

3.4.3 Performance

Figure 3.4 shows that, for data sizes greater than 16 bytes, a checkpoint operation under CCCP/NoSec requires less energy than a checkpoint to flash. Under CCCP/AuthConf, which adds encryption and MAC operations, a similar threshold exists between 64 and 128 bytes. Checkpointing via flash has an additional cost: if the checkpointing mechanism needs to overwrite existing data (e.g., old checkpoints) in flash memory, it must erase the corresponding flash segments and potentially replace whatever data it did not overwrite. Even if a flash write does not necessitate an immediate erasure, it makes less space available in the flash memory and therefore increases the probability that a long-running application will eventually need to erase

the data it wrote—that is, it incurs an *energy debt*. In the ideal case, an application can pay its energy debt easily if erasures happen to occur only when energy is abundant—i.e., in summer power seasons. However, since CCCP is designed to address scenarios in which energy availability fluctuates, we consider the case in which each write incurs an energy debt. Factoring in debt, we characterize the energy cost of a write of size *dsize* as

$$\begin{aligned}\text{Cost}^*(\text{write}(dsize)) = \text{Cost}(\text{seg. erase}) \times \frac{dsize}{\text{Size}(\text{seg.})} \\ + \text{Cost}(\text{write}(dsize)).\end{aligned}$$

In practice, because some erasures will likely occur in summer power seasons and some in winter power seasons, the energy cost of a flash write of size *dsize* falls between Cost(write(*dsize*)) (the ideal cost) and Cost*(write(*dsize*)) (the worst-case cost), inclusive.

The energy measurements we present in this chpater (e.g., in Figure 3.4) fail in some cases to strongly support the hypothesis that radio-based checkpointing is consistently less energy intensive than flash-based checkpointing. The imbalance is due to a missed opportunity for optimization on the WISP prototype. The transistor used for backscatter modulation on the WISP (Revision 4.0) draws 500 μW of power, far more than is typical of a comparable mechanism on a conventional RFID tag. Alien's Higgs 3, a conventional RFID tag, draws only 15.8 μW of power [7] (total) during operation—an order of magnitude difference that supports an alternative design choice for future CRFIDs.

3.4.3.1 System Overhead

An application on a CRFID can balance energy consumption against security by choosing one of CCCP's operating modes:

- CCCP/NoSec imposes the least overhead because it does not encrypt data or compute a MAC; it requires no computation and consumes no keystream material. However, CCCP/NoSec imposes a time overhead to receive computational state from a reader at power-up and to transmit new state at checkpoint time.
- CCCP/Auth avoids encryption overhead (like CCCP/NoSec) but requires time, energy, and keystream bits to compute a MAC over the plaintext checkpoint. However, it requires no energy or keystream bits for encryption because it does not encrypt the plaintext checkpoint.
- CCCP/AuthConf offers the most security, since it adds confidentiality to CCCP/Auth, but the extra security comes at the expense of time, energy, and keystream bits. In this mode, CCCP encrypts the computational state before computing a MAC and transmitting both. It requires as much keystream material as the size of the state plus a constant amount for authentication.

3.5 Applications

The outsourced memory introduced by CCCP expands the design space for applications on a computational RFID. This section offers some illustrative example applications.

CRFIDs as low-maintenance sensors. Consider a *cold-chain monitoring* application for pharmaceutical supplies, in which a CRFID carries an attached temperature sensor and stores in flash memory a temperature reading each time it is scanned. To prevent exhaustion of its flash memory, the CRFID periodically computes aggregate statistics on, then discards, stored readings. Some statistical computations (e.g., computation of quartiles) require memory-intensive manipulation of the data set. If the flash memory on the CRFID considerably exceeds the size of RAM, computation of such statistics would require many writes to flash, an energy-intensive operation. An alternative is to use outsourced memory for the computation. (In the case of cold-

chain monitoring, maintaining privacy of harvested data with respect to the reader may be unessential, but the *integrity* of the statistical computation is important.)

RFID sensor networks. Recent work [22] describes *RFID sensor networks* (RSNs) that combine RFID reader infrastructure with sensor-equipped computational RFIDs. RSNs do not simply replace traditional sensor networks because of several limitations. First, they require an infrastructure of readers that provide power to sensor nodes. Second, they are constrained by the distances (several meters) at which CRFIDs currently operate. Third, because RFID communication is asymmetric, the nodes of an RSN cannot exchange information with each other except through a more powerful reader. However, there are applications for which short-range networks of batteryless sensors would be appropriate; Yeager et al. offer several examples [136].

Computational RFIDs as smartcards. Some passive RFID tags are capable of executing strong cryptographic primitives. For example, various models of the Mifare DESfire can perform triple-DES or AES, while other RFID devices can compute elliptic-curve and RSA signatures, such as the RF360 introduced by Texas Instruments [127]. The RF360 is designed to allow public-key authentication in RFID-enabled identification documents, such as e-passports.

The RF360 incorporates an MSP430, but also includes a cryptographic co-processor, and is designed to operate at relatively short range as a high-frequency, ISO 14443 device. As we show in this chapter, CCCP creates the possibility of a more lightweight device. Such a "CCCP smartcard" has two notable benefits: (1) a CCCP smartcard eliminates the cost of cryptography-specific hardware; and (2) a CCCP smartcard can operate in a mode compatible with EPC Gen 2 and achieve read ranges beyond those of a high-frequency device like the RF360.

Some smartcards are capable of performing biometric authentication—generally fingerprint verification. Match-on-card, i.e., verification of the validity of a fingerprint through computation exclusively within the smartcard, has long stood as a technical

challenge. The U.S. National Institute of Standards and Technology (NIST) recently conducted an evaluation of a range of such algorithms in contactless cards [35]. CCCP is a promising tool for expanding the class of radio devices for which match-on-card is feasible. While CCCP does not follow a strict match-in-device paradigm—given that it outsources data to a reader—it nonetheless provides comparable security assurances.

Trusted computing: outsourcing computation via TPMs. CCCP permits a computational RFID to use external memory via an RFID reader. It can support an even broader design space if we use CCCP instead for secure outsourcing not of memory, but of *computational tasks*.

Trusted platform modules (TPMs) [14, 130] offer support for such outsourcing. A TPM is a hardware device, standard in the CPUs of modern PCs and servers, that can provide a secure attestation to the software configuration of the computing platform on which it operates. Briefly stated, an attestation takes the form of a digital signature on a digest of the software components loaded onto the device. (An attestation does not provide assurance against hardware tampering or subversion of running software.)

A computational RFID can in principle make use of a TPM-enabled reader—or platform communicating with the reader—to gain secure access to a more powerful external computer. The process for such use of a TPM is subtle. The operations of verifying a TPM attestation and creating a secure session are both cryptographic operations that require computationally intensive modular exponentiation. Hence the computational outsourcing process requires CCCP as a bootstrapping mechanism.

3.6 Summary

CRFIDs enable pervasive computing in places where batteries are difficult to maintain. However, the high energy necessary to erase and write to flash memory makes

storage difficult without a constant energy source. CCCP extends Mementos [101] by exploiting the backscatter transmission common on passive RFID systems to remotely store checkpoints on an untrusted RFID reader infrastructure. CCCP protects data with UHF-based MACs, opportunistic precomputation of keystream material for symmetric cryptography, and hole punching to store a counter used to enforce data freshness. Our measurements of a prototype implementation of CCCP on the WISP tag shows that radio-based, remote checkpoints require less energy than local, flash-based checkpoints—despite the overhead of the cryptography to restore the security semantics of local, trusted storage. CCCP gives a CRFID increased storage capacity at low energy cost and enables long-running computations to make progress despite continual power interruptions that destroy the contents of RAM. Moreover, the abstraction provided by CCCP allows application developers to focus on computation rather than space, energy, and security management. Flash memory generally requires a coarse-grained, high-power erase operation before writing a new value. Our hole punching technique allows CCCP to partially reuse unerased flash memory, thus reducing the frequency with which flash memory must be erased.

This page is intentionally left blank.

CHAPTER 4

HALF-WITS: SOFTWARE TECHNIQUES FOR NON-VOLATILE EMBEDDED STORAGE AT LOW VOLTAGES

This work analyzes the stochastic behavior of writing to embedded flash memory at voltages lower than recommended by a microcontroller's specifications to reduce energy consumption. Flash memory integrated *within* a microcontroller typically requires the entire chip to operate on a common supply voltage almost double what the CPU portion requires. Our approach tolerates a lower supply voltage so that the CPU may operate in a more energy efficient manner. Energy efficient coding algorithms then cope with flash memory that behaves unpredictably.

The software-only coding algorithms proposed in this work (*in-place writes*, *multiple-place writes*, *RS-Berger codes*, and *slow writes*) enable reliable storage at low voltages on unmodified hardware by exploiting the electrically cumulative nature of half-written data in write-once bits. For a sensor monitoring application using the MSP430, coding with in-place writes reduces the overall energy consumption by 34%. In-place writes are competitive when the time spent on low-voltage operations such as computation are at least four times greater than the time spent on writes to flash memory. The evaluation of the proposed schemes shows that tightly maintaining the digital abstraction for storage in embedded flash memory comes at a significant cost to energy consumption with minimal gain in reliability. We find our techniques most effective for embedded workloads that have significant duty cycling, rare writes, or energy harvesting.

Microcontroller	CPU Min. Voltage	Flash Write Min. Voltage
TI MSP430F1232/ 149/ 413 [128]	1.8 V	2.7 V
TI MSP430F2131 [128]	1.8 V	2.2 V
PIC32M [82]	2.3 V	3.0 V
ATmega128L [9]	2.7 V	4.5 V
STM32F103C6 [120]	2.0 V	3.3 V

Table 4.1. Flash memory restricts choices for the CPU voltage supply on microcontrollers because the CPU shares the same power rail as the on-chip flash memory.

4.1 Storage on Low-Power Devices: Limitations and Challenges

While the reliability, low cost, and high storage density of flash memory make it a natural choice for embedded systems [59], its relatively *high voltage requirement* (Table 4.1) introduces challenges for energy-efficient designs aiming to maximize the system's effective lifetime (e.g., the run time on a typical battery whose voltage declines over time). Instrumenting the system to operate at a fixed low voltage v_l is one way to reduce power consumption; however, achieving *consistently correct* results for flash writes are guaranteed only if v_l is higher than a manufacturer-specified threshold. Moreover, in energy-limited devices that cannot provide a constant supply voltage, scenarios may arise in which the flash memory is the only part of the circuit whose operating requirements are not met. In such cases, applications can expect normal operation when they are not performing flash writes and unpredictable behavior when they are.

Because embedded flash memory typically shares a common voltage supply with the CPU (separate power rails are cost prohibitive), a single voltage must be chosen that satisfies different components with different minimum voltage requirements. Current embedded systems address the voltage limitations of flash memory in one of the following ways:

i) A system can choose a high supply voltage sufficient for both reliable writes to flash memory and reliable CPU operation. This is a common choice for embedded systems with on-chip flash memory, but causes the CPU to consume more energy than necessary. For example, the TI MSP430F2131 microcontroller [128] in active mode consumes almost double the power when operating at 2.2 V instead of 1.8 V. Its onboard flash memory requires 2.2 V for reliable writes to flash memory.

ii) A system can choose a low supply voltage sufficient for CPU operation, but insufficient for reliable writes to flash memory. This choice allows the energy source to last longer and for the CPU to compute more efficiently. An example of such a system is the Intel WISP [112], a batteryless RFID tag that sets its operating voltage to 1.8 V—below its onboard flash memory's 2.2 V specified minimum—to save power. Flash memory cannot be written on this device. The microcontroller could use a low-power wireless interface (e.g., RF backscatter) to store data remotely. Such an approach, however, raises privacy as well as performance concerns [109].

iii) A system can modify hardware to enable dynamic voltage scaling. This approach requires additional analog circuitry such as voltage regulators and GPIO-controlled switches. Because many embedded systems are extremely cost sensitive, this choice is unattractive for high-volume manufacturing with low per-unit profit margins. An additional 50 cent part on a thermostat control can be cost prohibitive. Moreover, small changes may necessitate a new PCB layout—upsetting the delicate supply chain and invalidating stocked inventories of already fabricated PCBs.

Approach Our approach reduces the operating voltage of the microcontroller to a point at which the resulting power savings of the CPU portion of the workload exceeds the power cost of the algorithms for ensuring reliable writes (Figure 4.1). Our low-power storage scheme benefits from the accumulative property of flash memory by repeating writes to the same cell. Each write operation will increase the chance of success by forcing some number of state transitions. That is, a failed write is still

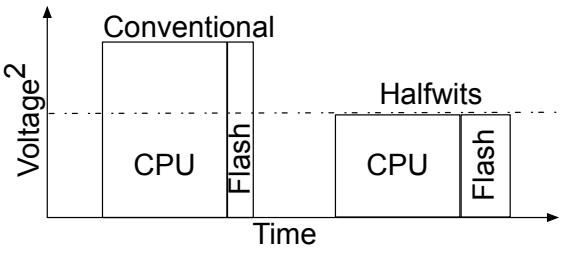
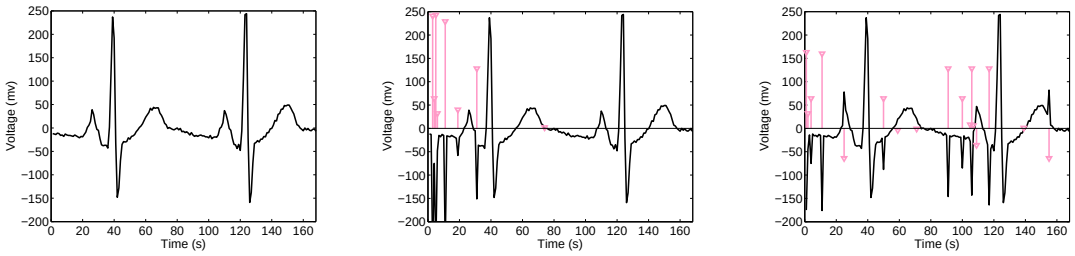


Figure 4.1. Operating at a lower voltage and tolerating errors instead of the conventional case of choosing the highest minimum voltage requirement may help decrease energy consumption. Considering that $Energy = voltage^2 \times time/resistance$, decreasing voltage decreases the energy consumption quadratically.

progress. The technique requires minimal or no hardware modification and also allows for RFID-scale and small-scale energy harvesting devices to better exploit capacitors as power supplies. The capacitor provides finite energy and therefore the voltage decays exponentially. The long tail of the curve provides insufficient voltage for conventional writes to flash memory, but it is sufficient for reliable storage with our techniques.

Of Wits and Half-Wits: Rivest and Shamir introduced the notion of write-once bits (Wits) in the context of coding theory to make write-once storage behave like read-write storage [107]. Bits in flash memory behave like wits because a programmed bit cannot be reprogrammed without calling an energy-intensive erase operation to a block of memory much larger than a single write. We coin the term *Half-Wits* to refer to wits used in a manner inconsistent with a manufacturer's specifications, resulting in stochastic behavior. Half-Wits in this work are wits of flash memory used below the recommended supply voltage.

In examining error rates at low voltage and constructing a system that provides reliable storage despite errors, our work suggests that it is appropriate to relax previously assumed constraints and reexamine the costly digital abstractions layered above on-chip flash memory.



(a) Writes at 2.0 V (b) Writes at 1.9 V (c) Writes at 1.8 V

Figure 4.2. As operating voltage decreases, flash-write errors increase. (a) shows an original ECG signal correctly stored at 2.0 V (despite operating below the recommended threshold). As the voltage decreases in (b) and further in (c), erroneous writes (light-colored spikes, height varying according to the magnitude of the error) become more common. The black line shows the reconstructed signal that includes the errors.

4.2 Behavior of Storage on *Half-Wits*

Before we can design effective coding algorithms, we must first understand the behavior of errors on Half-Wits. By tolerating a lower voltage, an energy-limited embedded device can decrease its power consumption and therefore extend its lifetime on a finite energy supply[1]. The minimum operating voltage of embedded devices that use nonvolatile on-chip storage is usually determined by the requirements of flash memory. For example, the TI MSP430 microcontroller can operate at 1.8 V, but its nominal minimum voltage for flash writing and erasure is 2.2 V (Table 4.1). Increasing operating voltage from 1.8 V to 2.2 V causes the CPU to draw about 50% more power without commensurate gain in clock speed because of the voltage squaring effect.

The cost of lowering voltage below flash memory requirements in order to save power is the extra work necessary to ensure reliable writes to flash memory. Figure 4.2 shows the result of running a MSP430F2131 at three different voltages—all lower than the nominal minimum for flash writes—to store electrocardiogram (ECG) data

[1]Or because of relaxed requirements, eliminate the need for multiple batteries in series to achieve a sufficient voltage.

samples from the PhysioNet database [46] in flash memory. Many medical sensor networks [74, 76, 116] that provide ECG measurements are energy-limited and use on-chip flash memory as primary storage.

These graphs support the intuition that flash writes may not be error-free at low voltages and that there exist voltage levels below the minimum recommended voltage at which flash writes function correctly. Moreover, a nonzero error rate may be tolerable for some applications. In the case of ECG data, the cardiac pulse interval can be recovered from noisy data stored at low voltage. To investigate the behavior of flash memory at low voltage and determine the factors influencing the error rate, we performed experiments on an automated testbed.

4.2.1 Experimental Methodology

We use a consistent experimental setup for all of the experiments in this work. Our choice of test platform is a TI MSP430 [128] microcontroller with on-chip flash memory. More specifically, we tested two types of TI microcontrollers: MSP430F2131 and MSP430F1232. The MSP430 is common in low-power embedded applications; we note especially its use in sensor motes [97] and RFID-scale batteryless devices [112, 140]. In our setup, an MSP430 microcontroller runs a test program that involves both computation and flash operation. We power the microcontroller with an external power supply held steady at a voltage below the nominal minimum for flash writes. An external chip captures the contents of flash memory after each experiment.

To automate the testing of flash write behavior, we have developed a flash memory testbed (Figure 4.3). The two major components of the testbed are a test platform and a connected monitoring platform. The monitoring platform is based on an additional MSP430 microcontroller. The test platform runs a test program at low voltage. When the test program completes, the test platform sends the result of the experiment to the monitoring chip via GPIO pins. The test and monitoring platforms share 8+1

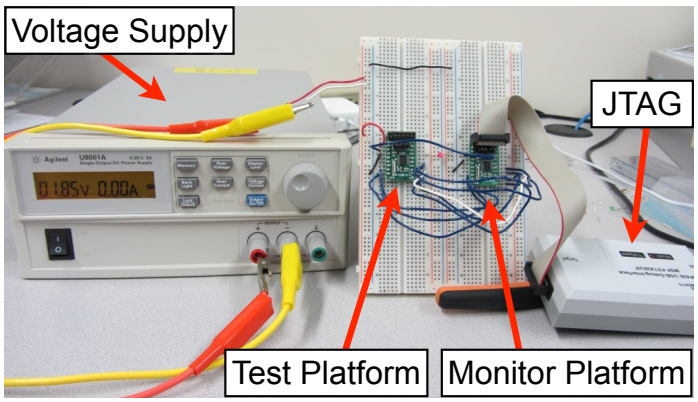


Figure 4.3. Automated flash memory testbed. A monitoring platform observes and logs the behavior of the flash memory test platform. The test platform runs at a voltage controlled by the experimenter, while the monitoring platform runs at a constant voltage within the manufacturer's specified voltage. This setup helps to automate the experiments.

GPIO pins to carry one byte of data and a clock signal. Once the test platform puts data on its eight data pins, it raises the clock pin. The monitoring chip reads data from its GPIO pins whenever it detects a rising clock signal and logs the results in its own flash memory. The monitoring chip runs at a voltage above the nominal minimum for its own flash writes, thereby storing reliably.

4.2.2 Unreliable, Low-Voltage Flash Memory Writes

The TI MSP430 [128] datasheet states that flash writes at any voltage lower than the nominal minimum voltage (which is 2.2 V in the case of MSP430F2131) are not guaranteed to succeed. However, as the graphs in Figure 4.2 show, not all flash writes fail at low voltages. On the contrary, in this specific experiment, most of the writes (95.24% at 1.9 V and 89.88% at 1.8 V) succeed.

In a NOR flash memory, all cells are initialized to 1, and writing data to a byte of flash memory means setting an appropriate number of bits to 0 by applying electrical charge to the corresponding flash cells. At low voltage, there may be insufficient charge to effect a transition to 0, and a flash write may store fewer 0 bits than

(Binary)	Intended	00001100	00001101	00001110	00010100	00100111	10100100
	Written	11101101	01011111	11111111	11111111	00101111	10101111
Hamming distance		4	3	5	6	1	3

Table 4.2. Erroneous flash writes at low voltage. Insufficient electrical charge may result in some bits failing to transition from 1 (the initial state) to 0.

requested [94]. To be specific, we define errors as follows: when a byte of data d_1 is written in a flash memory address and then data d_2 is read from that address, there is an error if $d_1 \neq d_2$. An experiment, explained next, investigates the behavior of low-voltage flash memory and gives bit-level results.

Using the automated flash testbed explained in Section 4.2.1, the test platform runs a program that writes numbers $\{0, \cdots, 255\}$ to flash memory, then sends the contents of its flash memory to the monitoring platform via GPIO pins. Table 4.2 compares the written data and the intended data for cases in which errors occurred. It demonstrates that, when both are represented as integers, the absolute value of the stored data is always greater than or equal to the absolute value of the intended data.

4.2.3 Determining Factors That Affect Error Rates

We consider the following potential factors that may affect the error rate of setting a bit to 0 in a flash memory at low voltage: voltage level, Hamming weight of the data, wear-out history, permutation of 0s, and neighbor cells. The effects of each of these variables are evaluated by designing an experiment to test a hypothesis. All the experiments are performed on flash memories with minimal previous usage unless stated otherwise.

Impact of Voltage Level: Our hypothesis is that the lower a chip's operating voltage (and that of its on-chip flash memory), the higher the error rate of flash writes. Figure 4.4 confirms this hypothesis; moreover, the graph shows that for

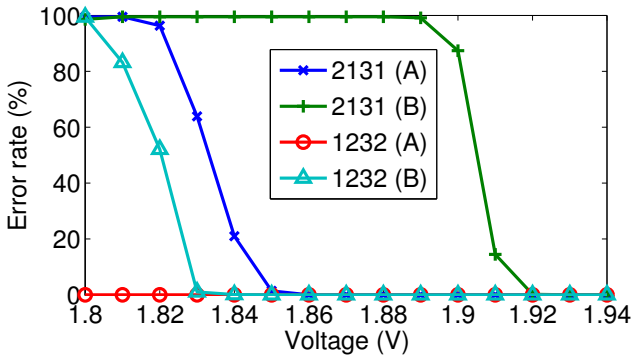


Figure 4.4. Flash write error rates decrease as voltage increases. This trend holds for all the chips (MSP430F2131 and MSP430F1232) we tested, though error rates differ even between chips of the same model.

different chips of exactly the same type, the error rate can be different even under equivalent voltages.

Experiment: Two MSP430F2131 and two MSP430F1232 microcontrollers run a program that writes zeros to the data segment of their flash memory. We increased the microcontroller's operating voltage in 10-mV steps, and used the monitoring platform to compute the byte error rates over 50 runs.

Impact of Hamming Weight: In an erased (i.e., having value 1) flash cell, writing a 1 is always error-free because no change to the cell is necessary. However, setting a cell to 0 might fail if there is not enough charge accumulated in that cell. Our hypothesis is that the lower the Hamming weight (number of 1s in the binary representation) of a number, the higher the probability of error when writing that number to flash at low voltage.

Based on per-byte Hamming weight, there are nine equivalence classes of integers that can be represented in one byte. The weight-8 equivalence class has only one member, 255, which can always be written to an erased flash cell without error. The other extreme case is the weight-0 equivalence class, containing only 0s, that requires

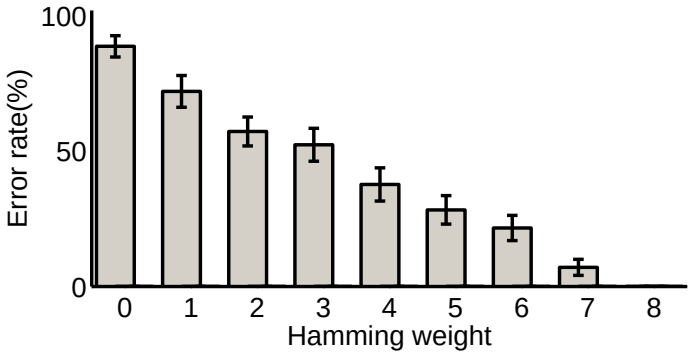


Figure 4.5. As the Hamming weight (number of 1s in the binary representation) of a number increases, the error rate of low-voltage flash writes decline. The data correspond to a MSP430F2131 running at 1.84 V.

all eight bits to transition to 0. Figure 4.5 shows the byte error rate for all nine equivalence classes, measured in the following experiment.

Experiment: At 1.84 V, a MSP430F2131 runs a program that writes numbers from the same equivalence class to one block (64 bytes) of flash memory. We used the monitoring platform to compute the average byte error rate of flash writes for each of the nine equivalence classes over 50 runs.

Corollary: To exploit the fact that the Hamming weight of a number affects error rate when written to flash, one can transform numbers into numbers with greater Hamming weights before writing them to flash memory.

Impact of Wear-out History: Flash memory has a limited lifetime (about 10^5 cycles of erasures) after which the erase operations fail to reset the bits to 1 reliably. We suspect that the more flash memory is erased (worn-out), the lower its error rate of setting bits to 0 would become. This counterintuitive hypothesis is consistent with the notion that flash erasures (settings bits to 1) become harder with wear-out. Figure 4.6 shows a heat map of bit error rate for three blocks of flash memory (192 bytes) on an MSP430F2131 microprocessor. Lighter colors in the heat map represent higher error

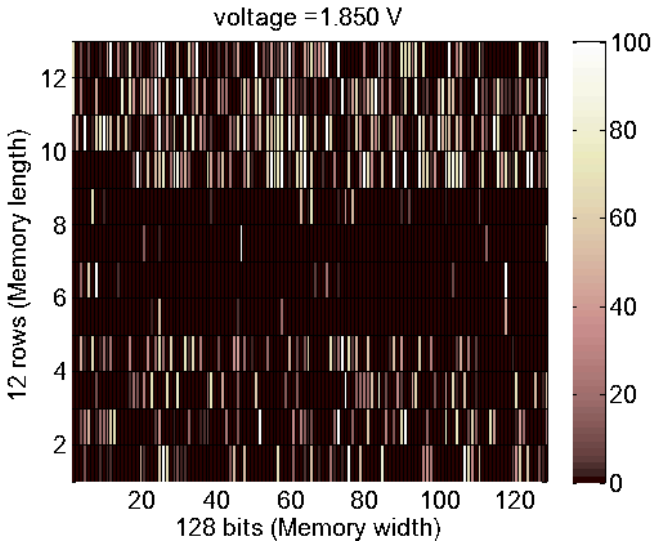


Figure 4.6. Worn-out flash memory blocks are biased toward ease of writing zeros. Lighter color represents higher average number of errors over 50 trials. The middle block has been write/erase cycled 6,000 times. The other two blocks are minimally used.

rates. The disproportionately dark color of the middle block is due to more frequent erasure of that block compared to the other two blocks.

Experiment: A MSP430F2131 runs a program that writes zeros to all three blocks of its flash memory. The MSP430 is first worn out such that one block has 6,000 write/erase cycles and two blocks have minimal previous usage. We used the monitoring platform to compute the average error rate for all bits in the three blocks of memory over 50 trials.

Corollary: Wear-out history affects error rate, so storing data in more than one location might decrease the error rate, especially if those locations are in different blocks of memory.

Impact of Permutation of 0s: Two numbers belonging to the same Hamming-weight equivalence class can have different permutations of 0 bits. We tested to see how the error rate depends on the permutation of 0s in one byte of data. For example, the numbers 240, 15, 170, and 71 all have four 0s in their binary representation but

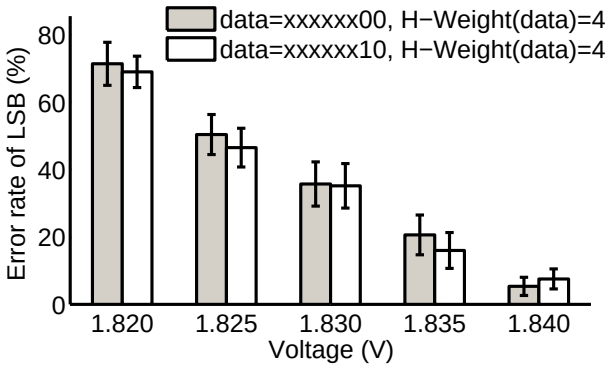


Figure 4.7. Impact of the neighbor cells on error rate is negligible. The graph shows that the value of the second LSB does not greatly affect the error rate of the LSB. The bars show the error rate of the LSB for writing numbers from the same Hamming-weight equivalence class whose two LSBs are set to either 00 (dark bars) or to 10 (light bars).

in different places (240 has 0s in the right nibble, and 15 has all of its 0s in its left nibble, etc.). The result of the experiment shows a similar byte error rate with mean of $39.85 \pm 4.29\%$ for numbers in the same equivalence class. The small standard deviation (4.29%) shows that the permutation of 0s does not significantly affect the error rate and therefore we do not consider this to be a factor in our design directions.

Experiment: A MSP430F2131 runs a program that cycles through eight numbers from the same Hamming-weight equivalence class, writing them to 192 consecutive bytes of flash memory. We used the monitoring platform to compute the average error rates for each of the 192 bytes over 50 trials.

Impact of Neighbor Cells: Another factor that might affect the error rate of storage in a flash cell at low voltage is the values of neighboring cells. However, our results suggest that a cell's error rate does not appear to depend on the values stored in neighboring cells (Figure 4.7).

Experiment: In order to determine if the error rate of a cell is affected by its neighbor, we consider all numbers from the same Hamming-weight equivalence class whose two Least Significant Bits (LSBs) are set to either 00 (case 1) or 10 (case 2).

An example of case 1 is number 60 (0b00111100) and an example of case 2 is number 30 (0b00011110). This experiment fixes the Hamming weight variable and changes the neighbor value of the LSB to be 0 or 1. We deem a write erroneous if the LSB is not set to 0. The experiment was done for a Hamming weight of four and it was repeated for five voltage levels in the interval of 1.82 V to 1.84 V with steps of 5 mV. The error rate for any voltage above 1.84 V was close to 0% and for any voltage below 1.82 was close to 100%. We used the monitoring platform to compute the average error rates of case 1 and case 2 for each of the voltage levels over 50 trials.

Impact of Temperature: Temperature is another factor that usually affects the performance of electronic components. We tested to see if the error rate of low-voltage flash writes depends on the temperature of the chip. The experimental results show that at higher temperatures, the error rate decreases. We found that for a TI MSP430F2131 operating at 1.83 V, the error rate is 63% at 25°C, while the error rate becomes negligible when the temperature goes up to 39°C. This result shows that error rate depends on environmental variables and cannot be assumed constant at a fixed voltage. Moreover, this result indicates that a microcontroller can adjust its configuration of flash writes based on the temperature to save energy.

Experiment: A MSP430F2131 runs a program that writes zeros to all three blocks of its flash memory. We used the monitoring platform to compute the average error rate for all bits in the three blocks of memory over 50 trials at a low temperature (25°C) and at a high temperature (39°C).

4.2.4 Accumulative Memory Behavior

It is helpful to understand a few details of the electrical nature of flash memory in order to appreciate the expected behavior of conventional digital abstractions when layered above embedded flash memory. Each flash memory cell is a floating-gate (FG) transistor made up of a source, drain, control gate, and floating gate. The floating

gate is separated from the source and drain by an insulating oxide layer that makes it difficult for electrons to travel into or out of the gate. Flash cells rely on this oxide to maintain logical state in the absence of power, making the memory non-volatile [94].

To write a memory cell (which has an erased value of 1), the control circuitry applies a high field to the source. The application of this field greatly increases the probability that electrons in the floating gate will tunnel to the source. If a sufficient number of electrons tunnel to the source, the cell is subsequently read as a 0. To erase a cell (that is to restore a 1), the control circuitry applies a high field to both the source and drain. This field energizes the electrons currently stored near the source, allowing them to jump the oxide barrier to the floating gate where they are once again trapped [94].

Not all electrons must transit in order for a write or erase operation to be successful. The operation only needs to change the state of some majority of the electrons so that subsequent read operations detect sufficient charge to discern the intended value. Lowering the applied voltage (and thus the field strength) lowers the probability of state change for each electron but, as noted earlier, electrons that do transit will remain in place.

A low-power storage scheme can benefit from this accumulative property by repeating writes to the same cell. Each write operation will increase the chance of success by forcing some number of state transitions. In other words, a failed write is still progress.

4.3 Design of a Low-Voltage Storage System

This section presents our design for a software system that enables reliable flash memory writes at low voltage. We first present a model that captures the basic characteristics and behavior of flash memory. We then set design goals for the model under consideration. We introduce three methods for reliable flash storage, which we

refer to as *in-place writes*, *multiple-place writes*, and *RS-Berger codes*. Each method aims to meet our design goals for reliable non-volatile storage.

4.3.1 Modeling Low-Voltage Flash Memory

A NOR flash memory has a set of n cells that are initially set to 1. We represent the state of the cells by $c_1, \ldots, c_n$; the value of c_i can be 0 or 1. A cell can be set to 0 using a write operation. The $1 \rightarrow 0$ transition might fail at low voltage while the $1 \rightarrow 1$ will obviously succeed. Flash memory at low voltage, where errors occur only in one direction, can be modeled as a Z-channel [69]. Flash memory is a write-once memory [107] and once a cell is set to 0 (i.e., once it is programmed), it cannot be changed back to 1 without using an erase operation. In flash memory, cells are organized by blocks, and an erase operation resets an entire block of cells. Block erasures are costly in terms of time and energy and they cause wear to flash cells.

Operations: There are two operations in this model: (1) An update operation that changes a subset of cells to 0 to represent a value, and (2) A decoding operation that maps cell states (i.e., memory state) to a value. Updating a variable means changing the values of $c_1, \ldots, c_n$ to $c'_1, \ldots, c'_n$. Assuming that no erase operation occurs, and therefore no bits are reset to 1 after being set to 0, we have $\forall i \in \{1, \ldots, n\}, c_i \geq c'_i$ after an update. If the update operation is performed when operating voltage is below the nominal minimum required for flash memory, the update operation may not be error-free.

4.3.2 Design Goals

Our storage techniques, which aim to provide reliable storage for low-power devices, are designed with the following metrics in mind:

- *Error rate:* The first and foremost design goal is to minimize the error rate to provide applications with reliable non-volatile storage.

- *Energy consumption:* The energy consumed to achieve an acceptably low error rate should not exceed the expected energy savings gained by running at a lower voltage.
- *Delay:* We define delay as the difference between the execution time to store data reliably at a low voltage and to store the same data at a high voltage. The delay caused by the storage technique should be reasonably small.

4.3.3 Proposed Methods

Toward the design goals discussed previously, we propose methods to deal with errors caused by using flash memory at low voltage.

4.3.3.1 In-Place Writes

Since the transition of a 1 to a 0 in a NOR flash memory at low voltage is stochastic rather than guaranteed, the *in-place writes* method repeats the write of each byte (to the same memory location) more than once if error occurs, up to a *threshold* number of attempts. Algorithm 4 gives the details for ENCODE and DECODE procedures for in-place writes. The *in-place writes* makes an attempt to write a byte into memory, reads that memory address, and if the read result does not match the attempted write value, the algorithm makes another attempt to write that value to the same memory address. The write attempts can be controlled using the *threshold*.

The reason *in-place writes* decrease the error rate is that, as explained in Section 4.2.4, each write attempt in the same memory location increases the accumulated charge and therefore raises the probability of storing the intended bit sequence successfully.

4.3.3.2 Multiple-Place Writes

Another approach to increase the reliability of flash writes at low voltage is to write a value to more than one location in flash memory if error occurs up to a *threshold* number of locations. Later, to retrieve the stored data, the *multiple-place*

Algorithm 4 The encoding and decoding algorithms for the *in-place writes* method to store *data* to *address* by repeating the writes up to a *threshold* number of attempts if necessary.

```
ENCODE(data, address, threshold)
1  WRITE_TO_FLASH(data,address)
2  result ← READ_FROM_FLASH(address)
3  repeat ← 1
4  while (result ≠ data) AND (repeat < threshold)
5        do WRITE_TO_FLASH(data,address)
6           result ← READ_FROM_FLASH(address)
7           repeat ← repeat +1

DECODE(address)
1  result ← READ_FROM_FLASH(address)
2  return result
```

writes method reads the data from the specified address and several other addresses associated with it, then returns the bitwise AND of all of the stored values. Algorithm 5 details ENCODE and DECODE procedures of the *multiple-place writes* method. The *multiple-place writes* makes an attempt to write a byte into one memory address, reads that memory address, and if the read result does not match the attempted write value, the algorithm makes another attempt to write that value to a different memory address. The write attempts can be controlled using the *threshold*.

The *multiple-place writes* approach can decrease the error rate because: All cells of flash memory are initially set to 1. An error means that writing a 0 has failed and a bit cell c_i has remained untouched (logical 1) although it was intended to be set to 0. If the cell write in one of the locations has not failed, and cell c_i is 0 in at least one location, getting the AND of the read values from all locations will make cell $c_i = 0$ in the AND result. The case of writing a 1 to a cell does not cause an error since it means changing a cell from 1 to 1.

Algorithm 5 The encoding and decoding algorithms for the *multiple-place writes* method to store *data* to *address* by repeating the writes up to a *threshold* number of locations if necessary. The distance between each of these associated locations is *offset*.

```
ENCODE(data, addr, threshold, offset)
1  WRITE_TO_FLASH(data,addr)
2  result ← READ_FROM_FLASH(addr)
3  repeat ← 1
4  while (result ≠ data) and (repeat < threshold)
5        do phy_addr ← addr + (repeat × offset)
6           WRITE_TO_FLASH(data,phy_addr)
7           n_result ← READ_FROM_FLASH(phy_addr)
8           result ← result & n_result
9           repeat ← repeat +1

DECODE(addr, threshold, offset)
1  for i ← 0 to (threshold −1)
2        do phy ← addr + (i × offset)
3           n_result ← READ_FROM_FLASH(phy)
4           result ← result & n_result
5  return result
```

4.3.3.3 RS-Berger Codes

Our third method to provide reliable flash memory at low voltage involves data coding. We use the concatenation of Reed-Solomon [103] and Berger [13] codes—which we call *RS-Berger codes*—to detect and correct errors at read time (Algorithm 6).

Reed-Solomon is a widely used error-correcting code that can correct twice as many erasures as errors. There are three parameters (n, k, d) accompanying the Reed-Solomon (RS) code. The parameter n is the total number of symbols in the codeword, and k is the number of information symbols in the codeword, and the parameter d is the minimum hamming distance of two codewords in the codebook. These three parameters should satisfy the following conditions: $d = n - k + 1$.

Algorithm 6 The encoding and decoding algorithms for *RS-Berger codes* write method. t is the maximum number of errors RS-Berger code can correct.

```
ENCODE(data_{1,..,N}, n)
1  for i ← 1 to N
2        do CW_i ← RS_ENCODE(data_i, n)
3           WRITE_TO_FLASH(CW_i, address_i)
4  for i ← 1 to n
5        do for j ← 1 to N
6                 do sym_{i,j} ← CW_j(i)
7           chk_i ← BERGER_ENCODE(sym_{i,(1,..,N)})
8           WRITE_TO_FLASH(chk_i, address_{N+1} + i-1)

DECODE(addr_{1,..,(N+1)}, n, t)
 1  for i ← 1 to N
 2        do chk_i ← READ_FROM_FLASH(addr_{N+1}+i-1)
 3  for i ← 1 to N
 4        do CW_i ← READ_FROM_FLASH(addr_i)
 5           for j ← 1 to n
 6                 do sym_{i,j} ← CW_i(j)
 7  errors ← {}
 8  for i ← 1 to n
 9        do err ← BERGER_DECODE(sym_{i,(1,..,N)}, chk_i)
10           if err = 0
11              then errors ← errors ∪ {i}
12  if |errors| ≤ t
13     then for i ← 1 to N
14                 do result_i ← RS_DECODE(CW_i, errors)
15           return result
16     else  return "fail to correct errors"
```

A (n, k, d)-RS code can correct up to $\frac{n-k}{2}$ errors and up to $n-k$ erasures. Therefore, if the locations of errors are known, an RS code's error-correcting capacity is improved twofold.

To detect the location of errors and therefore to improve the efficiency of the RS code, we use a Berger code, an error-detecting code that can detect all asymmetric errors [13]. As previously mentioned (Section 4.3.1), flash memory at low voltage can be modeled as a Z-channel for which a Berger code is suitable. A Berger codeword consists of two parts: k information bits and $\lceil \log_2(k+1) \rceil$ check bits. The check

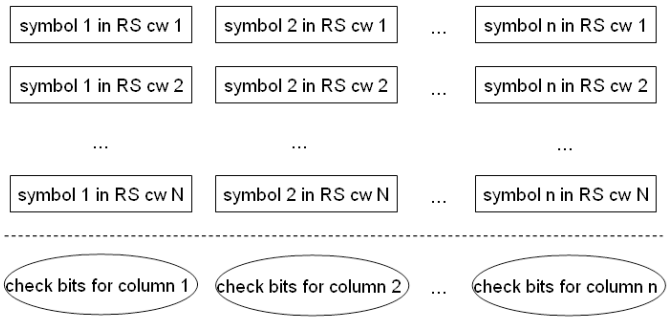


Figure 4.8. Structure of the input/output sequence of the Berger code.

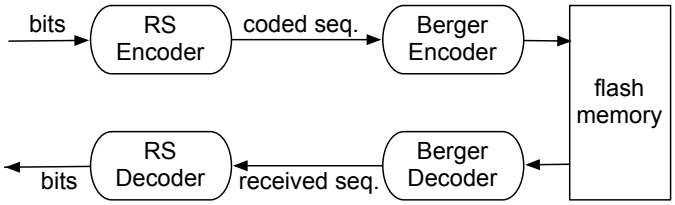


Figure 4.9. A diagram representing the RS-Berger code. An RS-Berger code is the concatenation of the Reed Solomon code and a Berger code.

bits of the Berger codeword represents the number of zeros in the k information bits. Berger code can detect any number of zero-to-one errors, as long as no one-to-zero errors occur in the same codeword.

The Berger code can detect all zero-to-one errors, because the number of zeros in the information-bit component will always be less than the number represented by the check-bit component.

We use an $(N+1) \times n$ matrix to represent RS-Berger codes (Figure 4.8). This matrix has N RS codewords, each of which has n symbols. Each symbol (m bits) is filled in one entry of the matrix. Each column of the matrix, consisting of $m \times N$ bits, supplies the information bits for one Berger code block. After Berger encoding, the $(N+1)$th row records the check bits for the Berger codes.

Figure 4.9 shows how the data are encoded and decoded using our *RS-Berger code*. When encoding the data, we first use RS code to generate n codewords (rows of the

matrix) and then we apply a Berger code to compute the check bits for each symbol for all codewords (each column of the matrix). When decoding data, we first use the Berger decoder to check whether or not each column is erroneous. If one entry in the column is erroneous, we consider all the symbols in the column erasures; otherwise, all the symbols in the column are considered correct. Then, once the error locations are known, we apply RS decoding to correct the erroneous sequences row by row.

4.4 Evaluation

Our storage techniques are designed for the resource limitations of low-power devices. In this section, we first evaluate the suitability of the three methods proposed in Section 4.3.3 for low-power devices; we then evaluate the hypothesis that for CPU-bound workloads, operating at low voltage and managing errors is more energy-efficient than fixing the operating voltage to the maximum of all the components' nominal minimum voltages.

Summary of results: For a sensor monitoring application that reads 256 data samples from flash memory, aggregates data, and stores the results in flash memory, use of *in-place writes* at 1.8 V reduces the energy consumption up to 34% versus running the same application at 2.2 V (minimum voltage requirement for the flash memory). This sensing application models a common workload for both wireless sensor nodes and RFID-scale devices.

Experimental setup: We used a consistent experimental setup to measure the energy consumption and execution time of each program. Using an oscilloscope, we measured the voltage of a small resistor in series with a MSP430 microcontroller programmed with a task (e.g., a flash write). The integration of the current (voltage divided by the resistance) over the execution time of the task multiplied by the operating voltage of the device gives the energy consumption of that task ($Energy =$

$\int I(t)\ dt \times V$). To facilitate precise identification of the task on the oscilloscope, the microcontroller toggled a GPIO pin immediately before and after the task.

4.4.1 Comparison of the Proposed Storage Methods

The workload used to measure the performance of each of the proposed methods is the storage of accelerometer traces—generated using the Intel WISP 4.1's 10-bit ADC sensor—to flash memory. The input trace is a series of three-dimensional 16-bit samples containing ten bits of information. We used a simple data compression method to store more data in the available flash memory. The compression method involved reading four samples of data, preparing the first byte of each sample to be stored in flash memory, then combining the remaining two bits of each sample into one byte of data. Using this compression scheme, we reduced every four samples (eight bytes) to five bytes.

The maximum number of write attempts for both *in-place writes* and *multiple-place writes* methods were set to two. The *RS-Berger codes* used three codewords of size 38 bytes (32 bytes data and 6 bytes parity). These settings enable all three methods to fit their data in 192 bytes of flash memory. Table 4.3 shows the energy consumption and time taken for the same workload under each method. Both *in-place writes* and *multiple-place writes* consume less energy and finish more quickly at 1.9 V than at 1.8 V. Both of these methods are feedback-based and repeat writes if they detect errors. Because there is a lower chance of error at 1.9 V, fewer rewrites are required than at 1.8 V, so less energy and time are required.

The *in-place writes* method slightly outperforms the *multiple-place writes* method at both voltage levels because its decoding procedure is less CPU-intensive. The *In-place writes* method has the best Error Correction Rate (ECR in Table 4.3) of all. The *multiple-place writes* method seems to be the most suitable when there are some memory cells that are hard to program and therefore rewriting in those cells is not

Method	V	Time (ms)	E (μJ)	ECR
In-place	1.8	24.16	59	96%
M-place	1.8	25.00	63	84%
RS-B	1.8	334.45	160	0%
In-place	1.9	15.43	38	100%
M-place	1.9	16.85	40	100%
RS-B	1.9	334.73	180	100%

Table 4.3. Performance comparison of the proposed methods at 1.8 V and 1.9 V. Error Correction Rate (ECR) shows the effectiveness of the methods.

helpful (Figure 4.6 gives an example of such a case). Compared to *RS-Berger codes* which always guarantee that a certain number of errors can be corrected, the *in-place writes* and *multiple-place writes* methods are less reliable—they offer no such guarantees. Therefore, for applications with a hard reliability requirement, *RS-Berger codes* may be more suitable if the application knows the error rate in advance and is willing to incur extra computational costs for RS-Berger encoding and decoding.

Error Correction Rate: As Table 4.3 illustrates, the two methods that do not use coding—in-place writes and multiple-place writes—incur similar energy consumption costs. We now compare the effectiveness of these two approaches with respect to the error correction rate.

Figure 4.10 and Figure 4.11 demonstrate that flash storage reliability improves as we increase the number of repeated writes/places at five different voltage levels (all below the nominal minimum voltage for flash writes).

Experiment: Using our automated testbed, the test platform runs a program that writes zeros to 192 consecutive bytes of flash memory (using *in-place writes* and *multiple-place writes* methods in two different experiments). We increase the maximum number of repeated writes from one to ten, one unit at a time. The monitoring platform counts the number of incorrectly stored bytes (those that are

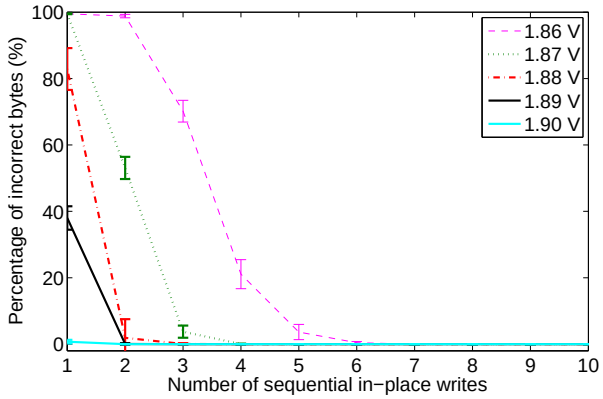


Figure 4.10. Reliability improvement using *in-place writes* over five different voltages.

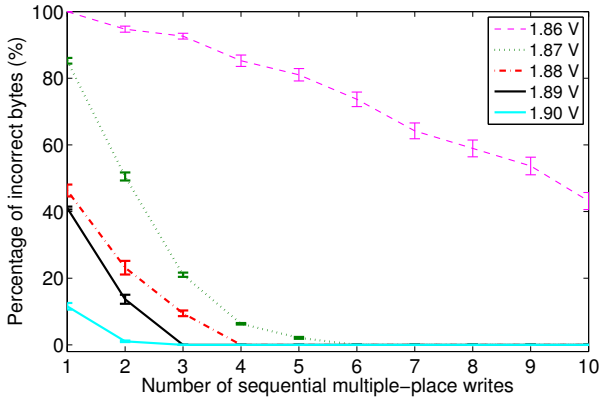


Figure 4.11. Reliability improvement using *multiple-place writes* over five different voltages.

not set to zero after the experiment). The experiment was repeated for five different voltages (1.86 V–1.90 V).

Figure 4.12 compares the error rate of the *in-place* and *multiple-place* write methods. We choose the same maximum number of repeated writes for both approaches. As the graph shows, the *in-place writes* method improves the error rate more dramatically. We attribute this phenomenon to the fact that electrons accumulate in flash cells with each programming attempt. Figure 4.12 also allows us to evaluate hybrids of the *in-place writes* and *multiple-place writes* methods. For example, choosing one

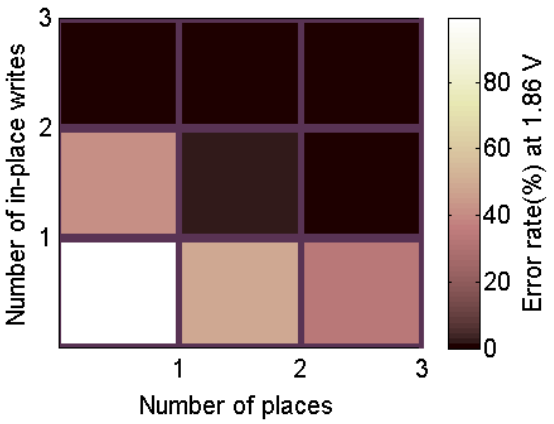


Figure 4.12. The *in-place writes* method reduces the error rate more effectively than do the *multiple-place writes* method or a hybrid of both methods.

place to write the value and repeating the write up to three times (up to three writes in total) works better than repeating the write up to twice in two places (up to four writes in total). This graph offers evidence that a pure *in-place writes* approach works better than a hybrid approach or a pure *multiple-place writes* approach. However, we do not conclude that the *in-place writes* method always outperforms the *multiple-place writes*. A winning case for *multiple-place writes* is when a flash memory has unbalanced blocks (different error rates), for example, the chip shown in Figure 4.6. While the *multiple-place writes* method requires more space, it could provide a more reliable storage compared to *in-place writes*.

4.4.2 *Half-Wits* Versus Wits in Practice

To evaluate our storage schemes, we consider three test cases representing CPU operations, flash read operations, and flash write operations.

The `RC5` [106] test case, a CPU-only workload, is a commonly used encryption algorithm that can cope with the resource limitations of low-power devices [27, 67]. `RC5` was implemented with a 32-bit word size, 18 rounds, and 16 bytes of secret key.

The `retrieve` and `store` test cases are both I/O-bound tasks. One reads and the other one writes 192 bytes of data from/to flash memory. CPU-bound operations in these test cases are minimal (essentially only a loop that calls a function to flash memory). The `store` program uses *in-place writes* with a maximum number of three (re)writes to deal with errors. Because flash read operations are fundamentally simpler than flash write operations, flash reads are reliable at low voltage.

We run each of the three test cases on a MSP430F2131 microcontroller at four different voltages that are all in the operating range of this microcontroller (1.8 V–3.5 V). Two voltage levels are below the recommended threshold for flash memory: 1.8 V and 1.9 V. Two voltage levels are at and above the recommended threshold: 2.2 V and 3.0 V. The microcontroller is set to work at its highest possible clock rate for each voltage level in order to gain the best energy performance. Figure 4.13 compares the normalized average energy consumption over five trials of each test case at each voltage. The energy has been normalized to the energy consumption of each workload at 3.0 V. By running at 1.8 V (below the nominal minimum voltage for flash writes on the MSP430F2131), the microcontroller consumes 48% and 33% less energy to finish the `RC5` and `retrieve` test cases respectively. However, our storage schemes do not seem beneficial for flash-write-intensive tasks (the `store` test case).

To evaluate the end-to-end performance of our storage methods, we have tested a sensor-monitoring application that is CPU-intensive and can benefit from a low-voltage storage. This application reads from flash memory 256 accelerometer samples (each ten bits); computes the maximum, minimum, mean, and standard deviation of the samples; and stores the aggregate information in flash memory. This monitoring application is a blend of CPU and I/O, but it is still a CPU-intensive workload. Table 4.4 shows that providing the system with a low-voltage storage mechanism via our methods helps to decrease the task's total energy consumption by 34%.

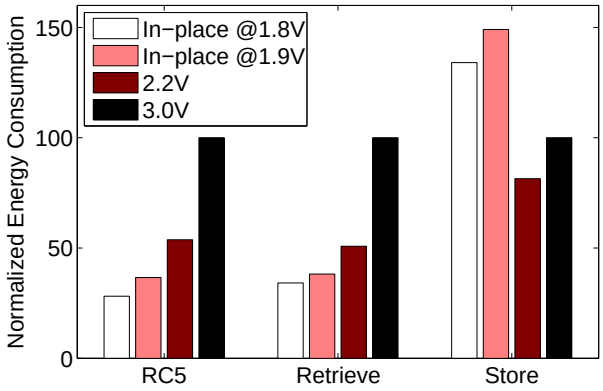


Figure 4.13. Micro-benchmarks: CPU (`RC5`), read (`retrieve`), and write (`store`) normalized energy consumption measured at four different voltage levels. The energy has been normalized to the energy consumption of each workload at 3.0 V. Although the `RC5` and `retrieve` test cases consume less energy at low voltage, this is not the case for the `store` test case (a write-intensive application) as the savings due to running the chip at low voltage do not compensate for the energy cost required to correct errors.

4.4.3 Finding a Crossover Point

We can empirically find the point at which the energy saved on computation compensates for the added cost of repeated flash writes. We compare a workload executed at 2.2 V to the same one running at 1.8 V using the *in-place writes* scheme with the threshold k set to 2. We make the worst-case assumption that all data must be written to flash twice (i.e., no bits change on the first attempt). The time spent

Method	**Clock Rate** MHZ	**Energy** μJ	**Time** ms
In-place, 1.8 V	6	270	151.15
In-place, 1.9 V	6	300	151.32
Standard, 2.2 V	8	410	113.24
Standard, 3.0 V	14	760	64.72

Table 4.4. Energy consumption and execution time for the accelerometer sensor application. At voltages below the recommended (1.8 V and 1.9 V), *in-place writes* method with a threshold of two is used.

on flash writes while running at 1.8 V is then twice the time spent when operating at 2.2 V. We also assume that the clock rate of the system is set to the highest specified for the CPU at each voltage. Specifically, the clock rate would be set to 6 MHz at 1.8 V and to 8 MHz at 2.2 V.

We empirically determined the power consumption of CPU and flash writes with 1.8 V and 2.2 V voltage supplies. $P_{C\_1.8} = 1.8\ mW$, $P_{C\_2.2} = 3.4\ mW$, $P_{F\_1.8} = 3.7\ mW$, and $P_{F\_2.2} = 5.8\ mW$. The variables T_C and T_F are the time spent in computation and on flash memory respectively. With these assumptions, we can write the following inequality to determine whether a given workload is likely to result in reduced energy consumption:

$$
\begin{gathered}
Energy_{1.8} \leq Energy_{2.2} \Rightarrow \\
P_{C\_1.8} \times T_{C\_1.8} + P_{F\_1.8} \times k \times T_{F\_1.8} \leq \\
P_{C\_2.2} \times T_{C\_2.2} + P_{F\_2.2} \times T_{F\_2.2} \Rightarrow \\
P_{C\_1.8} \times \tfrac{8MHz}{6MHz} \times T_{C\_2.2} + P_{F\_1.8} \times k \times \tfrac{8MHz}{6MHz} \times T_{F\_2.2} \leq \\
P_{C\_2.2} \times T_{C\_2.2} + P_{F\_2.2} \times T_{F\_2.2}
\end{gathered}
$$

The solution with $k = 2$ is $T_{C\_2.2} \geq 4 \times T_{F\_2.2}$. Therefore, *in-place writes* are competitive over normal flash writes when the time spent on low-voltage operations like computation is at least four times greater than the time spent on flash writes.

In order to verify the cross-over point with experiments, we ran several tasks with the percentile usage of 0% to 100% of flash memory. For simplicity, the example treats flash erase to be close enough in power consumption to flash write to be considered in the same category. The remainder of the workload is all computation. Energy is computed for several different distributions of work to prevent any assumptions about how much time is spent in each state. A one-second task duration is used to give data in unit time.

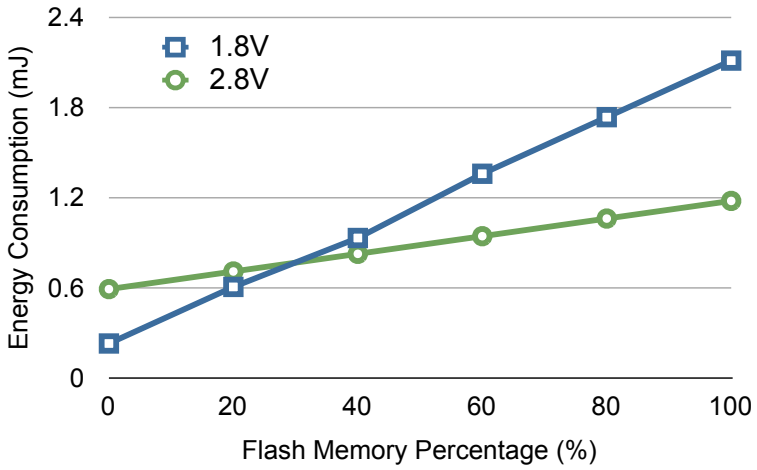


Figure 4.14. Energy consumption of TI MSP430 microcontroller for different workloads of flash memory. . The experimental results verify the calculation of finding the cross-over point at which *in-place writes* are competitive over normal flash writes. The cross-over point is where the time spent on computation is at least four times greater than the time spent on flash writes.

4.5 Applications

Most battery-powered and batteryless electronic devices use low-power microcontrollers. Any embedded device whose CPU and non-volatile storage share the same power rail might benefit from our low-voltage storage techniques.

4.5.1 Battery-Powered Electronic Products

We have compiled a list of more than one hundred low-power electronic products that choose flash memory as their non-volatile storage. By reverse-engineering these devices, we looked at three characteristics: the microcontroller inside, the use of flash memory, and the operating voltage. These three characteristics of a device determine if the power consumption can be reduced by operating the device at a lower voltage. An example of these devices is a smoke detector [68] which uses a Microchip PIC16F883 microcontroller [83]. This microcontroller has 4 KB of flash memory that gets used to store data for legal issues. The flash memory's nominal minimum voltage is 4.5 V, while the minimum requirement for the CPU is 2.0 V.

The measured operating voltage of the microcontroller is 4.7 V, which exceeds the minimum requirements of both the CPU and the flash memory. The high operating voltage might have been chosen for other purposes in addition to flash memory, but it is interesting to consider how the chip would behave at lower voltages.

4.5.2 Batteryless RFID-Scale Devices

There is another class of applications which run on batteryless devices that harvest their energy from a variety of sources—solar [49, 72, 65], kinetic [81], and RF [102, 23]. Energy is a main design constraint for all of these energy-harvesting devices. Most of these RFID-scale devices run environmental monitoring applications that are CPU intensive, which makes them ideal for our low-power storage schemes.

An example of an RFID-scale device is the Intel WISP [112], a batteryless RFID tag that sets its operating voltage to 1.8 V to save power. Flash memory cannot be written on this device since the operating voltage is below its onboard flash memory's 2.2 V specified minimum.

4.6 Improvements and Alternatives

This section describes several complementary ways to further improve the performance of our schemes.

4.6.1 Slow Writes

This method is similar to *in-place writes* in that it tries to accumulate enough charge in a cell to present a zero bit. However, instead of writing a bit multiple times, this method writes a bit once but slowly. The extra time allows for more charge to get stored in a cell. The idea is similar to Dynamic Voltage and Frequency Scaling (DVFS) methods [32]. Algorithm 7 details the simple ENCODE and DECODE procedures. One way to improve the *slow writes* is to choose a frequency level based on the operating voltage as well as the temperature and then try to adjust the frequency

based on the error rate. If the error rate is too high, the frequency has to be set to a smaller number to reduce the speed.

Algorithm 7 The encoding and decoding algorithms for the *slow writes* method to store *data* to *address* by slowing the write *threshold* number of times.

```
ENCODE(data, address, threshold)
1  SLOW_FLASH_CLOCK(threshold)
2  WRITE_TO_FLASH(data,address)

DECODE(address)
1  result ← READ_FROM_FLASH(address)
2  return result
```

Figure 4.15 shows the error rate of *slow writes* based on the voltage level and the speed of the writes. The speed of the writes has been adjusted by tuning the frequency of the flash memory. For an operating voltage as low as 1.80 V, the average error rate eventually drops to about zero percent if the speed is slowed down enough (In the case of this particular chip when the frequency is set to 285 KHz).

Since low-power embedded devices have a limited amount of energy available, saving power is usually a higher priority than reducing the delay. *Slow writes* follows this principle and reduces the speed of the writes in order to reduce the power consumption of the device. *Slow writes* would be beneficial specially for CPU-intensive applications in which their frequent use of CPU would cost less power while their rare flash memory use will be slower.

Experiment: A TI MSP430F2131 microcontroller runs a program that writes zeros to the data segment of its flash memory (192 bytes). We increased the microcontroller's operating voltage in 10-mV steps and increased the frequency of flash writes from 260 KHz to 1104 KHz. We used the monitoring platform to compute the byte error rates over 50 runs.

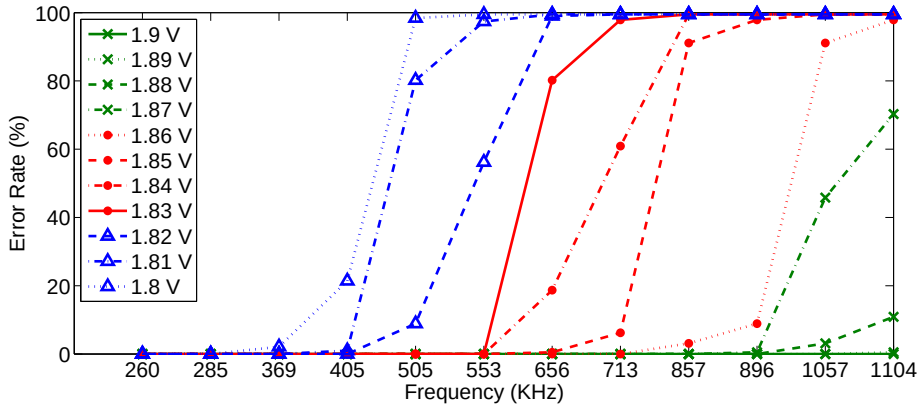


Figure 4.15. Error rate declines when the writes are performed at a lower frequency. For a voltage level as low as 1.8 V, the average error rate becomes zero when the writes are performed at 260 KHz while for 1.9 V (still well below the recommended voltage), the average error rate is zero even if the memory is used at full speed.

4.6.2 Hardware

One could add an adjustable voltage regulator [96] and about a dozen other analog components such that software could toggle a GPIO for discrete dynamic voltage scaling. A feedback loop that dynamically adjusts a voltage supply could help identify the minimum voltage at which no write errors are detected, but such boundaries can vary with temperature and wear-out. Thus, our coding algorithms would remain helpful to cope with potential errors. Our work seeks to avoid hardware modification that would require additional components or design changes to a Printed Circuit Board (PCB) because embedded applications are often cost-sensitive. Changing the PCB layout may require a manufacturer to flush its supply chain of parts typically manufactured in high volume. If an inexpensive, software-only approach with minimal disturbance to manufacturing can lead to significant savings in energy consumption, then it is hard to financially justify an expensive hardware approach that offers only comparable performance.

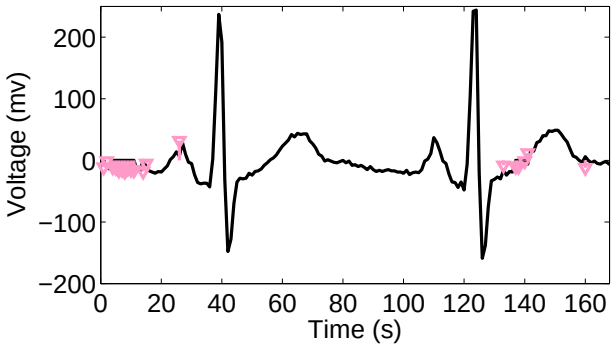


Figure 4.16. ECG data stored in flash memory at 1.89 V (the same chip from Figure 4.2) improved by using a sign bit. The light-colored bars show the difference between the ECG stored at low voltage and the original ECG data.

4.6.3 Sign Bits and Storing Complements

As discussed in Section 4.2.3, one of the major factors influencing the error rate is the Hamming weight of a number. One way to improve the performance of the low-voltage storage methods is to store numbers with greater Hamming weights (*weight* ≥ 4) in flash memory. If a number is *lightweight* (*weight* < 4), the complement of the number would be stored and a *sign bit* would be set for future data access. An array of sign bits can be stored separately from the data to avoid disturbing word alignment. A previous work [92] uses a similar technique for multi-level cell (MLC) flash memories with four levels; their techniques result in a significant decrease of energy consumption. Figure 4.16 shows that using the *sign-bit* scheme decreases the error rate at low voltage for the same ECG data used in Section 4.2. For this specific example, out of 168 bytes of ECG data, 160 bytes are *overweight*; therefore using the *sign-bit* scheme greatly decreased the error rate. The *sign-bit* approach involves very lightweight computation (counting the number of ones) and increases the number of writes by a factor of one-eighth. Therefore, the effect of this improvement on energy consumption and delay should be comparatively small.

4.6.4 Memory Mapping Table

To exploit the fact that numbers with greater Hamming weights have a lower probability of error, we can also map the most frequently used numbers in the user's data to the *heavier* numbers. The solution we suggest is to preprocess the data to sort numbers based on their frequency of use. A simple memory mapping table would map the most frequent numbers to the heaviest numbers. Such a table could be preloaded in flash memory so that storing the table would not consume energy at run time. Use of a memory mapping table would only increase the number of reads and would not increase the number of writes. Therefore, the energy consumption overhead and the delay should be smaller than the *sign bit* method.

4.6.5 An Ideal, Unrealizable Scheme

We initially tried to set the voltage to a level lower than recommended but high enough to avoid errors. This method could not be realized for two reasons: finding a voltage that satisfies this condition requires a large number of experiments per chip—error rate varies chip by chip (Figure 4.4)– and the error rate of flash writes varies depending on its lifespan and its environment (Section Section 4.2.3).

4.7 Summary and Future Work

The high voltage requirement of on-chip flash memory is a barrier to reducing the total energy consumption of low-power devices. This work examines the main factors affecting the behavior of flash memory at low voltage. Based on our observations of flash memory behavior at low voltage, we proposed three storage schemes—*in-place writes*, *multiple-place writes*, and *RS-Berger codes*—that aim to make flash memory available and reliable at low voltage while tolerating the resource limitations of low-power devices. Our evaluation shows that in-place writes can save 34% of energy consumption for a sensing workload on the MSP430 microcontroller. Our storage

techniques enable battery-powered devices to require fewer or smaller batteries or to become batteryless. Low-voltage storage would also help increase the lifespan and decrease the manufacturing cost of sensor devices.

Future work includes finding more energy-efficient coding schemes to combat flash write errors caused by low voltage. Currently, the system cannot take full advantage of dynamic voltage scaling. Another plan is to introduce benchmarks for the storage systems of low-power devices. The standard benchmarks used to evaluate the storage systems designed for desktop computers are not immediately applicable to the low-power domain.

This page is intentionally left blank.

CHAPTER 5

TARDIS: SRAM-BASED TIME KEEPING FOR EMBEDDED DEVICES WITHOUT CLOCKS

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 could better protect against attacks that depend on the existence of a rate-unlimited encryption oracle.

In this chapter, I present the TARDIS (Time and Remanence Decay in SRAM), a technique to help 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. Key challenges to implementing a practical TARDIS include compensating for temperature and handling variation across hardware.

This work 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.

Platform	Attack	# of Queries
MIFARE Classic	Brute-force [43]	≥1,500
MIFARE DESFire	Side-channel [91]	250,000
UHF RFID tags	Side-channel [90]	200
TI DST	Reverse eng. [20, 19]	∼75,000
GSM SIM card	Brute-force [45]	150,000

Table 5.1. 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.

5.1 Time Keeping in Intermittently-Powered Devices

Unawareness of time has left smart cards vulnerable to a number of successful attacks (Table 5.1). For instance, Oswald et al. [91] recently demonstrated how to extract the 112-bit key from a MIFARE DESFire contactless smartcard (used 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 [55]. Oren and Shamir [90] 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. [19] 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 [20].

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.

[1]No relation to the Clipper Chip [71].

5.1.1 Security Threats Due to Absence of a Trustworthy Sense of Time

A typical secure communication between a reader and a tag is shown in Figure 1.3. 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 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 [88]), 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 [42, 121]. This option introduces security concerns and may either require significant infrastructure or severely limit range and mobility.

5.1.2 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 [80]

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 [43, 91]. 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 [53] to distort the speed of memory decay; and (2) power-up attacks that keep the tag partially powered to prevent memory decay.

5.2 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 5.2. 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 5.1 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 5.5 with a 10 μF capacitor at $26°C$. No block transfer computation was necessary.

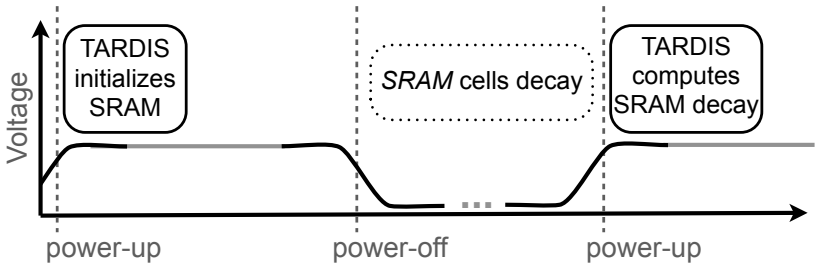


Figure 5.1. TARDIS estimates time by counting the number of SRAM cells that have a value of zero in power-up (*computes SRAM decay*). Initially, a portion of SRAM cells are set to one (*initializes SRAM*) and their values decay during power-off. The dots in the power-off indicate the arbitrary and unpredictable duration of power-off.

on the percentage of cells remaining 1. Algorithm 8 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 5.4), 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

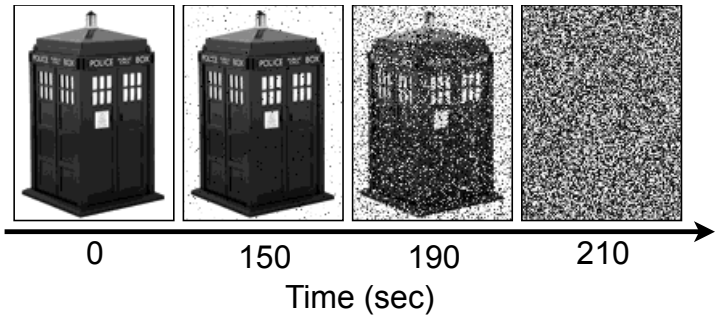


Figure 5.2. Programs without access to a trustworthy clock can determine time elapsed during a power failure by observing the contents of uninitialized SRAM. These bitmap images of the TARDIS [1] represent four separate trials of storing the bitmap in SRAM, creating an open circuit across the voltage 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.

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 5.4).

5.2.1 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 5.2 shows that energy 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[140] 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

Algorithm 8 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
     the history of temperature values stored in flash memory.
5     then return decay
6     else  return error
```

EXPIRED($addr, size$)

```
1  ≻ Checks whether SRAM decay has finished.
2  decay ← DECAY(addr,size)
3  if (decay ≥ %50 × 8 × size)
4     then return true
5     else  return false
```

TIME($addr, size, temperature$)

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

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.

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 5.2. Overhead of TARDIS INIT and DECAY procedures measured for TARDIS size of 256 bytes.

5.3 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 5.1 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 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 [140]. 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 [62]. 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 [62]. 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 [37]. 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 [11]. Another attack can accurately trace a specific passport by sending hundreds of queries per minute [33].

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 [10]. This delay increases to 14 seconds after 14 unsuccessful attempts (Figure 5.3) 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

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

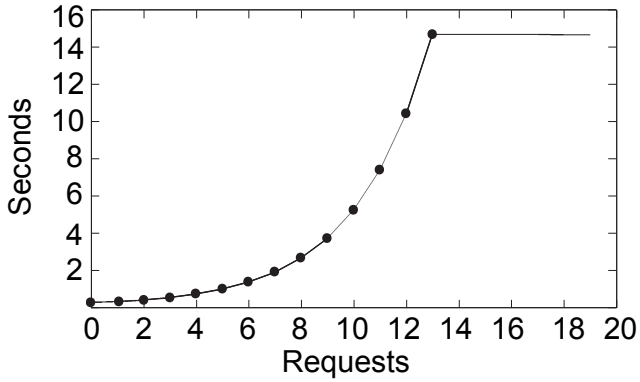


Figure 5.3. Measured response time of a 2010-issued French passport [10]. The passport imposes up to 14 seconds of delay on its responses after unsuccessful execution. The delay will remain until a correct reading happens even if the passport were removed from the reader's field for a long time.

these devices. Stajano et al. [119] 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 [56]. 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 [45]. The card could refuse to boot if it has been unpowered for a fair amount of time.

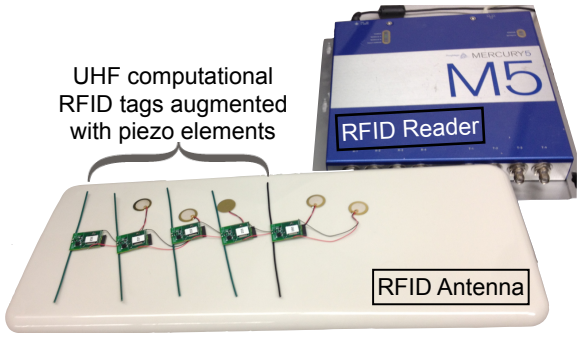


Figure 5.4. Our applications are implemented and tested on the Moo RFID sensors and are remotely powered by a RFID reader (ThingMagic M5 [129]).

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 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.

5.3.1 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 [54] 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 8. For the three implemented protocols,

Algorithm 9 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()
```

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 9. 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 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 [129] 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

[4]Our experiments (Section 5.4) have shown that SRAM decay finishes when the tag voltage reaches 50 mV.

chosen for our experiment. The resolution could potentially be extended to hours using additional capacitors (Table 5.4).

sectionSecurity 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 [53] 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 8, 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 [24] 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 5.7). 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 [135]. 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.

5.4 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 5.5), 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

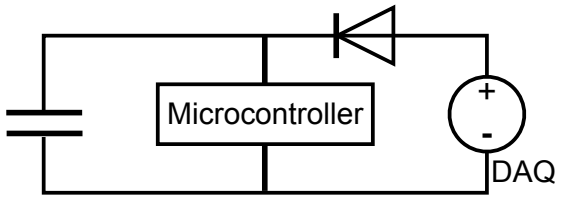


Figure 5.5. General circuit used during the experiments. The microcontroller is held in an environmental chamber to ensure consistent temperature during the tests. The Data Acquisition (DAQ) unit both provides power to the microcontroller and records the voltage decay.

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, 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 [122] 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 [89] 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 5.6 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

Term	Definition
SRAM Decay	Change of value in SRAM cells because of power outage
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 5.3. Definition of the terms used to explain the behavior of SRAM decay and the theory behind it.

the *off-time* from 0 to 400 seconds in 20-second increments. In the first stage, no memory cells have decayed; during the second stage, a fraction of the cells, but not all, have decayed; by the third stage the cells have decayed completely (see Table 5.3 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 5.5). 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 5.7) 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 [117] shows that low temperature can increase the remanence time of SRAM, and the work of Halderman et al. [53]

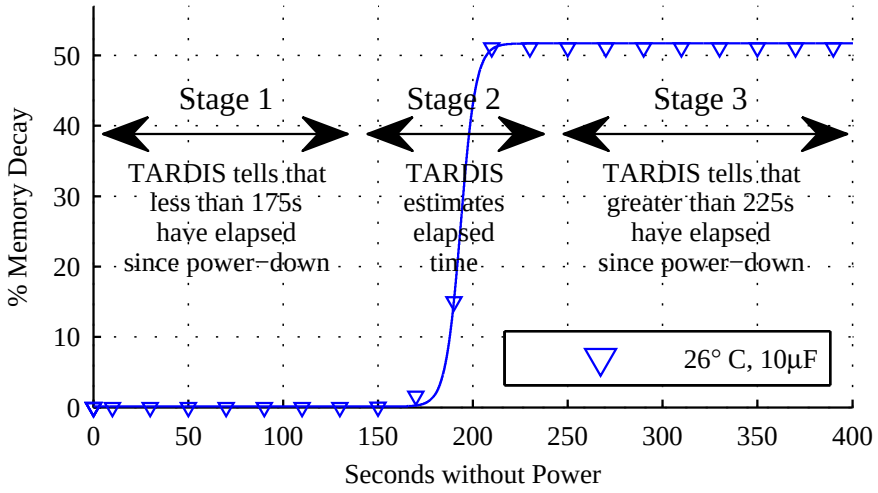


Figure 5.6. The TARDIS presents a three-stage response pattern according to its amount of decay. Before 175 seconds, the percentage of bits that retain their 1-value across a power-off is 100%. For times exceeding 225 seconds, the TARDIS memory has fully decayed. The decay of memory cells between these 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.

similarly shows that low temperature can extend the remanence time of DRAM. For the TARDIS using SRAM 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 5.8 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]

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

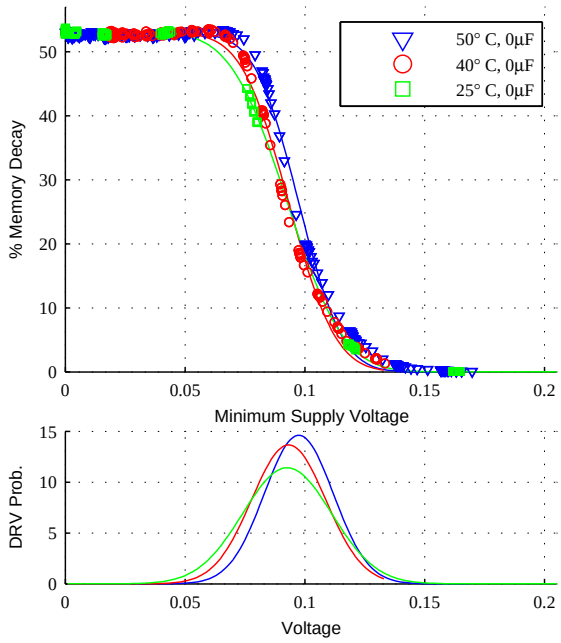


Figure 5.7. Regardless of temperature, the amount of decay depends almost entirely on the minimum supply voltage reached during a power-down. The bottom graph shows the 3-parameter DRV probabilities (Equation A.3) that best predict the observed relationships between decay and minimum supply voltage for 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 A).

Impact of Additional Capacitance: 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 5.4) to calculate the time frame achievable with each capacitor. Table 5.4 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 5.9.

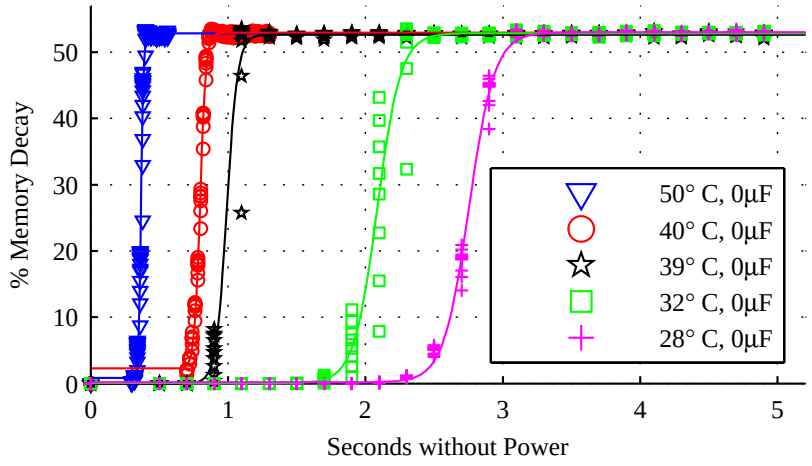


Figure 5.8. The duration of SRAM decay is non-zero across all temperatures even when no capacitor is used. For any given temperature, the duration of SRAM decay is consistent across trials. Increasing the temperature from $28°C$ to $50°C$ reduces the duration of both Stage 1 and Stage 2 decay by approximately 80%.

Results ranging from seconds to days open the path for a wide variety of applications 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 5.10. It should be noted that SRAM size is not the only difference between these three models, as they also have slightly different power consumptions.

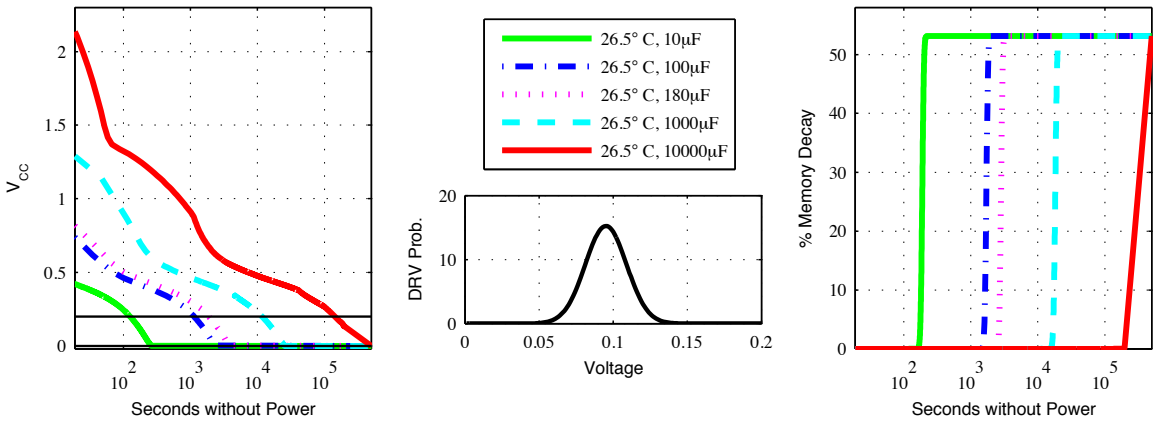


Figure 5.9. For five different capacitor values, measured supply voltage traces are combined with a pre-characterized DRV distribution to predict decay as a function of time. The decaying supply voltages after power is turned off are shown at left. The known DRV probabilities (Equation A.3) for $26.5°C$ are shown at center. Equation A.4 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.

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 5.11 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} through a D1N4148 diode, and the decoupling capacitor

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 5.4. 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.

is 70 nF. Each transistor is assigned a random threshold voltage deviation chosen uniformly from range ± 100 mV. Each line in Figure 5.12 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 5.12 as an increase in the magnitude of the differential voltage of each cell.

5.5 Inside an SRAM Cell

Each SRAM cell holds state using two cross-coupled inverters as shown in Figure 5.13; 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

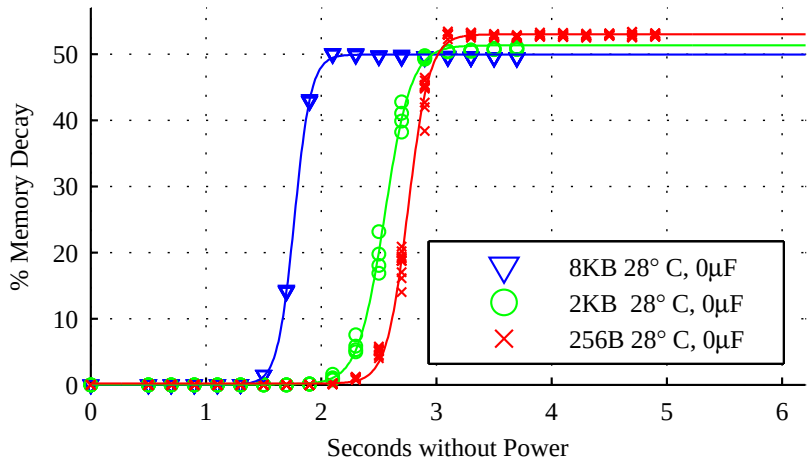


Figure 5.10. Different microcontrollers within the TI MSP430 family with different SRAM sizes exhibit different decay times, but follow the same general trend. The MSP430F2618, MSP430F169, and MSP430F2131 respectively have 8 KB, 2 KB, and 256 B of SRAM.

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) [98]. 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 5.7).

5.5.1 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

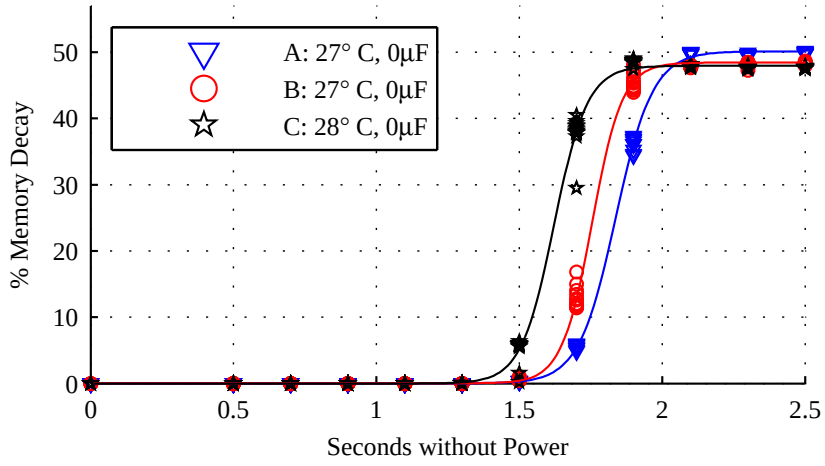


Figure 5.11. Decay versus time in 3 different instances of the MSP430F2131 microcontroller at similar temperatures. The durations of Stage 1 and Stage 2 decay match closely across instances.

that each cell typically favors one power-up state over the other [57, 48], 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 5.1, 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 paragraphs explain why both are present in our TARDIS application.

$$\frac{dv_{CC}}{dt} = \frac{I_{CC}}{C_{CC}} \tag{5.1}$$

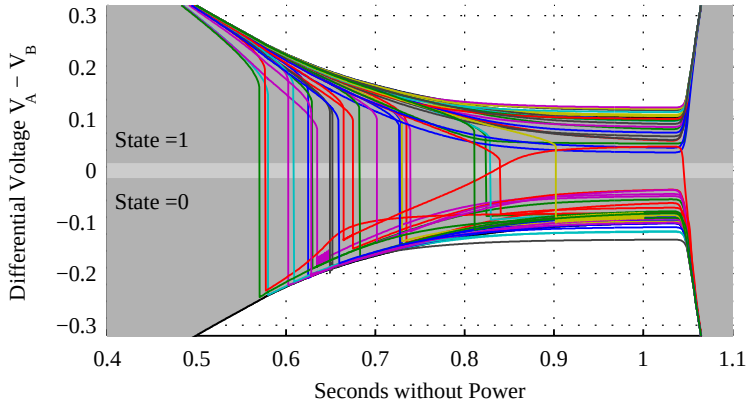


Figure 5.12. The differential voltage of SRAM cells during decay. The envelope of $\pm V_{CC}$ is shaded in grey. All cells are in the 1 state when power is first turned off. As V_{CC} decays, some cells flip from 1 to 0. The cells stabilize when power is restored. The number of zeros after the restoration of power is used to estimate the duration of the power outage.

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.

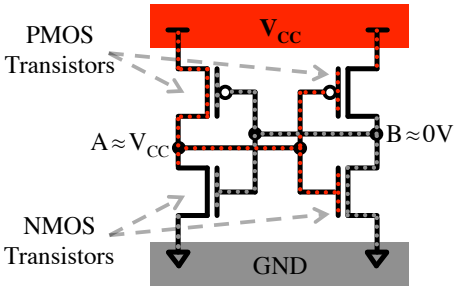


Figure 5.13. The state-holding portion of an SRAM cell consists of two cross-coupled inverters tied to the chip's power and ground nodes.

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 5.14, 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 [126]. 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 [126]. 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 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, causing the supply voltage to drop below DRVs sooner. Second, temperature expedites memory decay by increasing the DRV of SRAM cells [98],

[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 [117] by providing a low impedance leakage path to rapidly drain the capacitance; we have observed this same result in our experiments as well.

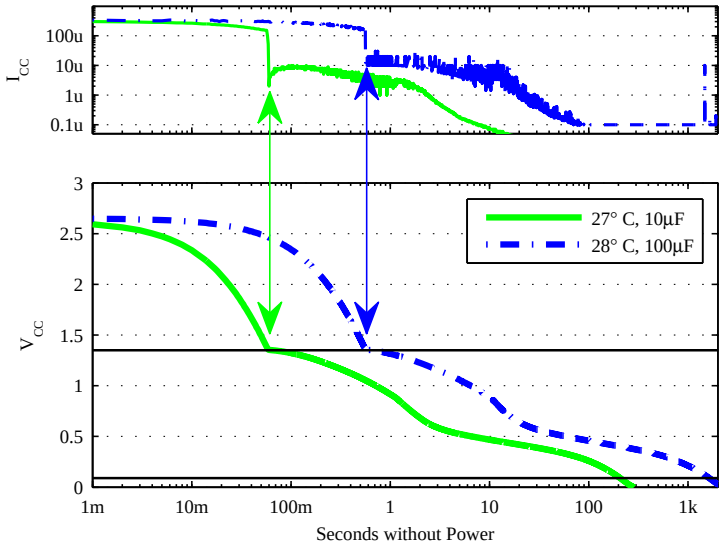


Figure 5.14. Supply voltage and current during two power-down events with different capacitors. The voltage V_{CC} is measured directly, and the current I_{CC} is calculated per Equation 5.1 using the measured $\frac{dV_{CC}}{dt}$ and known capacitor values. The voltage initially decays rapidly due to 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.

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 $100°C$ [98].

5.5.2 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 [48, 57], 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.

5.6 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 [37] 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.

5.7 Summary

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.

This page is intentionally left blank.

CHAPTER 6

FUTURE DIRECTIONS: SENSING AT LOW VOLTAGE

An Analog to Digital Converter (ADC) is one of the on-chip components that requires a relatively high supply voltage. For example, the Texas Instruments MSP430F1232 requires the user to provide the ADC with at least 2.2 V to guarantee functionality. This voltage requirement hinders the whole system from running at lower voltages if the power rail is shared among all components on board. Simple calculation suggests that operating a device at 2.2 V would require about 1.5 times more power than operating the device at 1.8 V.

We define error as the measurement difference of ADC readings at low and high voltage. Our preliminary results show that the ADC could work at voltages lower than specified by the chip manufacturers with introducing negligible errors (1.30%). We coin the term *nonsensor* to refer to any sensor used in a manner inconsistent with a manufacturer's specifications (e.g., voltages lower than expected), resulting in stochastic behavior and sometimes generating "nonsense" data.

6.1 Analog to Digital Converter (ADC)

To convert a continuous (analog) quantity to a discrete time presentation in digital form, Analog to Digital Converters (ADCs) are used. An ADC takes samples to convert, for example, an input analog voltage to a digital number proportional to the magnitude of the voltage. The resolution of an ADC is determined based on the number of distinct values it can output from an analog range. For example, the TI

MSP430F1232 has an ADC of 10-bit resolution and it can generate 1024 different levels.

An ADC usually requires higher voltage than the CPU, and if ADC and CPU share the same power rail, the whole chip has to operate at a higher voltage to satisfy the ADC requirement. Table 1.2 shows several examples of popular microcontrollers with ADCs that restrict choices for the CPU voltage supply.

To test the ADC performance at low voltages, we built a logger system both to automate the experiments and to minimize the effect of experimental setup on the results. We first introduce our logger systems and then provide the test results.

6.2 Logger System

For the nonsensors experiments, we needed to store the experimental results in a non-volatile storage. However, as explained before in Chapter 4, on-chip flash memory requires high voltage and does not work reliably at low voltages needed to test the nonsensors. We could use the Half-Wits method to log the experimental data in the on-chip flash memory. However, as the number of *in-place writes* would not be the same for different voltages, using Half-Wits might affect the experimental result (different voltage patterns would be applied and the temperature might change).

To log the experimental results reliably and without interfering with the experiment process, we built two logger systems: one that works with the UMass Moo [140] (Figure 6.1), and another one for a TI MSP430F1232 (Figure 6.2). In both of the logger systems, the test subject and the monitor are connected via three pins: clock signal, data line, and the ground. The logging platform is similar to that of the FrankenWISP [49] and the one used for Half-Wits (Section 4.2.1).

The test platform runs a test program at a desirable voltage. When the test program completes, the test platform sends the result of the experiment to the monitoring chip via GPIO pins. The test and monitoring platforms share 1+1 GPIO pins

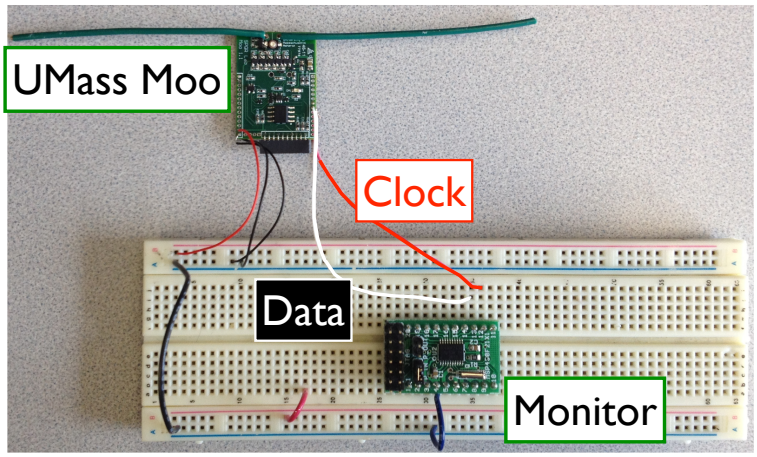


Figure 6.1. Logging system for voltage measurements.

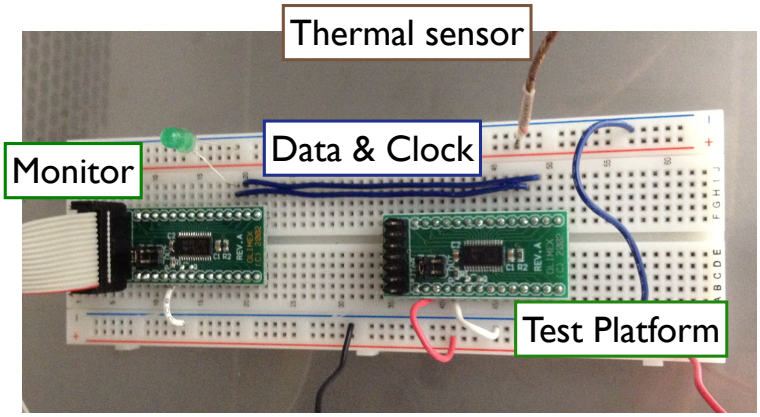


Figure 6.2. Logging system for testing temperature. The whole system is tested inside a thermal chamber.

to carry one bit of data and a clock signal. Once the test platform puts data on its data pin, it raises the clock pin. The monitoring chip reads data from its GPIO data pin whenever it detects a rising clock signal and logs the results in its own flash memory. The monitoring chip runs at a voltage above the nominal minimum for its own flash writes, thereby storing reliably.

In the case of testing the UMass Moo, we have used a TI MSP430F2131 as a monitoring platform. The unused GPIO pins 36 and 37 of the UMass Moo were connected to the P1.0 and P2.4 of the TI MSP430 microcontroller (Figure 6.1).

For testing temperature, we use a similar experimental setup but instead employ two TI MSP430F1232 MCUs (Figure 6.2). The two MSP430s (one used as the test platform, the other one used as the monitor) are connected through GPIO pins 36 and 37.

6.3 Preliminary Results: ADC Measurements at Low Voltage

We consider the impact of low voltage on two types of measurements made using ADC: measuring voltage on one of the capacitors of the UMass moo and measuring internal temperature of a microcontroller in a controlled environment. The effects on each of these measurements are evaluated by designing an experiment.

6.3.1 Voltage Sensor

Our hypothesis is that if the voltage being measured is lower than 1.8 V, the reliability of measurements will not be affected by operating the ADC at low voltage. Figure 6.3 shows how the operating voltage of a microcontroller can be measured at low voltages with a similar error percentage as at high voltages; this confirms our hypothesis. For this experiment, we define error to be the difference between the voltage measured by the ADC and the one measured by the multimeter. The mean of the error is 1.30% and 0.71% for low(1.8 V) and high(>2.4 V) voltages respectively.

In this experiment, we talk about three voltages: (1) MCU input voltage, which is either the output of the voltage regulator and fixed at 1.8 V or the output voltage of a JTAG going straight to the microcontroller, (2) supply voltage, which is the input voltage given directly to the system and it is usually supplied using a power supply, and (3) measured voltage, which is voltage measured across a specific pin. For example, it could measure the voltage of a specific capacitor in the system.

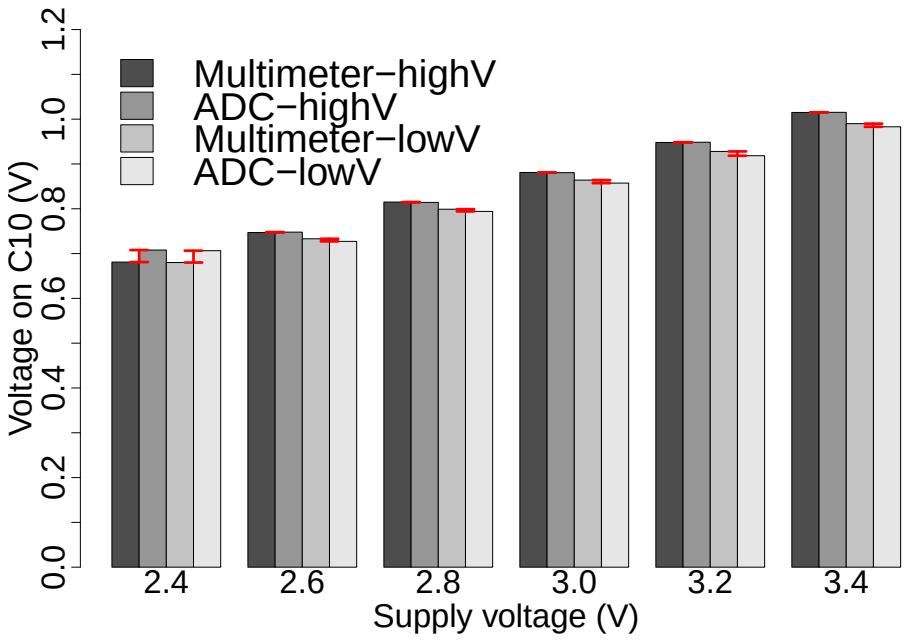


Figure 6.3. Voltage measurements at low and high supply voltages using ADC and a multimeter. The mean of the error (the difference between the voltage measured using the ADC and the multimeter) is 1.30% and 0.71% for low (1.8 V) and high (>2.4 V) voltages respectively.

Experiment: A UMass Moo uses its ADC (with 12-bit resolution) to measure the voltage of a pin named "VSENSE_ANALOG". This pin measures the voltage of capacitor C10 [140] and it is one third of the MCU input voltage. The experiment repeats the ADC measurements 500 times and they are all performed on the Moo with the MCU input voltages of low (1.8 V) and high ($> 2.4\ V$). The UMass Moo has a voltage regulator that converts the supply voltage of any value to 1.8 V, which is lower than the ADC requirement (2.2 V). When the Moo is connected to the JTAG debugger, the MCU voltage is the same as the supply voltage.

All the measurements performed in the first experiment are sampling a voltage lower than 1.8 V to verify our first hypothesis. To perform a complete test in a broader voltage range (for voltages greater than 1.8 V), we have repeated the above experiment and measured the supply voltage (instead of the voltage of C10 which is

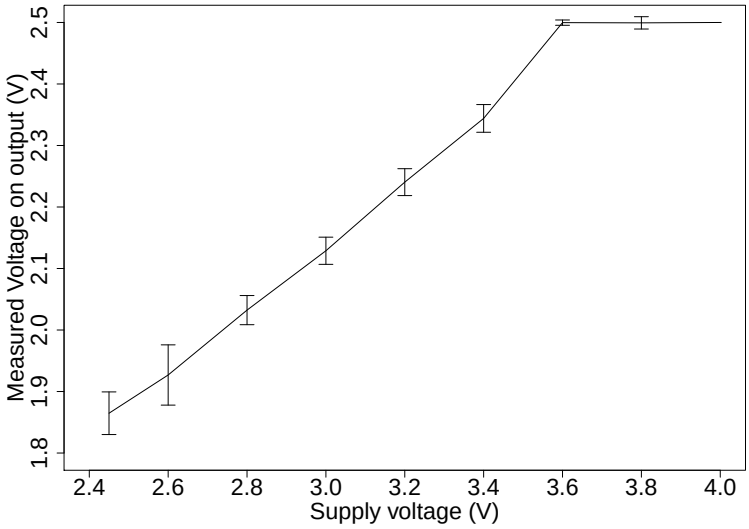


Figure 6.4. Voltage measurement of the Vout using the ADC operating at 1.8 V. The ADC cannot measure voltages greater than 2.5 V as the on-chip reference voltage generator cannot be set to anything higher than 2.5 V.

one third of the supply voltage). In this experiment, the voltage being measured goes as high as 3.2 V and all the measurements taken are in agreement with the multimeter and C10 measurements. However, the ADC reference voltage generator goes as high as 2.5 V and the ADC cannot measure anything higher than 2.5 V; hence the straight line after the point of 3.6 V on the supply voltage (Figure 6.4).

6.3.2 Internal Temperature Sensor

Most of the microcontrollers have an internal temperature sensor. If the microcontroller is not sleeping, the internal temperature will increase. This internal temperature cannot be used to read the ambient temperature and would need an external temperature sensor.

Our hypothesis is that the temperature measurements will be taken reliably at voltages as low as 1.8 V. This is because even temperatures as high as 80°C will

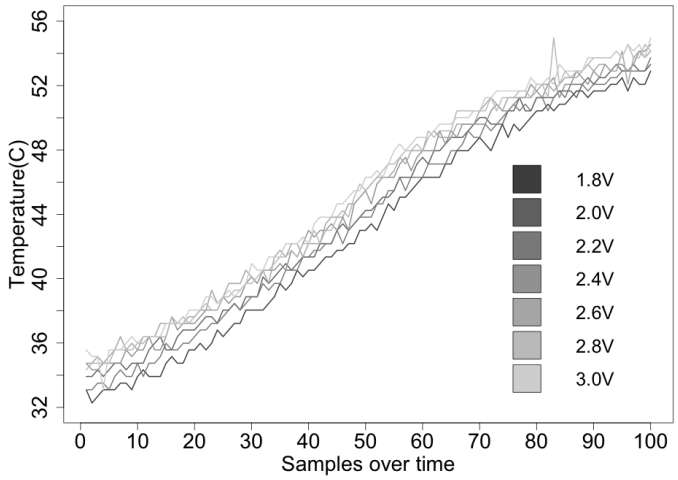


Figure 6.5. Measured temperature at different operating voltages. The temperature has been raised from 30°C to 60°C.

correspond to a voltage lower than 1.8 V. As Figure 6.5 shows, the temperature sensor functions at low voltages similar to that at high voltages.

Experiment: A TI MSP430F1232 microcontroller runs a program that uses the ADC and the internal temperature sensor to take temperature measurements. This program takes 100 temperature samples with about 5 ms of delay between two consecutive measurements. Once, the sample is ready, the microcontroller sends the sample to another microcontroller (monitor) to log the result. The ADC returns a 12-bit value which makes the test microcontroller send the monitor 2 bytes at a time. The experiment is performed in a thermal chamber that allows for controlling the temperature. The thermal chamber increases the temperature from 30°C to 60°C in about 2 minutes. The experiment is done at voltages in the range of 1.8 V—3.0 V.

One important note about this graph is that even though the surrounding temperature is changing from 30°C to 60°C, the internal temperature of the MCU does not change as quickly. Moreover, after each experiment, we waited for the environment to cool down and return to 30°C. However, the starting point is not exactly the same for all voltages as we could only measure the environmental temperature.

6.4 Applications and coordination with Half-Wits

Battery-powered and batteryless devices needing to minimize their energy consumption could operate at a lower voltage and still take reliable ADC measurements. All applications that can benefit from the Half-Wits, if they use ADC for any reason, can also benefit from the nonsensors. Moreover, sensor applications that transmit data in real time and do not store data in the on-chip flash memory would also benefit from nonsensors.

The nonsensors work complements the Half-Wits work, as it allows yet another on-chip component to operate at a lower voltage. For example, consider the case of a sensor mote or a smoke detector. Both of these examples take a sensor measurement and then store the result in on-chip non-volatile storage. Both of the components, ADC and the on-chip flash memory, require high voltage to operate; however, a combination of Half-Wits and nonsensors allows the device to operate at 1.8 V and reduce its energy consumption.

6.5 Future Directions

The preliminary results of nonsensors show that the on-board ADC of the TI MSP430 microcontrollers can be operated at low voltages without losing much precision. This allows existing MCUs to operate at low voltage and save energy without any hardware modifications. However, the extendibility of this work has to be confirmed for other types of ADCs that are assigned a separate power rail or external sensors sampled by ADCs.

Operating External ADCs and Sensors at Low Voltages: Even though in many existing MCUs, the CPU and ADC share the same power line, we would like to extend the nonsensors to ADCs with an independent power rail. This makes nonsensors extendable to future microcontrollers that allow each on-chip component have its own power rail.

Sensor	Min. Voltage Requirement	Typical Voltage Requirement
Sensirion Humidity Sensor SHT1X	2.4	3.0
Infineon Analog Absolute Pressure Sensor	4.5	5.0
TI Temperature Sensor LM20	2.4	2.4

Table 6.1. Sensors usually are required to operate at a high supply voltage.

Many sensors used in low-power devices require high voltages to operate reliably. Table 6.1 presents a few examples of sensors typically used in sensor nodes or other low-power devices. So far, we have only tested the ADC on the microcontroller. If the same results can be confirmed on other sensors, the energy consumption of low-power devices can be reduced.

Comparison of Nonsensors with Hardware Alternatives: There are numerous works on designing low-voltage ADCs that trade off low-voltage or low-power with precision. Mortezapour et al. [85] proposed a 1-V, 8-bit, 50-kS/s successive approximation ADC and more recently Chow et al. [34] proposed a similar ADC with about 10-bit precision. Chang et al. [29] designed a 1.4 V 10-bit 25-MS/s ADC. While the research on designing ADCs has proposed low-voltage and low-power alternatives, the current microcontrollers are still using ADCs with high voltage requirements. Nonsensors could fill the gap between existing and future devices and provide current systems with a software solution to reduce energy consumption.

Nonsensors under fluctuating voltage: While the operating voltage of most battery-powered devices is fixed in small time intervals, the batteryless devices suffer from fluctuating voltage due to changes in energy harvesting conditions. For the class of batteryless devices such as computational RFID tags, we need to evaluate the functionality of nonsensors. We could use a device called *Ekho* [141], designed and developed at UMass, to reproduce the same voltage trace of a Intel WISP [112]

or UMass Moo [140]. A series of experiments need be conducted to get samples from a nonsensor while the voltage is constantly changing. As the operating voltage of the system is low, we can use another microcontroller (running at a high voltage), to log the experimental results. Similar to the automated experimental setup in the Half-Wits project, the two microcontrollers would use data and clock signals.

Nonsensors on more platforms: Our preliminary experiments show that the ADC on the TI MSP430s can be used at lower voltages than recommended. However, we would like to examine the extendibility of nonsensors over different platforms. This step is necessary to show that our methods can cover most of the applications running on a diverse set of low-power products. Our previous study revealed that the most popular microcontrollers (apart from TI) in low-power products are from Microchip, ST, and ARM. Setting up the experiments for different microcontrollers requires learning about different IDEs, compilers, programmers, and debuggers. However, we will be reusing the same setup as for the Half-Wits and TARDIS projects. In all three projects, our goal is to show that the results are not platform-dependent.

6.6 Summary

The high voltage requirement of an on-chip ADC is a barrier to reducing the total energy consumption of low-power devices. This work examines the influence of operating voltage on the reliability of ADCs. Based on our observations on measuring voltage and temperature, the reliability of an ADC is not affected noticeably. Moreover, there are many applications that do not require a high precision in their sensor readings. The nonsensors would allow such applications to reduce their energy consumption by operating at a lower voltage.

CHAPTER 7

RELATED WORK

CCCP extends research on secure and energy-efficient storage for RFID tags. Section 7.1 discusses previous work on power-aware cryptography and storage for RFID tags. The Half-Wits work relates to storage for low-power systems, energy proportionality, and coding schemes (Section 7.2). TARDIS provides a secure notion of time for intermittently-powered devices by relying on SRAM decay properties. The last section of this chapter (Section 7.3) describes how other low-power or secure timers relate to TARDIS.

7.1 Secure Storage for Computational RFID Tags

Several other systems share design goals with CCCP. The following sections discuss systems that aim to either improve energy efficiency of batteryless RFID systems or to enhance security of storage.

Power-Aware Cryptography: Several systems share CCCP's goal of exploiting properties of RFID systems to enhance security and privacy. For instance, Shamir's SQUASH hash algorithm [115] exploits the underutilized radio link between a tag and a reader to reduce the amount of cryptographic computation necessary on a tag. While number-theoretic hash functions typically require significant computational resources for modular arithmetic, the SQUASH function eliminates costly modular reductions and produces large (unreduced) hash outputs that a tag can send directly to a reader. Tags can thus use the SQUASH function to engage in secure challenge-response protocols with minimal computational resources on the tag. The scheme

is provably as one-way as Rabin encryption. Like SQUASH, CCCP exploits the relatively low cost of radio communication between a tag and a reader to increase security. While SQUASH increases radio communication to reduce computation, CCCP increases radio communication to reduce writes to flash memory. CCCP shares some goals with power-aware encryption systems such as that proposed by Chandramouli et al. [28]. Both systems are designed to consume little energy while offering the security of well-known cryptographic primitives and both are motivated by a study of power profiling results, but they have different goals. Chandramouli et al. focus on deriving an energy consumption model and establishing a relationship between energy consumption and security, and they offer an encryption scheme that might allow CCCP to consume less energy during its precomputation of keystream bits. However, CCCP's opportunistic precomputation occurs during periods of abundant energy, when the choice of encryption scheme is not of the utmost importance. CCCP's precomputation allows it to use time- and energy-efficient XOR operations at checkpoint time, when energy is low; an alternative encryption scheme would have to save time or energy over simple XOR operations to be useful when energy consumption matters.

Secure File Systems: CCCP uses cryptographic techniques from past work on secure file systems and secure content distribution. CFS [17], the SFS read-only file system [40], and Plutus [64] investigated how to provide secure storage layered on various degrees of untrusted infrastructure. The key generation techniques in secure file systems help CCCP to precompute keystream materials during power seasons. While scalability and throughput are the main challenges in such file systems, CCCP primarily addresses energy and memory constraints. The semantics of CCCP storage are similar to the semantics of secure file systems. None of the systems explicitly and directly prevent denial of service. Storing information on untrusted RFID readers trades off the gain in storage capacity and energy conservation versus the risk of

losing data due to compromise or destruction of the external storage. To mitigate the risk against denial of service, CCCP could choose to replicate data as do secure file systems.

Batteryless RFID Systems: CCCP is closely related to Mementos [102] in that both systems provide checkpointing of program execution on CRFIDs. Whereas Mementos relies purely on flash memory and focuses on finding optimal checkpoint frequencies via static and dynamic analysis, CCCP relies primarily on untrusted remote storage via radio and focuses on low-power cryptographic protections to ensure that remotely stored data is as secure as if it were stored locally. CCCP provides secure storage for CRFIDs, and CRFIDs are closely related to existing passively powered RFID tags conforming to the EPC Gen 2 standard [37]. At times the RFID and sensor world fuse together. Buettner et al. [22] propose *RFID sensor networks* (RSNs) as a replacement for wireless sensor networks in applications where batteries are inconvenient, and the authors describe RSNs built on WISP CRFIDs. However, the RSN work does not consider remote storage options for CRFIDs.

Storage-Centric Sensor Networks: CCCP shares a number of properties with systems built for sensor networks. Storage-centric sensor networks [36, 78] have focused on reducing radio communication and increasing writes to flash memory to conserve energy. One of our motivating observations is that this relationship is inverted in the CRFID model: CCCP reduces writes to flash memory in favor of increasing radio communication. Performing cryptography is hard on both a CRFID and its elder cousin the sensor mote. Previous systems, such as SPINS [95] and TinySec [67] for sensor networks, have faced design choices similar to CCCP's. SPINS and TinySec use RC5 because of its small code size and efficiency, but the battery-powered platform underlying these systems differs in fundamental ways from batteryless computational RFIDs. For a side-by-side comparison of such embedded systems, see Table 1 of Chae et al. [26].

7.2 Non-Volatile Embedded Storage at Low Voltages

Half-Wits reduces the energy consumption of low-power systems with embedded flash memory by making energy proportional to the workload. This section starts with works related to energy proportionality. The rest of this section discusses work on storage for low-power devices and error correction schemes.

Energy proportionality: Our approaches share the philosophy that energy consumption should scale proportionally to utilization or error rates rather than proportional to a worst-case scenario. Blaauw et al. [15] reduce power consumption by lowering the operating voltage of a pipelined CPU. Certain pipeline stages may produce incorrect computation that require recomputation, but the errors can be made rare to allow better scalability of power consumption. Misailovic et al. [84] demonstrate that the programs whose loops performs fewer iterations cause tolerable errors while their execution time becomes shorter. Weddle et al. [134] introduce PARAID, a scheme that scales power based on the user demand while maintaining the reliability of the system. Their present work also tries to scale power based on the utilization of flash memory without losing storage reliability. Our approaches share this philosophy of scaling performance with utilization. Our performance metric is energy consumption, writes to flash memory represent our utilization, and energy-efficient error correction is our coping mechanism.

Storage for Low-Power Embedded Devices: Recent research focuses on optimizing use of off-chip flash memory. Off-chip memory allows for special features and larger memories than found on microcontrollers, but introduces additional costs for components. Microhash [137] is a memory index structure tailored for sensor devices with a large external flash memory. Mathur et al. [79] perform an extensive study of available flash memory candidates for sensor devices and demonstrate that an off-chip parallel NAND flash memory decreases the energy consumption of storage. Considering the off-chip NAND flash memory as the best candidate for sensor

devices, Agrawal et al. [5] propose a method that allows sensor devices to exploit their flash memory while adapting to different amount of RAM. However, our storage schemes are designed for already deployed low-power devices that use on-chip flash memory. Moreover, while devices at the scale of sensor nodes might switch to block-grained, large off-chip flash memory, RFID-scale platforms might not benefit from this transition because of their challenging resource limitations to drive I/O.

Error Correction Codes for Storage: Most previously published flash error correction codes [30, 41, 47] are designed for NAND flash memory. Chen et al. [31] mention that NOR flash normally does not require error correction. These techniques consider neither the asymmetry in low-voltage flash memory nor the resource limitations of low-power embedded devices. Many previous codes [12, 60, 139, 124] leverage the fact that each cell of MLC flash memory represents more than one bit of information. But the fact that single-level cells (SLC) are more suitable for embedded devices, in addition to the occurrence of errors in low-voltage conditions, requires a reconsideration of these codes for SLCs at low voltage. Zemor et al. [138] introduce error-correcting WOM codes for flash memory. They suggest codes that are able to correct up to one error when the flash memory is given enough voltage. This work does not account for errors that occur at low voltage. Godard et al. [44] propose hierarchical code correction and reliability management for NOR flash memory. This work considers on-chip ECCs such as Hamming and parity codes to correct the errors in NOR flash memory.

7.3 Time Keeping for Intermittently-Powered Devices:

TARDIS extends research on security of intermittently-powered devices by relying on SRAM decay properties. The following sections discuss security of RFID tags, secure and low-power timers, and memory remanence.

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 [62]; 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 [11]. The attack could be slowed down if the e-passport had a trustworthy notion of time. The minimalist model [61] 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 [108]; 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 [70]. This inability to acquire secure time precludes the use of many cryptographic protocols, including timed-release cryptography [77, 104].

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 [123]. Circuits in wristwatches combine analog components and small digital designs to operate at hundreds of nW [132]. A counter designed for smart cards uses adiabatic logic to operate at 14KHz while consuming 11nW of power [125]. A gate-leakage-based oscillator implements a temperature-invariant clock that operates at sub-Hz

frequencies while consuming 1pW at 300mV [73]. A TI-recommended technique [100] 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 [39] 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 [50]. 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 [8]. Experimental data demonstrating low-temperature data remanence on a variety of SRAMs is provided by Skorobogatov [117], 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 [131], as well as on DRAM [53]. 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 [113]. 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.

This page is intentionally left blank.

CHAPTER 8

CONCLUSION

The thesis of this work is that we can use unconventional and probabilistic computing that bends traditional abstractions and interfaces in order to reduce energy consumption while protecting program semantics. Designing methods that improve the energy efficiency of low-power devices faces a major challenge: protecting program semantics and preserving system performance while improving energy efficiency. The functionality, reliability, security, and other performance metrics of a system must not be sacrificed for the sake of reducing energy consumption.

In this thesis I propose four methods to reduce the energy consumption of low-power devices: 1) CCCP, which provides an energy-efficient storage alternative to local non-volatile storage by relying on cryptographic backscatter radio communication, 2) Half-Wits, which allows for operating the system at lower voltages in order to reduce energy consumption while it maintains storage reliability using software-only coding algorithms, 3) TARDIS, which exploits the decay properties of SRAM to estimate the duration of a power failure and provides a secure notion of time to systems without clocks, 4) Nonsensors, which allows for operating analog to digital converters at low voltages in order to reduce energy consumption of the whole system.

I have evaluated the energy savings of my proposed techniques through experiments on actual low-power devices. The results show that sometimes tightly maintaining the digital abstraction for embedded systems comes at a significant cost to energy consumption with minimal gain in reliability. Moreover, it is possible to use

the available resources opportunistically to provide less energy hungry alternatives to low-power systems.

APPENDIX A

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.

$$\Pr(g = 0) = \alpha \tag{A.1}$$

Among cells that are eligible for memory decay, we assume that DRVs are normally distributed with mean μ and standard deviation σ (Equation A.2).

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

The probability of a randomly selected cell being eligible for memory decay and having $DRV = v$ is given by Equation A.3. 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.

$$\Pr\left((g=0) \wedge (DRV=v)\right) = \frac{\alpha}{\sigma\sqrt{2\pi}} e^{-(v-\mu)^2/\left(2\sigma^2\right)} \tag{A.3}$$

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 A.4; this equation is 1 minus the CDF of a normally distributed random variable, scaled by α.

$$D_{PRED}(v_{min}, \alpha, \mu, \sigma) = \alpha \left(1 - \frac{1 + erf\left(\frac{v_{min}-\mu}{\sigma\sqrt{2}}\right)}{2}\right) \tag{A.4}$$

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 A.5. We initially use the set of measurements to find the model parameters that minimize the prediction error (see Figure 5.7).

$$ERR(\alpha, \mu, \sigma) = \sum_{i=0}^{n-1} \left(D_{PRED}\left(v_i, \alpha, \mu, \sigma\right) - D\left(v_i\right)\right)^2 \tag{A.5}$$

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 5.9).

APPENDIX B

BIOGRAPHY

Mastooreh (Negin) Salajegheh studied her Ph.D. at the Department of Computer Science of University of Massachusetts Amherst under the supervision of Dr. Kevin Fu. Her research interests are broadly in low-power embedded systems and trustworthy computing. She has worked on system security, RFID-scaled devices, probabilistic storage, near field communications, and security of medical infrastructures.

She was one of the top four finishers in the UMass Amherst Innovation Challenge Competition held on December 2011. Her team, named SMASH (for Smarter Flash), introduced techniques to reduce the power consumption of battery-powered devices through smarter flash memory.

Figure B.1. Negin

She spent summer 2011 at Microsoft Research working on peer-to-peer communication through Near Field Communications. In 2010, she won the best synthesis project award for her work on low-voltage non-volatile storage. Negin has published and presented papers in ACM Transactions in Embedded Computing Systems, USENIX Security, USNEIX FAST, Microsoft Research, Intel Research, and the Grace Hopper celebration of women in computing.

She received her M.S. degree in Information Networking from Carnegie Mellon University in 2006. Her M.Sc. thesis was on securing hierarchical sensor networks and under the supervision of Dr. Tassos Dimitriou. She received her B.S. degree in Computer Engineering from Sharif University of Technology in 2004.

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`, Dec. 2008.

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

[4] Agency, United States Environmental Protection. Common wastes and materials, batteries. http://www.epa.gov/wastes/conserve/materials/battery.htm.

[5] Agrawal, Devesh, Li, Boduo, Cao, Zhao, Ganesan, Deepak, Diao, Yanlei, and Shenoy, Prashant. Exploiting the interplay between memory and flash storage in embedded sensor devices. In *Proceedings of the 16th IEEE Conference on Embedded and Real-time Computing Systems (RTCSA)* (2010), pp. 227–236.

[6] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Computer Networks 38*, 4 (2002), 393–422.

[7] Alien Technology. Product Overview: Higgs-3 EPC Class 1 Gen 2 RFID Tag IC, July 2008.

[8] Anderson, R, and Kuhn, M. Tamper resistance: a cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce* (1996).

[9] Atmel AVR Solutions. ATmega128L. `http://www.atmel.com/atmel/acrobat/doc2467.pdf`.

[10] Avoine, Gildas. Personal communication on French passports. 2012.

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

[12] Barg, A., and Mazumdar, A. Codes in permutations and error correction for rank modulation. *IEEE Transactions on Information Theory 56*, 7 (2010), 3158–3165.

[13] Berger, J.M. A note on error detection codes for asymmetric channels. *Information and Control 4*, 1 (1961), 68–73.

[14] Berger, Stefan, Cáceres, Ramón, Goldman, Kenneth A., Perez, Ronald, Sailer, Reiner, and van Doorn, Leendert. vTPM: virtualizing the trusted platform module. In *Proceedings of the 15th USENIX Security Symposium* (2006), USENIX Association.

[15] Blaauw, D., and Das, S. CPU, heal thyself. *IEEE Spectrum 46*, 8 (2009), 40–56.

[16] Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P. UMAC: Fast and secure message authentication. In *CRYPTO* (1999), Springer-Verlag, pp. 216–233.

[17] Blaze, Matt. A cryptographic file system for UNIX. In *1st ACM Conference on Communications and Computing Security* (November 1993), pp. 9–16.

[18] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. PRESENT: An ultra-lightweight block cipher. In *Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems* (Berlin, Heidelberg, 2007), CHES '07, Springer-Verlag, pp. 450–466.

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

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

[21] Brassard, Gilles. On computationally secure authentication tags requiring short secret shared keys. In *CRYPTO* (1982), pp. 79–86.

[22] Buettner, Michael, Greenstein, Ben, Sample, Alanson, Smith, Joshua R., and Wetherall, David. Revisiting smart dust with RFID sensor networks. In *Proc. 7th ACM Workshop on Hot Topics in Networks (HotNets-VII)* (Oct. 2008).

[23] Buettner, Michael, Greenstein, Ben, and Wetherall, David. Dewdrop: An energy-aware task scheduler for computational RFID. In *Proc. 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI '11)* (Mar. 2011), USENIX Association.

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

[25] Carter, L., and Wegman, M. Universal hash functions. In *Journal of Computer and System Sciences* (1979), Elsevier, pp. 143–154.

[26] Chae, Hee-Jin, Salajegheh, Mastooreh, Yeager, Daniel J., Smith, Joshua R., and Fu, Kevin. *Wirelessly Powered Sensor Networks and Computational RFID*. Springer, ch. Maximalist Cryptography and Computation on the WISP UHF RFID Tag. To Appear.

[27] Chae, Hee-Jin, Yeager, Daniel J., Smith, Joshua R., and Fu, Kevin. Maximalist cryptography and computation on the WISP UHF RFID tag. In *Proceedings of the Conference on RFID Security* (July 2007).

[28] Chandramouli, R., Bapatla, S., Subbalakshmi, K. P., and Uma, R. N. Battery power-aware encryption. *ACM Trans. Inf. Syst. Secur. 9*, 2 (2006), 162–180.

[29] Chang, Dong-Young, and Moon, Un-Ku. A 1.4-V 10-bit 25-MS/s pipelined ADC using opamp-reset switching technique. *Solid-State Circuits, IEEE Journal of 38*, 8 (aug. 2003), 1401 – 1404.

[30] Chen, Bainan, Zhang, Xinmiao, and Wang, Zhongfeng. Error correction for multi-level NAND flash memory using Reed-Solomon codes. In *IEEE Workshop on Signal Processing Systems (SiPS 2008)* (Oct. 2008), pp. 94–99.

[31] Chen, Scott. What types of ECC should be used on flash memory? `http://www.spansion.com/Support/AppNotes/Types_of_ECC_Used_on_Flash_AN_01_e.pdf`, 2007.

[32] Cheng, Wayne H. Approaches and designs of dynamic voltage and frequency scaling. Master's thesis, University of California, Davis, CA, USA, Jan. 2008. `http://www.ece.ucdavis.edu/vcl/pubs/theses/2008-1`.

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

[34] Chow, Hwang-Cherng, and Chen, Yi-Hung. 1-V 10-bit successive approximation ADC for low power biomedical applications. In *18th European Conference on Circuit Theory and Design* (aug. 2007), pp. 196 –199.

[35] Cooper, D., Dang, H., Lee, P., MacGregor, W., and Mehta, K. *Secure Biometric Match-on-Card Feasibility Report*, 2007.

[36] Diao, Yanlei, Ganesan, Deepak, Mathur, Gaurav, and Shenoy, Prashant. Rethinking data management for storage-centric sensor networks. In *Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR)* (January 2007).

[37] EPCglobal. *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHZ–960 MHZ, Version 1.2.0*, 2008.

[38] Etzel, Mark, Patel, Sarvar, and Ramzan, Zulfikar. Square hash: Fast message authentication via optimized universal hash functions. In *In Proc. CRYPTO 99, Lecture Notes in Computer Science* (1999), Springer-Verlag, pp. 234–251.

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

[40] Fu, Kevin, Kaashoek, M. Frans, and Mazieres, David. Fast and secure distributed read-only file system. *ACM Transactions on Computer Systems 20*, 1 (Feb. 2002), 1–24.

[41] Fujino, M., and Moshnyaga, V.G. An efficient Hamming distance comparator for low-power applications. In *9th International Conference on Electronics, Circuits and Systems* (2002), vol. 2, pp. 641–644.

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

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

[44] Godard, Benoit, Daga, Jean-Michel, Torres, Lionel, and Sassatelli, Gilles. Hierarchical code correction and reliability management in embedded NOR flash memories. In *Proceedings of the 2008 13th European Test Symposium* (2008), pp. 84–90.

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

[46] Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. Ch., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation 101*, 23 (2000), e215–e220. Circulation Electronic Pages: `http://circ.ahajournals.org/cgi/content/full/101/23/e215`.

[47] Gregori, S., Cabrini, A., Khouri, O., and Torelli, G. On-chip error correcting techniques for new-generation flash memories. *Proceedings of the IEEE 91*, 4 (April 2003), 602–616.

[48] Guajardo, J, Kumar, S, Schrijen, GJ, and Tuyls, P. FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems (CHES)* (2007), pp. 86–80.

[49] Gummeson, Jeremy, Clark, Shane S., Fu, Kevin, and Ganesan, Deepak. On the limits of effective micro-energy harvesting on mobile CRFID sensors. In *Proceedings of 8th Annual ACM/USENIX International Conference on Mobile Systems, Applications, and Services* (June 2010), pp. 195–208.

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

[51] Gutterman, Zvi, Pinkas, Benny, and Reinman, Tzachy. Analysis of the Linux random number generator. In *IEEE Symposium on Security and Privacy* (2006), IEEE Computer Society, pp. 371–385.

[52] Haddad, Sameer, Chang, Chi, Swaminathan, Balaji, and Lien, Jih. Degradations due to hole trapping in flash memory cells. In *Electron Device Letters* (March 1989), IEEE, pp. 117–119.

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

[54] Halperin, Daniel, Heydt-Benjamin, Thomas S., Ransford, Benjamin, Clark, Shane S., Defend, Benessa, Morgan, Will, Fu, Kevin, Kohno, Tadayoshi, and Maisel, William H. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy* (May 2008), IEEE Computer Society, pp. 129–142.

[55] Heydt-Benjamin, Thomas S., Bailey, Dan V., Fu, Kevin, Juels, Ari, and O'Hare, Tom. 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* (February 2007), pp. 2–14.

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

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

[58] Homer. *Odyssey*, vol. XI. ca. 750 B.C.

[59] Jiang, Anxiao, Mateescu, R., Schwartz, M., and Bruck, J. Rank modulation for flash memories. In *IEEE International Symposium on Information Theory (ISIT)* (2008), pp. 1731–1735.

[60] Jiang, Anxiao, Schwartz, Moshe, and Bruck, Jehoshua. Correcting charge-constrained errors in the rank-modulation scheme. *IEEE Transactions on Information Theory 56*, 5 (2010), 2112–2120.

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

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

[63] Kahn, J. M., Katz, R. H., and Pister, K. S. J. Next century challenges: mobile networking for "Smart Dust". In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (1999), pp. 271–278.

[64] Kallahalla, Mahesh, Riedel, Erik, Swaminathan, Ram, Wang, Qian, and Fu, Kevin. Plutus: Scalable secure file sharing on untrusted storage. In *Proc. USENIX Conference on File and Storage Technologies* (San Francisco, CA, December 2003).

[65] Kansal, Aman, Hsu, Jason, Zahedi, Sadaf, and Srivastava, Mani B. Power management in energy harvesting sensor networks. *ACM Transaction Embedded Computing Systems 6* (September 2007).

[66] Kaps, Jens-Peter. *Cryptography for Ultra-Low Power Devices*. PhD thesis, Worcester Polytechnic Institute, 2006.

[67] Karlof, Chris, Sastry, Naveen, and Wagner, David. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)* (November 2004).

[68] Kidde, A UTC Fire & Security Company. Fire & Carbon Monoxide Safety Products (USA). `http://www.kidde.com/`.

[69] Klove, Torleiv. Error correcting codes for the asymmetric channel. Tech. rep., Informatics, University of Bergen, 1995.

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

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

[72] Lin, Kris, Hsu, Jason, Zahedi, Sadaf, Lee, David C, Friedman, Jonathan, Kansal, Aman, Raghunathan, Vijay, and Srivastava, Mani B. Heliomote: Enabling long-lived sensor networks through solar energy harvesting. In *Proceedings of ACM Sensys* (November 2005).

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

[74] Lo, Benny P. L., Thiemjarus, Surapa, King, Rachel, and Yang, Guang-Zhong. Body sensor network - a wireless sensor platform for pervasive healthcare monitoring. In *Adjunct Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE)* (2005), pp. 77–80.

[75] Mainwaring, Alan, Culler, David, Polastre, Joseph, Szewczyk, Robert, and Anderson, John. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications* (2002), pp. 88–97.

[76] Malan, David, Fulford-jones, Thaddeus, Welsh, Matt, and Moulton, Steve. CodeBlue: An ad hoc sensor network infrastructure for emergency medical care. In *International Workshop on Wearable and Implantable Body Sensor Networks* (2004).

[77] Mao, Wenbo. Timed-release cryptography. In *Selected Areas in Cryptography VIII* (2001), Prentice Hall, pp. 342–357.

[78] Mathur, Gaurav, Desnoyers, Peter, Ganesan, Deepak, and Shenoy, Prashant. CAPSULE: An energy-optimized object storage system for memory-constrained sensor devices. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys)* (November 2006).

[79] Mathur, Gaurav, Desnoyers, Peter, Ganesan, Deepak, and Shenoy, Prashant. Ultra-low power data storage for sensor networks. In *Proceedings of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (2006), pp. 374–381.

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

[81] Meninger, S., Mur-Miranda, J. O., Amirtharajah, R., Chandrakasan, A., and Lang, J. H. Vibration-to-electric energy conversion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 9*, 1 (February 2001), 64–76.

[82] Microchip. 32-bit PIC MCUs. `http://www.microchip.com/en_US/family/pic32/`.

[83] Microchip Technology Incorporated. Microchip eXtreme Low Power Microcontrollers. `http://www.microchip.com/`.

[84] Misailovic, Sasa, Sidiroglou, Stelios, Hoffmann, Henry, and Rinard, Martin. Quality of service profiling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE)* (2010), pp. 25–34.

[85] Mortezapour, S., and Lee, E.K.F. A 1-V, 8-bit successive approximation ADC in standard CMOS process. *Solid-State Circuits, IEEE Journal of 35*, 4 (april 2000), 642 –646.

[86] Nelson, Jacob, Sampson, Adrian, and Ceze, Luis. Dense Approximate Storage in Phase-Change Memory. In *ASPLOS* (2011).

[87] Nohl, Karsten, Evans, David, Starbug, Starbug, and Plötz, Henryk. Reverse-engineering a cryptographic RFID tag. In *Proceedings of the 17th conference on Security symposium* (2008), pp. 185–193.

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

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

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

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

[92] Papirla, Veera, and Chakrabarti, Chaitali. Energy-aware error control coding for flash memories. In *Proceedings of the 46th Annual Design Automation Conference (DAC)* (2009), ACM/EDAC/IEEE, pp. 658–663.

[93] Paradiso, Joseph A., and Starner, Thad. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive Computing 4*, 1 (Jan. 2005), 18–27.

[94] Pavan, P., Bez, R., Olivo, P., and Zanoni, E. Flash memory cells-an overview. *Proceedings of the IEEE 85*, 8 (Aug 1997), 1248–1271.

[95] Perrig, Adrian, Szewczyk, Robert, Wen, Victor, Culler, David, and Tygar, J. D. SPINS: Security protocols for sensor networks. *Wireless Networks 8*, 5 (Sept. 2002), 521–534.

[96] Pillai, Padmanabhan, and Shin, Kang G. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the eighteenth ACM symposium on Operating systems principles* (2001), SOSP '01, ACM, pp. 89–102.

[97] Polastre, Joseph, Szewczyk, Robert, and Culler, David. Telos: Enabling ultra-low power wireless research. In *Proc. 4th Int'l Symposium on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS '05)* (Apr. 2005), IEEE.

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

[99] Rahmati, Amir, Salajegheh, Mastooreh, Holcomb, Dan, Sorber, Jacob, Burleson, Wayne P., and Fu, Kevin. TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks. In *Proceedings of the 21st USENIX Security Symposium* (Bellevue, WA, Aug. 2012).

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

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

[102] Ransford, Benjamin, Sorber, Jacob, and Fu, Kevin. Mementos: System support for long-running computation on RFID-scale devices. In *Proceedings of the 16th Architectural Support for Programming Languages and Operating Systems (ASPLOS 2011)* (March 2011).

[103] Reed, I. S., and Solomon, G. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics 8*, 2 (1960), 300–304.

[104] Rivest, R. L., Shamir, A., and Wagner, D. A. Time-lock puzzles and timed-release crypto. Tech. rep., Cambridge, MA, USA, 1996.

[105] Rivest, Ronald L. The RC5 encryption algorithm. *Dr Dobb's Journal—Software Tools for the Professional Programmer 20*, 1 (1995), 146–149.

[106] Rivest, Ronald L. The RC5 encryption algorithm. In *Fast Software Encryption* (1995), Bart Preneel, Ed., Springer, pp. 86–96. (Proceedings Second Int'l Workshop, Dec. 1994, Leuven, Belgium).

[107] Rivest, Ronald L., and Shamir, Adi. How to reuse a write-once memory. *Information and Control 55* (1982), 1–19.

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

[109] Salajegheh, Mastooreh, Clark, Shane, Ransford, Benjamin, Fu, Kevin, and Juels, Ari. CCCP: Secure remote storage for computational RFIDs. In *Proceedings of the 18th USENIX Security Symposium* (Montreal, Canada, August 2009).

[110] Salajegheh, Mastooreh, Wang, Yue, Fu, Kevin, Jiang, Anxiao (Andrew), and Learned-Miller, Erik. Exploiting half-wits: Smarter storage for low-power devices. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST '11)* (San Jose, CA, Feb. 2011).

[111] Salajegheh, Mastooreh, Wang, Yue, Jiang, Anxiao (Andrew), Learned-Miller, Erik, and Fu, Kevin. Half-wits: Software techniques for low-voltage probabilistic storage on microcontrollers with NOR flash memory. *ACM Transactions on Embedded Computing Systems, Special Issue on Probabilistic Embedded Computing*. To Appear.

[112] Sample, Alanson P., Yeager, Daniel J., Powledge, Pauline S., Mamishev, Alexander V., and Smith, Joshua R. Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement 57*, 11 (Nov. 2008), 2608–2615.

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

[114] Schneier, Bruce. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[115] Shamir, Adi. SQUASH—a new MAC with provable security properties for highly constrained devices such as RFID tags. In *Proceedings of the 15th International Workshop on Fast Software Encryption (FSE)* (2008), Springer-Verlag, pp. 144–157.

[116] Shnayder, Victor, Chen, Bor-rong, Lorincz, Konrad, Jones, Thaddeus R. F. Fulford, and Welsh, Matt. Sensor networks for medical care. In *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)* (2005), pp. 314–314.

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

[118] Sorber, Jacob, Kostadinov, Alexander, Garber, Matthew, Brennan, Matthew, Corner, Mark D., and Berger, Emery D. Eon: A language and runtime system for perpetual systems. In *Proceedings of The Fifth Int'l ACM Conference on Embedded Networked Sensor Systems (SenSys '07)* (Nov. 2007), pp. 161–174.

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

[120] STMicroelectronics. 32-bit ST MCUs. `http://www.st.com/internet/mcu/product/164481.jsp`.

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

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

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

[124] Tamo, Itzhak, and Schwartz, Moshe. Correcting limited-magnitude errors in the rank-modulation scheme. *IEEE Transaction on Information Theory 56*, 6 (2010), 2551–2560.

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

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

[127] Texas Instruments Incorporated. http://www.ti.com/rfid/shtml/news-releases-11-12-07.shtml.

[128] Texas Instruments Incorporated. MSP430 Ultra-Low Power Microcontrollers. http://www.ti.com/msp430.

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

[130] Trusted computing group. http://www.trustedcomputinggroup.org.

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

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

[133] Webster, John G. *Design of Cardiac Pacemakers*. IEEE Press, 1995.

[134] Weddle, Charles, Oldham, Mathew, Qian, Jin, Wang, An-I Andy, Reiher, Peter, and Kuenning, Geoff. PARAID: A gear-shifting power-aware RAID. *ACM Transactions on Storage (TOS) 3*, 3 (2007), Article 13:1–33.

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

[136] Yeager, D.J., Powledge, P.S., Prasad, R., Wetherall, D., and Smith, J.R. Wirelessly-Charged UHF Tags for Sensor Data Collection. In *IEEE Int'l Conference on RFID* (2008), pp. 320–327.

[137] Zeinalipour-Yazti, Demetrios, Lin, Song, Kalogeraki, Vana, Gunopulos, Dimitrios, and Najjar, Walid A. Microhash: An efficient index structure for flash-based sensor devices. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies* (2005), pp. 31–44.

[138] Zemor, Gilles, and Cohen, Gerard D. Error-correcting WOM-codes. *IEEE Transactions on Information Theory 37*, 3 (May 1991), 730–734.

[139] Zhang, Fan, Pster, Henry D., and Jiang, Anxiao. LDPC codes for rank modulation in flash memories. In *Proc. IEEE International Symposium on Information Theory (ISIT)* (2010), pp. 859–863.

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

[141] Zhang, Hong, Salajegheh, Mastooreh, Fu, Kevin, and Sorber, Jacob. Ekho: Bridging the gap between simulation and reality in tiny energy-harvesting sensors. In *Workshop on Power Aware Computing and Systems* (October 2011).