Persistent Clocks for Batteryless Sensing Devices

JOSIAH HESTER and NICOLE TOBIAS, Clemson University
AMIR RAHMATI, University of Michigan
LANNY SITANAYAH, Clemson University
DANIEL HOLCOMB, University of Massachusetts Amherst
KEVIN FU, University of Michigan
WAYNE P. BURLESON, University of Massachusetts Amherst
JACOB SORBER, Clemson University

Sensing platforms are becoming batteryless to enable the vision of the Internet of Things, where trillions of devices collect data, interact with each other, and interact with people. However, these batteryless sensing platforms—that rely purely on energy harvesting—are rarely able to maintain a sense of time after a power failure. This makes working with sensor data that is time sensitive especially difficult. We propose two novel, zero-power timekeepers that use remanence decay to measure the time elapsed between power failures. Our approaches compute the elapsed time from the amount of decay of a capacitive device, either on-chip Static Random-Access Memory (SRAM) or a dedicated capacitor. This enables hourglass-like timers that give intermittently powered sensing devices a persistent sense of time. Our evaluation shows that applications using either timekeeper can keep time accurately through power failures as long as 45s with low overhead.

CCS Concepts: • **Computer systems organization** → **Embedded hardware**; *Reliability*; • **Security and privacy** → Embedded systems security;

Additional Key Words and Phrases: Remanence timekeepers, batteryless, CRFID, SRAM, clocks, RTC

ACM Reference Format:
Josiah Hester, Nicole Tobias, Amir Rahmati, Lanny Sitanayah, Daniel Holcomb, Kevin Fu, Wayne P. Burleson, and Jacob Sorber. 2016. Persistent clocks for batteryless sensing devices. ACM Trans. Embed. Comput. Syst. 15, 4, Article 77 (August 2016), 28 pages.
DOI: http://dx.doi.org/10.1145/2903140

1. INTRODUCTION

The future of sensing lies in supporting the vision of the Internet of Things (IoT), where trillions of heterogeneous devices collect, and share data across many contexts, and for many applications, from infrastructure monitoring to wearable computing. This vision echoes the original formulation of “smart dust” [Kahn et al. 1999], which advocated for millimeter scale sensors, deployed in a dense volume, for the benefit of our everyday

This research is supported by NSF grants CNS-0831244, CNS-0845874, CNS-0923313, CNS-0964641, ACI-1212680, SRC task 1836.074, Gigascale Systems Research Center, and a Sloan Research Fellowship. Any opinions, findings, conclusions, and recommendations expressed in these materials are those of the authors and do not necessarily reflect the views of the sponsors. Portions of this work are patent pending.
Authors' addresses: J. Hester, N. Tobias, and J. Sorber, School of Computing, Clemson University, 100 McAdams Hall, 821 McMillan Rd., Clemson SC, 29670; emails: {jhester, rtobias, jsorber}@clemson.edu; A. Rahmati and K. Fu, Department of Electrical Engineering & Computer Science, University of Michigan, Beyster Building, 2260 Hayward Street, Ann Arbor, MI 48109-2121; emails: {rahmati, kevinfu}@umich.edu; D. Holcomb and W. P. Burleson, Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Knowles Engineering Bldg, 151 Holdsworth Way, Amherst MA 01003-9284; emails: {holcomb, burleson}@engin.umass.edu.

lives. These sensors must harvest energy, to achieve the long lifetimes necessary (often measured in decades), and to reduce the size and cost of a larger energy store. Currently, many sensors combine energy harvesting with a battery or supercapacitor, however, the inclusion of a battery, and in many cases a supercapacitor, is fundamentally at odds with the IoT vision.

Batteries increase device size, weight, and cost, and age significantly over only a few years. Most importantly, the recycling and disposal of large amounts of batteries present serious environmental concerns. These challenges have inspired a range of smaller, cheaper, computing devices without batteries that can be deployed maintenance-free for decades, powered by the abundant supply of environmental energy, all stored in tiny capacitors that are much easier to recycle, and cheaper to produce. This new generation of batteryless sensing devices can be used for passive RF-powered interactions (reimagining contactless smart cards, and RFID tags), active data gathering to monitor the health of a structure, or to better understand the behaviors of wildlife populations. The combination of energy harvesting and batteryless computing promises to enable a wide range of new applications in the future IoT.

However, keeping time on these untethered batteryless devices can be challenging. Harvested energy is often variable and difficult to predict, and many devices can store only enough energy in a tiny capacitor for a few seconds of operation [NXP Semiconductors MIFARE Classic 2012; Sample et al. 2008; Buettner et al. 2008]. Volatile power supplies and tight energy budgets often lead to short bursts of operation punctuated by frequent power failures—events that cause traditional timekeeping options (e.g., the local system clock or a Real-Time-Clock (RTC) [Rousseau 2001]) to stop working, leaving the device with a severely limited ability to reason about the passing of time outside of the current burst of operation.

The inability to keep time is a critical limitation for many mobile applications. Analyzing sensor readings to determine how a signal is changing is often impossible without knowing how much time passed between samples. Upon detecting an important event, application designers and end users often want to know when the event occurred. Security protocols rely on a trustworthy time source to determine whether a request was valid or malicious. Existing systems may leverage an external device (e.g., RFID tag reader) to act as a time source, which increases an application's infrastructure costs, limits device range and mobility, and introduces security vulnerabilities. In spite of these work-arounds, batteryless computing devices have, to date, been unsuitable for many applications that must reason about time.

This article presents two alternative batteryless techniques for keeping time on intermittently powered batteryless devices using remanence decay: TARDIS and CusTARD. Time and Remanence Decay in SRAM (TARDIS) is a software-only technique that tallies the percentage of decayed Static Random-Access Memory (SRAM) cells in an SRAM array to estimate the duration of a power failure. Experimental results indicate that a TARDIS can reliably estimate the duration of power failures from a few seconds up to several hours. However, TARDIS only provides coarse-grained timing information not suitable for data-driven applications requiring accurate timestamps. TARDIS was originally presented at the 21st USENIX Security Symposium [Rahmati et al. 2012].

The second approach, Custom Time And Remanence Decay (CusTARD), provides finer grained timing at the cost of a millimeter scale hardware addition centered around a capacitor. When a device equipped with CusTARD turns on, it charges the capacitor to a specific voltage. When the device turns off, energy within the capacitor slowly begins to dissipate. During the next power-up cycle, CusTARD measures the capacitor voltage and estimates the time as a function of how much the voltage decayed. Even though this method requires a small amount of additional hardware to function, it allows better measurement resolution, is less invasive to sensor software, and is more configurable than TARDIS.

1.1. Contributions

The tiny sensing systems that comprise a key part of the future IoT will be batteryless. This work seeks to enable the sensing devices of the IoT by providing methods to help with a core problem of current batteryless sensing; that of timekeeping. Towards that goal, our contributions include the following:

(1) The algorithmic building blocks for computing elapsed time from capacitive decay.
(2) Characterization of remanence decay under different temperatures, capacitances, SRAM sizes, and circuit instances.
(3) Software-only proof-of-concept implementation named TARDIS that uses on-chip SRAM to keep time.
(4) A tiny hardware prototype named CusTARD that uses dedicated capacitors to keep time.
(5) Evaluation of both techniques, including an evaluation of common use cases.

2. BATTERYLESS SENSING: BACKGROUND, OBSERVATIONS, AND CHALLENGES

Sensing platforms, from the very early days, have been powered by batteries. However, this must change, to enable new applications that have come from the IoT. Batteries are already a detriment to sensing devices, because of physical factors such as size, weight, and cost, the human cost of maintenance, and environmental impact. Worse, improvements in battery technology have historically come slowly, never keeping pace with the rapid advances occurring in the sensing community. The problems of batteries will only compound with the scale of the IoT, as connected devices go from millions, to billions, to trillions. Future ubiquitous sensing applications cannot afford these costs. Next, we describe the three main factors that prevent battery usage in modern and future sensing applications in the IoT:

Size, Weight, Cost: Batteries are expensive, often one of the most expensive components of a sensor, especially when optimizing for energy density. Batteries are also heavy, and large, often taking up the most space on a sensing device. These factors limit the applications and deployed sensor density. The cost of individual devices must be extremely low, and the size footprint small, to make the IoT vision feasible.

Wear and Maintenance: Batteries wear out quickly in wireless sensor networks; even when carefully managed. All batteries will eventually age and die, no matter the control circuitry. Often the battery is severely damaged by overcharging and overdischarging, reducing the lifetime even further [Buchmann 2001]. Replacement and maintenance, particularly for a large number of sensing devices, becomes prohibitively costly as this maintenance is generally done by humans. It will be not possible to change the batteries of the trillions of tiny sensors that will make up the IoT.

Environmental Harm: The most important issues with batteries powering the IoT are the environmental concerns associated with processing and disposing of large numbers of dead batteries [Larcher and Tarascon 2015]. Fundamental and technological obstacles must be overcome before large scale deployment of batteries is environmentally sustainable. This problem is already visible in the area of consumer electronics recycling; the weight of dead, rechargeable lithium batteries in China is expected to surpass 500 thousand metric tons by 2020 [Zeng et al. 2012]. Additionally, dead batteries leech harmful chemicals into the soil, including chromium, lead, lithium, and thallium [Kang et al. 2013]. Before the IoT can truly succeed, the environmental burden of recycling trillions of batteries must be addressed, or subverted.

2.1. Batteryless Devices and Applications

The physical factors, maintenance costs, and environmental harm of batteries, have caused the creation of a new class of sensing device that is batteryless, operates purely on harvested energy, and computes intermittently. Motivated by new mobile applications with strict size and cost constraints, lifetime requirements measured in decades, as well as recent advances in low-power microcontrollers, developers of sensor platforms have decided to leave their batteries behind and attempt to sense on transient power. Batteryless devices usually have five major components: (1) an energy harvester (solar, kinetic, RF, microbial, and others), (2) a small capacitor for energy storage, (3) a microcontroller, (4) a communication channel (radio or backscatter), and (5) sensors and actuators.

Batteryless devices are perhaps most visible in the application realm that RFID occupies. Contactless smart cards and Computational RFID tags (CRFIDs) [Ransford et al. 2008; Sample et al. 2008; Zhang et al. 2011; Yeager et al. 2010]—typically have limited computational power, rely on wireless transmissions from a reader both for energy and for timing information, and lose power frequently due to minimal energy storage. These batteryless devices enable new sensing applications in wearable technology, especially mHealth [Ranasinghe et al. 2012], water monitoring through microbial energy harvesting [Donovan et al. 2008], environmental monitoring [Minami et al. 2005], greenhouse monitoring [Hester et al. 2015], and many other applications. Each of the applications described were not possible with conventional sensing techniques; either because of extended lifetime requirements, deployment and maintenance impossibilities, or just economics. These applications change the way we look at sensing; however, each of these applications suffer from the same problem—that of reliable timekeeping.

When the available energy of batteryless devices runs out, and the capacitor depletes, the microcontroller, volatile RAM, and all clocks are reset. All of the sensor platforms previous timestamps (and therefore sensor data) have no meaning, since the local clock will start back at zero. For example, when a contactless transit card is brought sufficiently close to a reader in a subway, the card gets enough energy to perform the requested tasks. As soon as the card is out of the reader range, it loses power and is unable to operate until presented to another reader. Since a tag loses power in the absence of a reader, it does not have any estimation of the time between two interactions with a reader. Partly because of the trouble with timekeeping, intermittent computing has not been applied to complex or critical sensing problems.

2.2. Timekeeping Methods

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. Current embedded systems address the timekeeping issue in one of the following ways:

RTC: A system can power a RTC. This is not practical on intermittently powered devices due to tight energy budgets and inconsistent power. Even a low-power RTC (e.g., NXP PCF2123 [NXP Semiconductors SPI Real time clock/calendar 2012] or Abracon AB08X5 [Abracon Real time clock 2015] RTC chip) increases device size by using a battery or supercapacitor. Additionally, these devices often trade off a long startup time (up to 1.5s for the AB08X5) for low power operation, making them difficult to use with intermittent systems that restart multiple times a second. While in theory these devices could last decades on a small coin cell, in practice this is rarely the case. Constant recharging in deployment, hostile environmental conditions, poor charging circuitry design, and mismatched loads have reduced the lifetime of batteries below expectations in many sensor deployments [Szewczyk et al. 2004; Dehwah et al. 2015].

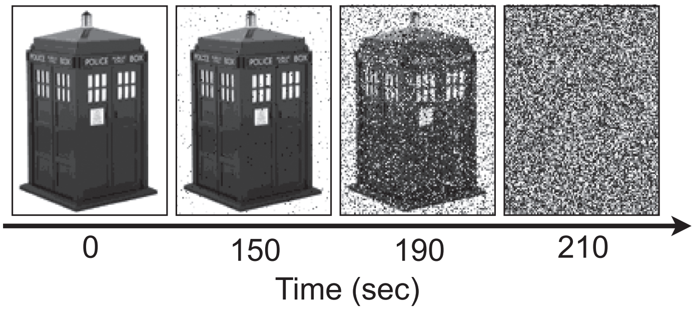


Fig. 1. Programs without access to a trustworthy clock can determine time elapsed during a power failure by observing the contents of uninitialized SRAM. These bitmap images of another TARDIS [Newman et al. 1963] represent four separate trials of storing the bitmap in SRAM, creating an open circuit across the supply voltage for the specified time at 26°C, then immediately returning a normal supply voltage and reading uninitialized SRAM upon reboot. No TARDIS was harmed or dematerialized in this experiment.

External Time: A system can keep time by accessing an external device (e.g., an RFID tag reader) or by secure time synchronization [Ganeriwal et al. 2005; Sun et al. 2006]. An external timekeeper can work well for some types of applications that are not time sensitive, or are deployed in easily accessible locations. However, relying on an external timekeeper introduces security concerns and may require significant infrastructure or severely limit range and mobility.

Approximation: A system can approximate or guess time by checkpointing state and integrating known constraints from the duty cycle or environment. However, this provides unreliable or inaccurate time and will not work for applications where constraints are fluid and the environment is unpredictable. Moreover, approximations are application dependent, and heavily rely on contextual information known beforehand by the developer.

None of the timekeeping techniques described can provide precise timekeeping that is applicable to general applications. In contrast to the RTC method, keeping external time, or approximation techniques, remanence-based timekeeping moves batteryless sensing toward a general solution to the problem of time in intermittently powered systems.

3. REMANENCE-BASED TIMEKEEPING

When devices turn off, the supply voltage does not immediately fall to 0V. Charge remains in the decoupling capacitance that designers add between V_{CC} and *gnd*, maintaining the supply voltage long after the device is off. During normal operation, this decoupling capacitance serves to stabilize the supply voltage to the functional blocks of the chip, including SRAM. This charge eventually decays to nothing the longer a device is without power. An example of this is shown in Figure 1 where the effect of time on SRAM decay in the absence of power is visualized. In this experiment, a 100×135 pixel bitmap image of a certain TARDIS [Newman et al. 1963] was written into the SRAM of a TI MSP430 microcontroller. The contents of the memory were read 150, 190, and 210s after the power was disconnected. The degree of image distortion is a function of the duration of power failure.[1]

[1]The 14.6KB 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(a) with a 10μF capacitor at 26°C.

Table I. Comparison of Remanence Timekeepers

	TARDIS	CusTARD
Storage capacitor	Supply node	Dedicated capacitor on I/O
Measurement mechanism	% of SRAM cells decayed	ADC to measure capacitor voltage
Measurement precision	Low	High
Discharge current	Current draw of entire chip	Leakage current through microcontroller I/O pin

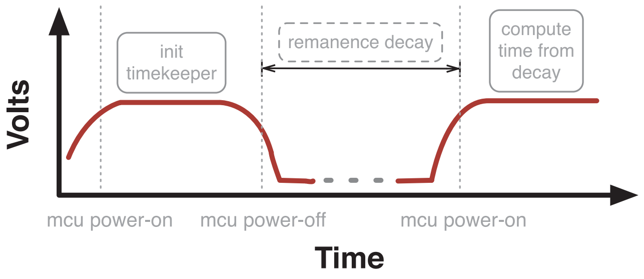


Fig. 2. Remanence timekeepers estimate time from the amount of remanence decay. TARDIS estimates time by counting the number of SRAM cells that have a value of zero in power-up (*compute time from decay*). Initially, a portion of SRAM cells are set to 1 (*init timekeeper*) and their values decay during a power failure. CusTARD measures how much the voltage on the CusTARD capacitor has decayed, using an on-board ADC (*compute time from decay*). The lower the voltage, the longer the microcontroller has been off. Initially, the CusTARD capacitor is charged to a specific voltage (*init timekeeper*). This figure shows the voltage charging the MCU.

The supply voltage decays according to Equation (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.

$$\frac{dV_{CC}}{dt} = \frac{I_{CC}}{C_{CC}}. \tag{1}$$

By measuring the voltage decay, and correlating that decay to known amounts of time, we can make timekeepers that work even when the microcontroller is off. We have developed two approaches exploiting the remanence decay of SRAM and capacitors. We name the first approach TARDIS. TARDIS measures the amount of SRAM decay after returning from a power failure to estimate time elapsed since the power failure. The TARDIS approach is described in Section 3.1. TARDIS was originally presented at the 21st USENIX Security Symposium [Rahmati et al. 2012]. The second technique, named CusTARD measures the voltage decay of a dedicated capacitor in a tiny hardware addition to estimate time. We describe CusTARD in Section 3.2. Table I compares how the two approaches make use of Equation (1).

3.1. TARDIS

We name the first approach to exploit remanence decay for timekeeping TARDIS. TARDIS exploits SRAM decay during a power failure to estimate time. Figure 2 shows the general mechanism of TARDIS. When a device is powered up, TARDIS initializes a region of 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 device is powered up, TARDIS estimates the time elapsed after the power loss based on the percentage of cells remaining 1.

Memory decay occurs in SRAM when a cell loses its state during a power cycle and subsequently initializes to the opposite state upon restoration of power. Given that each cell typically favors one power-up state over the other [Holcomb et al. 2009; Guajardo et al. 2007], memory decay can be observed only when the last-written state opposes

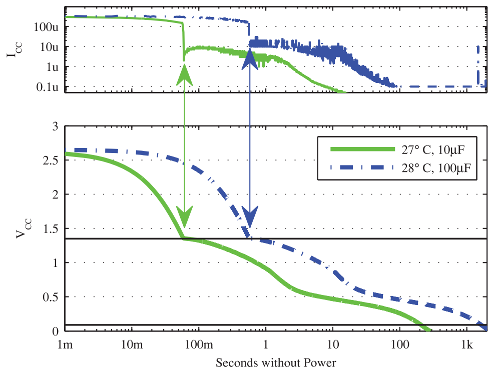


Fig. 3. Supply voltage and current during two power-down events with different capacitors. The voltage V_{CC} is measured directly, and the current I_{CC} is calculated per Equation (1) using the measured $\frac{dV_{CC}}{dt}$ and known capacitor values. The voltage initially decays rapidly due to the high current draw of the microcontroller. When V_{CC} reaches 1.4V, 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.

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 Data Retention Voltage (DRV) [Qin et al. 2004]. 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 50–250mV. This estimate is consistent with published work that measures the DRVs of instances of the same MSP430 microcontroller model used in this work [Holcomb et al. 2012]. Our own analysis in this work estimates a majority of DRVs to be in the range of 50–160mV (Figure 7(a)). 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.

Figure 3 shows current and voltage decay of an SRAM equipped MSP430F2131 [Texas Instruments Inc. 2011]. Immediately after power is disconnected, supply voltages are above 1.4V and the microcontroller is operational. The observed current is between 250μA and 350μA. 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 1.4V, 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 0.1μA for the 256B of SRAM in the MSP430F2131 [Texas Instruments Inc. 2011]. The current further decreases as the supply voltage drops into subthreshold, and cells begin to experience memory decay.[2]

Algorithm 1 gives more details about the implementation of TARDIS.

MEASURE_TEMPERATURE: To detect and compensate for temperature changes that could affect the decay rate (Section 5), TARDIS uses an on-board temperature sensor as is found on most microcontrollers. The procedure MEASURE_TEMPERATURE stores

[2]Note that setting V_{CC} to 0V during the power-down, instead of leaving it floating, reduces voltage and memory decay times by at least an order of magnitude [Skorobogatov 2002] by providing a low impedance leakage path to rapidly drain the capacitance; we have observed this same result in our experiments as well.

ALGORITHM 1: TARDIS Implementation

```
INIT($addr$, $size$)
1  for $i \leftarrow 1$ to $size$
2      do $memory(addr + i - 1) \leftarrow 0xFF$
3  $temperature \leftarrow$ MEASURE_TEMPERATURE()

DECAY($addr$, $size$)
1  $decay \leftarrow$ COUNT0S($addr$, $size$)
2  ≻ Proc. COUNT0S counts the number of 0s in an area of memory.
3  if TEMPERATURE_ANALYZE($temperature$)
4  ≻ This procedure decides if the temperature change derived in INIT is normal
5     then return $decay$
6     else return error

EXPIRED($addr$, $size$)
1  ≻ Checks whether SRAM decay has finished.
2  $decay \leftarrow$ DECAY($addr$,$size$)
3  if ($decay \geq 40\% \times 8 \times size$)
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  $decay \leftarrow$ DECAY($addr$,$size$)$/(8 \times size)$
3  $time \leftarrow$ ESTIMATE($decay$,$temperature$)
4  return $\{time, decay\}$
```

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 *time* returned can be derived from *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 ($0 \leq decay \leq 40\%$), the return value *time* is an estimate of the elapsed time based on *decay*. If the SRAM decay has finished ($decay \simeq 40\%$), the return result is $\{time, 40\}$ meaning that the time elapsed is greater than *time*.

EXPIRED: The TARDIS EXPIRED procedure returns whether the block of SRAM has completely decayed or not.

ESTIMATE: The procedure ESTIMATE uses a lookup table filled with entries of decay, temperature, and time stored in nonvolatile memory. This table is computed based on a set of experiments on SRAM in different temperatures. Once the time is looked up based on the measured decay and the current temperature, the result is returned as *time* by the ESTIMATE procedure. The precompiled 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).

It should be noted that TARDIS does not require identifying eligible SRAM cells. The TARDIS algorithm simply reads all selected cells, but extracts no information from ineligible cells. Given that roughly half of cells are eligible, and the eligible cells are randomly located, any nontrivial amount of memory will contain many eligible cells, allowing for scalable TARDIS sizes.

3.2. CusTARD

The second approach to exploit remanence decay for timekeeping is called CusTARD. CusTARD is a small hardware addition centered around a ceramic capacitor. Figure 2 shows how CusTARD works. When the microcontroller powers on, the capacitor is charged to a reference voltage generated by the microcontroller through a General Purpose Input/Output (GPIO) pin. It should be noted that this GPIO is dedicated to charging, and once done, is set to a high impedance input pin, meaning that no load effects occur on the microcontroller. When the microcontroller eventually loses power, the CusTARD capacitor's voltage slowly decays. Once the microcontroller regains power, it uses an on-board ADC (Analog-to-Digital Converter) to read the voltage on the CusTARD capacitor. This voltage is put in a lookup table and adjusted for temperature to determine the time elapsed since the microcontroller lost power. By decoupling the CusTARD capacitor from the microcontroller, the supply voltage of the CusTARD capacitor decays at a much slower rate (because of a lower current draw). This means CusTARD can measure longer times between shutdowns. The use of an ADC[3] allows greater measurement precision than with TARDIS.

Algorithm 2 gives more details about the implementation of CusTARD.

ALGORITHM 2: CusTARD Implementation

```
INIT()
1  ▹ Charge the capacitor using a GPIO on the microcontroller.
2  CHARGE_CAPACITOR()
3  temperature ← MEASURE_TEMPERATURE()

TIME(temperature)
1  ▹ Estimate the passage of time by comparing the voltage on the capacitor to a precompiled
     table.
2  voltage ← READ_CAPACITOR_VOLTAGE()
3  time ← ESTIMATE(voltage,temperature)
4  return {time}
```

INIT: CusTARD first charges the timekeeping capacitor. Then, similar to TARDIS, CusTARD uses the on-board temperature sensor to detect and compensate for temperature changes that affect the decay rate. The procedure MEASURE_TEMPERATURE stores inside-the-chip temperature in the FRAM memory upon power-up.

TIME: The CusTARD TIME procedure returns the *time* since the microcontroller last had power. Using a precompiled table, the voltage on the capacitor is used as a lookup to a time value.

3.3. CusTARD Variation

An RTC with a backup button battery or supercapacitor cannot be used with batteryless sensing devices. However, a sufficiently low-power RTC, with certain startup characteristics, powered by a single small (0.1μF or less) capacitor could be useful for some systems. A variation on CusTARD would be to replace the resistor R_2 in Figure 4 with an ultra-low-power RTC such as the NXP PCF2123 [NXP Semiconductors SPI Real time clock/calendar 2012] or Abracon AB08X5 [Abracon Real time clock 2015]. Using this scheme, the sensor could have access to very precise timekeeping information, but at a much higher cost than previous methods.

[3]An ADC is a ubiquitous component in low-power microcontrollers.

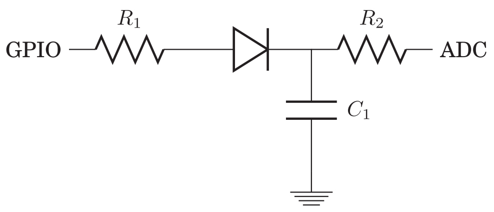


Fig. 4. Schematic of CusTARD showing the charge throttling resistor (R_1) and diode, a capacitor (C_1), and a high value discharge resistor (R_2).

The extra board space, designation of at least three GPIO lines for the SPI interface, in addition to all the circuitry required for CusTARD can be prohibitive for some applications. The ADC is still required so that the system knows if the RTC reset. Additionally, many RTCs have a nontrivial startup time (up to and exceeding 1.5s) which if not handled carefully can skew time results.

4. IMPLEMENTATION

In this section, we describe the hardware and software for two remanence-based timekeeping techniques. We also describe the profiling infrastructure, and the tools used in the evaluation.

TARDIS: We implemented a TARDIS library that runs on the Umich Moo [Zhang et al. 2011], that provides the procedures listed in Algorithm 1. No additional hardware is needed for using TARDIS in an embedded application. The Moo is a computational RFID tag that is composed of a a MSP430F2131 processor, sensors, and a charge pump for harvesting energy from RFID readers. The MSP430F2131 has 256B of SRAM, part of which is dedicated for use by the TARDIS algorithm. The processor also comes equipped with a standard temperature sensor, which is used for temperature compensation in our algorithm.

CusTARD: We implemented a prototype of CusTARD with only four surface mount components; a charge throttling resistor, a reverse current protection diode, a capacitor, and a high value discharge resistor. The schematic is shown in Figure 4. The CusTARD capacitor is charged by a GPIO through the charge resistor and diode, the measurement resistor is connected to an ADC pin on the microcontroller. The charge resistor limits the amount of current supplied to the capacitor; higher values increase the charge time. The diode acts as a charge buffer when the microcontroller is off, while the measurement resistor buffers the capacitor from unknown pin states of the microcontroller's ADC when the microcontroller is off. We used commodity multilayer ceramic capacitors because of their low leakage and better stability over a wider temperature range than comparable aluminum or tantalum capacitors. The total size of the prototype is 12.60mm by 12.01mm. All hardware components used for CusTARD are commodity, off-the-shelf. The CusTARD prototype was used with the UMich Moo and the MSP430FR5739 for all evaluation experiments. The processor on the MSP430FR5739 comes equipped with a 10-bit ADC, enabling CusTARD functionality.

Experimentation: We used Data Acquisition (DAQ) units to record and output voltages and assist with experimentation. For all experiments with TARDIS, we used an Agilent U2541A series DAQ. For all experiments with CusTARD, we used either a Measurement Computing USB-201 or a Freescale ARM microcontroller with 16-bit ADC. To control the temperature during experiments, we used environmentally controlled chambers. For all TARDIS experiments, we used a Sun Electronics EC12 Environmental Chamber [Sun Electronic Systems 2011] capable of 0.5°C precision, and an OSXL450 infrared noncontact thermometer [Omega Engineering 2007] with

$\pm2^{\circ}$C accuracy to verify that the microcontroller has reached thermal equilibrium within the chamber before testing. For all of the experiments with CusTARD, we use a Kintrex IRT0421 infrared noncontact thermometer, or the contact thermometer on a Fluke 87V. For testing CusTARD below room temperature, we used a custom built thermoelectric cooler that uses Peltier modules. For testing capacitor voltage decay above room temperature, we used a 12V Flexible Heating Pad. For all temperature testing, experiments were only run after thermal equilibrium was reached, temperature was monitored throughout each experiment, and adjustments were made to keep temperatures within $\pm0.5^{\circ}$C of the target.

Profiling: For the most accurate use of both CusTARD and TARDIS, profiling of the decay rate of the SRAM or capacitor of the remanence timekeeper is required. This profiling automatically charges and discharges the SRAM or capacitor under test, and maps time values to decay values. The profiler consists of two programs; one program on the device under test (in our case the UMich Moo), called the *profiling client*, and one on an external ARM microcontroller, called the *profiling server*. The *profiling server* (1) powers up the target device, (2) waits for the target to send either the number of SRAM cells that have decayed (TARDIS), or the value on the capacitor (CusTARD), as well as the temperature gathered from the internal temperature sensor of the device, (3) waits for the capacitors to charge, (4) cuts power to the target device, and (5) waits for a predetermined time, and then repeats at step (1). In this way, a model of the decay can be built, by mapping SRAM or ADC values, along with temperature values, to actual powered off times of the device. As this profiling is going on, the *profiling server* sends this information back to a desktop computer for later parsing. The *profiling client* only needs to manage the remanence-based timekeeper, and send values to the *profiling server*. In our implementation of the profiling firmware, the *profiling client* and *profiling server* communicate with a simple two wire protocol. This process can be time consuming; however, it must only be conducted once for the device to be deployed. Lookup tables for CusTARD were created from these data traces using the Numpy/SciPy libraries [Van Der Walt et al. 2011] integrated in Python scripts. These scripts parse the output of the profiling hardware, and produce C header files (containing CusTARD functions and tables) that can be included in any program on CusTARD equipped sensors.

5. EVALUATION

In this section, we evaluate the timekeeping ability of two types of remanance-based timekeepers, TARDIS, and CusTARD. We also evaluate a variation on the CusTARD timekeeper. Since the timekeeping accuracy and precision of these techniques depend on capacitive decay, in our evaluation we examine the decay behavior of SRAM and the CusTARD capacitor and three factors that have major effects on this behavior, and therefore its timekeeping ability.

5.1. Methodology

We use two similar experimental setups to evaluate remanence-based timekeeping methods. The experiments use the circuits depicted in Figure 10, and follow the same general procedure.

TARDIS Experimental Setup: The circuit used in all TARDIS experiments is shown in Figure 5(a). 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 resets all bits to 1 in preparation for the next time power is disconnected. A DAQ unit precisely controls the timing of power-ups and power-downs between 3 and 0V, and also measures the voltage across the

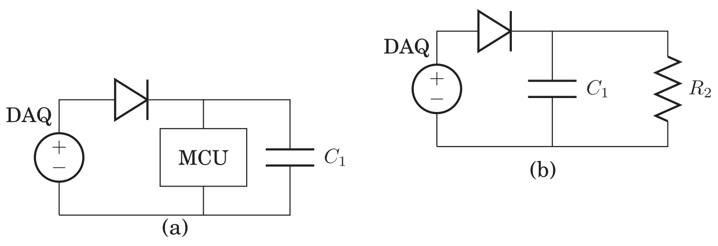


Fig. 5. Circuits used during our experiments. The TARDIS test circuit is shown in Figure 5(a). A portion of SRAM in the MCU is devoted to TARDIS. The unit is held in an environmental chamber to ensure consistent temperature during the tests. Figure 5(b) shows the circuit used in the experimental setup for evaluating factors that affect CusTARD. The DAQ for both circuits provides power to the microcontroller or capacitor and records the voltage decay.

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 V_{CC} during the off-time when the DAQ is still physically connected but is not supplying power. In all TARDIS experiments, microcontrollers from the TI MSP430 family are used to ensure consistency. The microcontroller used in all TARDIS experiments is MSP430F2131 with 256B of SRAM unless stated otherwise.

CusTARD Experimental Setup: For experiments evaluating factors of voltage decay, we used a DAQ that runs a program that charges (through a GPIO) the CusTARD capacitor to the regulated supply voltage (3.0V) minus the voltage drop of the diode. The power is then effectively disconnected until the voltage on the capacitor is less than 1% of the original value. The circuit for the experimental setup is shown in Figure 5(b).

In the remainder of this section, we evaluate factors that influence TARDIS (Section 5.2) and CusTARD (Section 5.3), we then evaluate a variation of CusTARD that uses an RTC (Section 5.4), and assess the overhead of each timekeeper (Section 5.5).

5.2. Factors Affecting SRAM Decay

In our evaluation of TARDIS, we examine the decay behavior of SRAM and three factors that have major effects on this behavior. Intuitively, the major component effecting SRAM decay is the size of the storage capacitor that powers the microcontroller. We present more details in the following sections.

Defining Stages of Decay: Three distinct stages of decay are observed in all experiments. Figure 6 illustrates the three stages of SRAM decay measured on a TI MSP430F2131 with 256B of SRAM and a 10μF capacitor at 26°C. We vary the *off-time* from 0 to 400s in 20-s 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 II for a summary of term definitions). Observations made during Stages 1 or 3 provide a single bit of 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 versus Voltage: The decay of SRAM is a function of its supply voltage. 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 7(a)) for five sets of tests (each at least 10 trials) demonstrate this. The same setup as explained before was used and five different temperatures (one with a 10mF capacitor and four of them without) were tested.

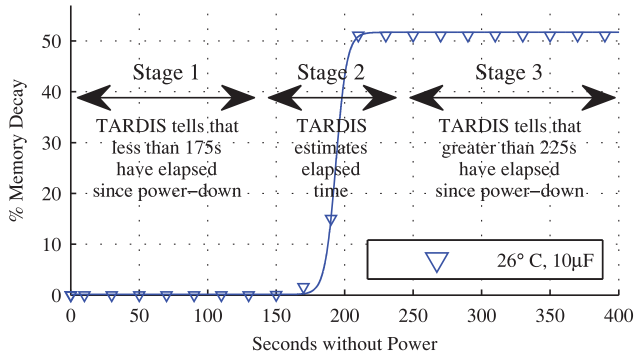


Fig. 6. TARDIS presents a three-stage response pattern according to its amount of decay. Before 175s, the percentage of bits that retain their 1-value across a power failure is 100%. For times exceeding 225s, the TARDIS memory has fully decayed. The decay of memory cells between these two thresholds can provide us with a more accurate measurement of time during that period. This graph presents our results measured on a TI MSP430F2131 with 256B of SRAM and a 10μF capacitor at 26°C.

Table II. Definition of the Terms Used to Explain the Behavior of SRAM Decay and the Theory Behind It

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 the first SRAM cell and the 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	DRV, 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.

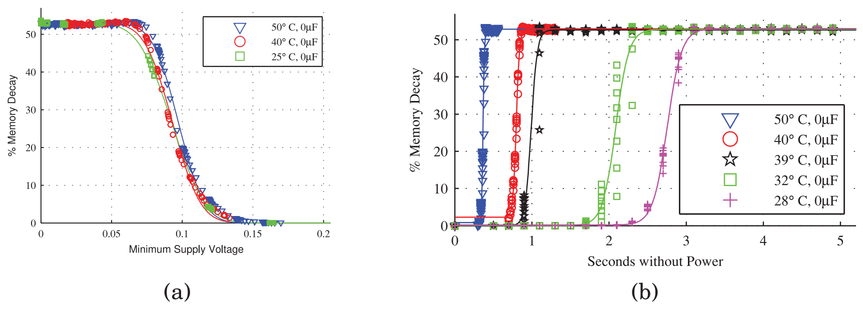


Fig. 7. Figure 7(a) shows that regardless of temperature, the amount of decay depends almost entirely on the minimum supply voltage reached during a power-down. Therefore, TARDIS can generally only resolve whether supply voltage remained above 0.15, dropped below 0.05, or reached a minimum value between these two. Figure 7(b) shows that the duration of SRAM decay is nonzero across all temperatures even when no capacitor is used.

Table III. Estimated Time in Stage 1 and Stage 2 of TARDIS Increases as Capacitor Size Increases. The Experiments are Done on a MSP430F2131 Microcontroller at 26.5°C and an SRAM size of 256B. Stage 1 is the Time After the Power Failure but Before the SRAM Decay. Stage 2 Represents the Duration of SRAM Decay

Cap. Size	**Stage 1 (s)**	**Stage 2 (s)**
0μF	1.22e0	8.80e-1
10μF	1.75e2	5.00e1
100μF	1.13e3	8.47e2
1000μF	1.17e4	9.50e3
10000μF	1.43e5	>5.34e4[5]

Impact of Temperature: The work of Skorobogatov [Skorobogatov 2002] shows that low temperature can increase the remanence time of SRAM. For 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°–50°C. The off-time of the microcontroller varied from 0 to a maximum of 5s. Figure 7(b) shows that the decay time is nonzero across all temperatures. This indicates that TARDIS could work at various temperatures as long as changes in the temperature are compensated for. For TARDIS, this compensation is done by using temperature sensors that are available in many of today's microcontrollers.[4] Further methods of dealing with temperature changes, and temperature attacks, are discussed in the conference paper by Rahmati et al. [2012].

Impact of Additional Supply Capacitance: Capacitors can greatly extend the resolution time of TARDIS, as they delay the onset of the SRAM segment changing to the default state. In our experiment, we have tested five different capacitors ranging from 10μF to 10mF 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 5.2) to calculate the time frame that can be resolved with each capacitor. Table III summarizes the results for the duration of TARDIS Stages 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 8.

Results ranging from seconds to days open the path for a wide variety of applications for TARDIS, as it can now be configured 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., [Vishay 2008; UMASS Security and Privacy Research Lab 2011]). Although a10mF capacitor size might be large compared to the size of today's transiently powered devices, the progress in capacitor size and capacity may very well make their use possible in the near future. However, at that size of capacitor, application developers may be better served using a CusTARD-based approach.

Impact of SRAM Size: We hypothesize that SRAM size has an inverse relation with decay time, 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 256B, 2KB, and 8KB at 28°C with no capacitor. The

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

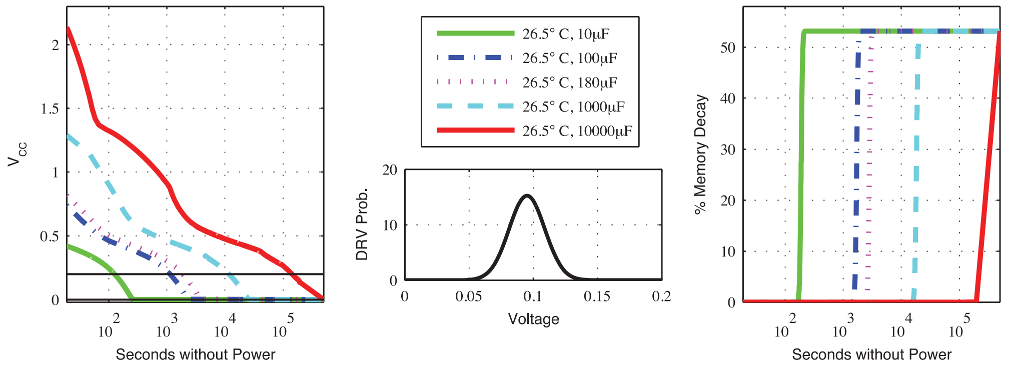


Fig. 8. For five different capacitor values, measured supply voltage traces are combined with a precharacterized DRV distribution to predict memory decay as a function of time. The decaying supply voltages after power is turned off are shown at left. DRV probabilities predict memory decay as a function of minimum supply voltage based on experimental data (Figure 7(a)). The two horizontal lines in the left image at approximately 150 and 50mV are the voltages where the first and last bits of SRAM will respectively decay.

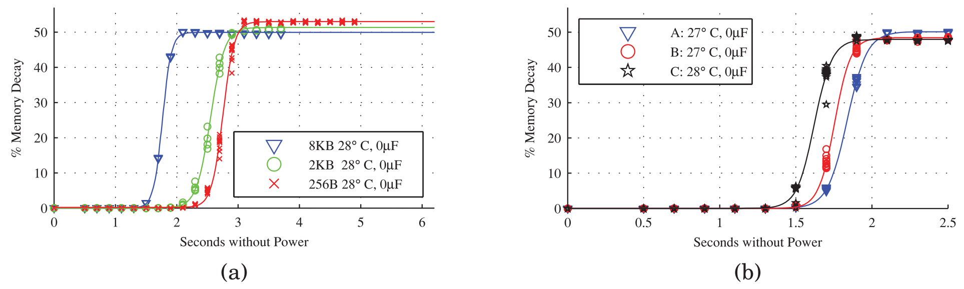
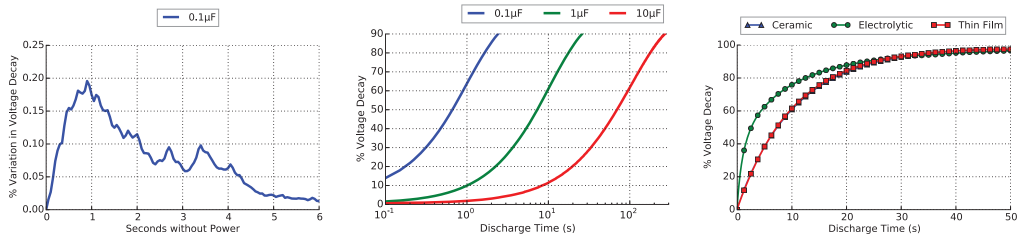


Fig. 9. (a) shows the effect of SRAM size on decay time. (b) shows the decay versus time in three different instances of the MSP430F2131.

experiment results are consistent with our hypothesis and are shown in Figure 9(a). It should be noted that SRAM size is not the only difference between these three models, as they also have slightly different power consumptions.

Impact of Chip Variation: The chip-to-chip variation of the same microcontroller model is not expected to have a major effect on TARDIS. We tested three instances of the MSP430F2131 with 256B of memory and no capacitor at 27°C. The result shown in Figure 9(b) 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 1° higher). This result indicates that TARDIS would work consistently across different chips of the same platform.

Summary: Each of the factors evaluated contribute to the final timekeeping ability of TARDIS. Temperature variation will cause timekeeping error; however, temperature differences are easily captured by the precision of most internal temperature sensors (e.g., MSP430 temperature readings have an error of 3°C). However, length of measurement time will always increase with temperature, as shown. Smaller SRAM size is preferable because of the memory constraints of batteryless devices, and the higher leakage rate on the capacitor. However, it is possible that no SRAM space is available on the microcontroller because of usage by the existing program. In this case, an alternative method such as CusTARD should be used. The size of the energy storage capacitor for the microcontroller has the effect of moving the precision timing window



(a) (b) (c)

Fig. 10. (a) shows that for any given temperature, the duration of capacitor voltage decay is consistent across trials. Discharge curves never varied more than 0.2% from the mean decay. (b) shows the voltage decay over time for three different CusTARD capacitor sizes. The charge time and energy cost increases with the size of the capacitor. (c) shows the effect of capacitor type on the duration of voltage decay.

of Stage 2 (Figure 6) by increasing the length of Stage 1. This extends the total measurement window. The great advantage of the TARDIS technique, is that it can be added to existing devices with only a code update. With millions of devices already in use in the IoT, TARDIS can begin to enable timekeeping in "smart dust" sensors.

5.3. Factors Affecting Capacitor Decay

In our evaluation of CusTARD, we examine the decay behavior of capacitors and four factors that can affect this behavior. All experiments use the same circuit (Figure 5(b)). For all experiments, the CusTARD capacitor is monitored using the DAQ as described in Section 5.1.

Impact of Temperature: Capacitor characteristics change with temperature, depending on the dielectric used, and the temperature coefficient. The rate of voltage decay and the ability to hold charge will change with its temperature. Designers are often forced to trade off capacity, cost, size, or leakage with temperature stability. To observe the effect of temperature on CusTARD, we ran the experiment described in Section 5.1 for five different temperatures with a 0.1μF multilayer ceramic capacitor with X7R dielectric. For each temperature, 10 charge and discharge cycles were performed. For all temperatures tested, decay rate differences between temperatures were negligible. Variances between temperatures were at most 0.2% as shown in Figure 10(a). The X7R dielectric makes CusTARD resilient to temperature changes. CusTARD is less sensitive to temperature than TARDIS because its discharge current is determined by the resistance of the discharge resistor, and capacitor chemistry prevents major variation.

Impact of Capacitor Size: For the same temperature, charge voltage, and capacitor type, a larger capacitor will have a longer voltage decay. We performed an experiment where capacitors of different sizes were discharged over the same resistance, after being charged through a diode. All experiments were conducted at 26°C. The time the capacitor took to discharge was measured. The results of this experiment are shown in Figure 10(b). The larger capacitor does come with a reliability and energy cost. Since the larger capacitor takes longer to charge up to the reference voltage, during this charge time, the microcontroller could be reset, reducing reliability of the timekeeper. The larger capacitor also requires more energy to charge it to the reference voltage, meaning that energy starved applications may not always be able to support a larger CusTARD. The larger the capacitor the longer it takes for the voltage to decay past the point where it will register on the ADC. For the 10-bit ADC with 3V reference used in our experiments, this value was 3mV; using a lower reference and more precise ADC would greatly extend the measurement range. The cost trade-off is shown in Table V.

The values in Table V were calculated assuming ideal components; the cost in practice could be much worse.

Impact of Capacitor Variation: The dielectric of the CusTARD capacitor can affect the length of voltage decay. We profiled three different capacitor types: electrolytic, thin film, and ceramic, at the same temperature and the same capacity. The results of this experiment are shown in Figure 10(c). Electrolytic capacitors are known for their high leakage current, which is shown in the figure. However, after the voltage is mostly decayed, the rate of decay becomes nearly equal with the other capacitor types. All experiments for the preceding capacitor types were conducted with 1μF capacitors. By using multilayer ceramics, voltage decay can be extended for the same capacitance as thin film and electrolytic.

Impact of Chip Variation: Since CusTARD is a dedicated hardware addition, the choice of microcontroller will not have an appreciable effect on the voltage decay of the CusTARD capacitor. However, certain characteristics of the microcontroller will have an effect on the application level interaction between CusTARD and the microcontroller. The resolution of the on-board ADC will determine the granularity of the timekeeper; a higher resolution ADC will allow for more precise timekeeping. The stability of the supply voltage will also affect usage of CusTARD. The stable supply voltage decreases the error of the ADC reference. A more stable supply voltage (regulated) will increase accuracy from run-to-run, while a higher supply voltage will increase the length of voltage decay. The type of memory where timestamps are stored also has an incidental effect on the usage of CusTARD. Using FRAM instead of Flash allows quicker writes and fewer write failures due to insufficient supply voltage. We have found that an FRAM enabled MSP430 with a 10-bit ADC and a 3V regulated supply voltage is more than adequate for most applications.

Summary: CusTARD overcomes many of the disadvantages of TARDIS by using a hardware addition to keep time. CusTARD does not affect the usable memory space of the microcontroller, and gives much more precision for the same amount of time. However, components must be chosen carefully to have dielectrics with low sensitivity to temperature. Additionally, chips without ADCs are limited in the timing precision, however, microcontrollers of this type are becoming increasingly uncommon. CusTARD does present other potential security challenges when the attacker has physical access to the device, as the capacitor can be removed, or the voltage changed. This form of attack would be expensive, and difficult to execute on hundreds or more devices. The most significant advantage to the CusTARD approach is in terms of size and cost; capacitors (of the size necessary for CusTARD) are one of the cheapest electrical components, and can be found in packages with an area smaller than 0.08mm^2 while costing tenths of a cent. This low cost timekeeper can enable sensor deployments in high volumes, to meet the “smart dust” and IoT vision.

5.4. CusTARD RTC Variation

For batteryless applications where very precise time is critical, an alternative form of CusTARD with an RTC can be used. This can be done using the same configuration as CusTARD, with some modifications, namely, (1) replacing the discharge resistor with an RTC, (2) running Serial Peripheral Interface (SPI) I/O lines to the RTC from the microcontroller, and (3) writing a software driver to communicate with the RTC. The ADC must still be in place because, once the capacitor voltage drops below the operating threshold of the RTC, the timekeeping abilities become nondeterministic. The microcontroller must detect when this happens, and upon reboot, reset the RTC time.

Table IV. Overhead of TARDIS INIT and DECAY Procedures Measured for TARDIS *size* of 256 bytes

Procedure	**Energy Cost**	**Exec. Time**
INIT	$11.53\mu J \pm 2.47$	$2.80\mu s \pm 0.00$
DECAY	$37.22\mu J \pm 9.31$	$12.40\mu s \pm 1.10$

The same factors that effect CusTARD also effect the RTC variation.

(1) **Temperature:** Modern RTCs are already temperature compensated, and use circuitry and crystals that are not as susceptible to this variation. Additionally, as long as temperature resistant CusTARD capacitors are used, there will be little variation in terms of measurement window length.
(2) **Capacitor Size:** Just as with the standard CusTARD approach, a larger capacitor will significantly increase the measurement window.
(3) **Capacitor Variation:** Choosing a temperature resistant dielectric will increase the measurement window.
(4) **Chip Variation:** The RTC variation is more susceptible to chip variation; in addition to the ADC, the Microcontroller (MCU) must have an SPI port. However, SPI is a common peripheral in modern microcontrollers.

There are additional costs above those required by CusTARD, including the setup costs with communication over SPI, the reset costs when the RTC loses power, and the potential decrease in the size of the measurement window for the same size capacitor as standard CusTARD.

The initial setup costs, above those of standard CusTARD, are mainly centered around configuring the registers on the RTC. For the NXP PCF2123 RTC [NXP Semiconductors SPI Real time clock/calendar 2012], during the initial setup the RTC can draw as much as $80\mu A$ and take 500ms to become ready. However, once the setup is complete, the NXP PCF2123 RTC [NXP Semiconductors SPI Real time clock/calendar 2012] can measure time up to 2.1s after a power loss, while the Abracon AB08X5 [Abracon Real time clock 2015] can measure time up to 19s after a power loss.

Summary: Despite the issues of cost, the RTC variation of CusTARD could be an important part of the IoT sensing scope. Because of cost (425 times more expensive in quantities of 9000 or more than standard CusTARD), it is doubtful that these devices could be deployed densely. However, because of heterogeneity of the IoT, one could imagine that a CusTARD RTC variation equipped batteryless sensor could serve as a calibration hub for other sensors, potentially transmitting timing information for sensors equipped with standard CusTARD, who lost their time completely after a long time without harvested energy.

5.5. Overhead

The two most resource-consuming procedures of 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 IV shows that energy consumed in total by these two procedures is about $48.75\mu J$ and it runs in 15.20ms.

Our experiments of time and energy measurements are performed on Moo RFID sensor tags that use an MSP430F2618 microcontroller with 8KB of memory and a $10\mu F$ capacitor. A tag is programmed to perform one of the procedures, and the start and end of the task are marked by toggling a GPIO pin. The tag's capacitor is charged up to 4.5V 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

Table V. CusTARD Energy and Charge Time Costs

Size	Energy Cost	Charge Time
0.01μF	0.0013μJ	0.0461ms
0.1μF	0.0128μJ	0.4605ms
1.0μF	0.1276μJ	4.6052ms
10.0μF	1.2755μJ	46.0517ms

power harvesting at a precise capacitor voltage. We measured the voltage decay of the capacitor and the GPIO pin toggling using an oscilloscope. The energy consumption of the task is the difference of energy ($\frac{1}{2}CV^2$) at the start and end of the task. The reported measurement is the average of ten trials.

The overhead of using CusTARD comes from (1) energy required to charge the capacitor to the regulated supply voltage, (2) the time required to charge the capacitor to the supply voltage, and (3) the energy cost of using the ADC to read the voltage on the capacitor. Table V shows the cost in time and energy for charging the CusTARD capacitor. The charge time is the amount of time it takes to charge the CusTARD capacitor to 99% of the supply voltage after the diode, or $t = -\ln \frac{V-0.99V}{V} RC$, where V is the supply voltage minus the forward voltage drop of the diode, R is the charge resistor, and C is the capacitance. It should be noted that the charging of the capacitor is nonblocking; a single instruction sets the GPIO pin to high, starting the charging of the capacitor; immediately following this instruction, the processor can continue with regular function. The energy cost is determined by the amount of capacitance, the voltage on the capacitor, and the heat dissipation of the resistor: $E = CV^2$. With the MSP430FR5739 used in CusTARD experimentation, the maximum energy required for the ADC read with default ADC settings was 11.52nJ. The overhead of using the CusTARD variation is nearly the same as that of the standard approach. The main difference being in cost: while the Abracon AB08X5 RTC can be bought for \$0.73 in quantities of 9000, a small ceramic capacitor capable of the same measurement length can be bought for \$0.00172 in quantities of 10,000.

The calculated overheads of each approach show that CusTARD uses an order of magnitude less energy, and takes less time than TARDIS for computation. However, CusTARD requires additional hardware, while TARDIS can be implemented using existing hardware platforms with only software.

6. USE CASES

We implemented one use case for each remanence timekeeper, using TARDIS and CusTARD in different embedded applications. Section 6.1 highlights how using TARDIS can help secure protocols for intermittently powered computational RFID tags like the UMich Moo, or Intel WISP. Section 6.2 shows how using CusTARD enables accurate, fine-grained timekeeping suitable for timestamping data collection even with numerous power failures. Both of these applications have implications for future IoT deployments as both secure computing and data collection are critical components of that vision.

6.1. Securing Protocols with 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 VI gives a summary of today's practical attacks on intermittently powered devices. With TARDIS, these applications could throttle their response rates and improve their security. We discuss three security protocols that could strengthen their defense against brute-force attacks by using TARDIS. To demonstrate the ease of integrating

Table VI. 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

Platform	Attack	#Queries
MIFARE Classic	Brute-force [Garcia et al. 2009]	$\geq$1500
MIFARE DESFire	Side-channel [Oswald and Paar 2011]	250,000
UHF RFID tags	Side-channel [Oren and Shamir 2007]	200
TI DST	Reverse eng. [Bono 2012; Bono et al. 2005]	$\sim$75,000
GSM SIM card	Brute-force [Goldberg and Bricenco 1999]	150,000

TARDIS, we have implemented and tested each of these security protocols on the Moo, a batteryless microcontroller-based RFID tag with sensors but without a clock [Zhang et al. 2011]. Our prototypes demonstrate the feasibility of 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 [Juels 2006]. 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 [Juels 2006]. By providing a secure notion of time, TARDIS makes it possible to implement *sleepy tags* that can sleep temporarily without requiring additional key PINs or cryptographic solutions.

To extend the sleep time of sleepy tags, one could use a counter along with 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 count by one and initializes TARDIS again. This loop will continue while the count is not zero. For example, using a counter initially set to 1000 and a TARDIS resolution time of 10s (requiring a supply capacitor of 10μF or less according to Figure 8), the tag could maintain more than 2h of delay.

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. 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 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 1min 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 [Avoine et al. 2008]. Another attack can accurately trace a specific passport by sending hundreds of queries per minute [Chothia and Smirnov 2010].

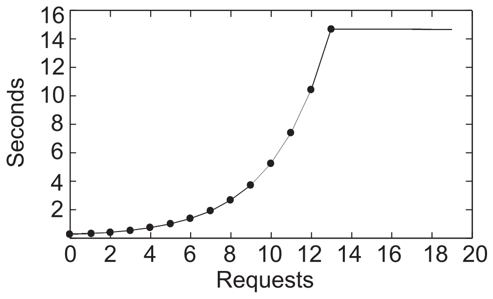


Fig. 11. Measured response time of a 2010-issued French passport [Avoine 2012]. The passport imposes up to 14s of delay on its responses after unsuccessful execution. The delay will remain until a correct reading happens even if the passport were removed from the reader's field for a long time.

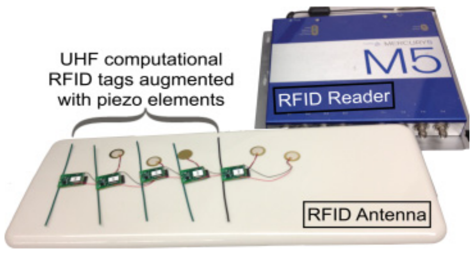


Fig. 12. TARDIS applications were implemented and tested on the Moo RFID sensors and remotely powered by a RFID reader (ThingMagic M5 [ThingMagic Inc. 2007]).

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 [Avoine 2012]. This delay increases to 14s after 14 unsuccessful attempts (Figure 11) 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. 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 (14s) would be enough to implement this function.

Implementation: For the three protocols described, 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 3. The tag was programmed to call the EXPIRED procedure upon power-up; immediately after calling expire, the SRAM is reinitialized. If the timer had expired, it would respond to the reader; otherwise, the tag would buzz an attached 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 [ThingMagic Inc. 2007] and its corresponding antenna to query the tag (shown in Figure 12). 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.

To measure the TARDIS resolution time on this platform, we powered up the tag to 3.0V using an external power supply and then disconnected it. We observed the voltage decay over time on an oscilloscope and measured the elapsed time between loss of power and when SRAM decay has finished. We conducted our experiments on five tags, which use a 10μF capacitor as its primary power source. The TARDIS resolution time on average was 12.03s with a standard deviation of 0.11s. A similar tag, which uses a

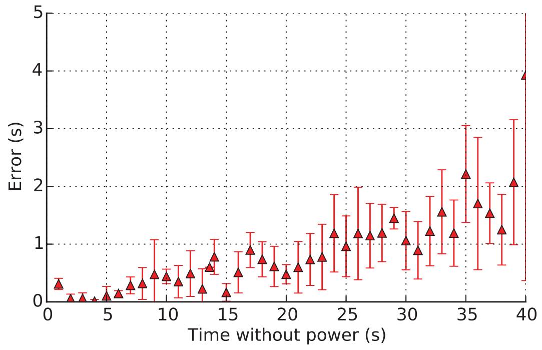


Fig. 13. This figure shows the timekeeping accuracy of CusTARD when used in an embedded system, with the mean error and standard deviation shown for each time interval. CusTARD in this configuration provides accurate timing of shutdowns up to 40s in length, with less than a 10% mean error for seven trials. CusTARD in this experiment was equipped with a 0.1μF multilayer ceramic capacitor. The y-axis shows the difference in seconds from the actual time elapsed that CusTARD gave to the microcontroller.

ALGORITHM 3: An example of TARDIS usage in a protocol.

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

100μF capacitor, yields a TARDIS resolution time of 145.85s. These time measurements are specific to the platform we have chosen for our experiment. The resolution could potentially be extended to hours using additional capacitors (Table III).

6.2. Keeping Time with CusTARD

We conducted experiments to evaluate the timekeeping ability of CusTARD when used in an embedded system. We used a CusTARD with a 0.1μF capacitor attached to a microcontroller. The microcontroller was programmatically turned on and off (through a diode) for different time intervals. For each interval, the microcontroller transmitted over a serial connection the amount of time in milliseconds it thought it had been shut down, which was gathered using the CusTARD hardware addition and our CusTARD software library. This value was compared to the actual time that the microcontroller had been off. Each time interval from zero to 40 with a step size of 1s was tried seven times.

The results of this experiment are shown in Figure 13. Using CusTARD, the microcontroller was able to keep track of the time since shutdown up to 40s with high accuracy. As shutdown time increases, accuracy decreases. CusTARD incurred a low energy cost of 312.5nJ to charge the capacitor, combined with a maximum of 11.52nJ to use the ADC on the microcontroller.

7. RELATED WORK

The related work for remanence-based timekeeping comes from both security applications, and advancements in low power technology.

RFID Security and Privacy: There are many applications for transiently powered devices that require a method to throttle the responses from a tag. The inability of intermittently powered devices to control their response rates has made them susceptible to various attacks. For example, 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 [Juels 2006]; however, lack of a persistent clock that enables wake up of the tag in time has made the use of the “sleep” command inconvenient. E-passports have been subject to brute-force attacks [Avoine et al. 2008], where the key can be discovered in real time. The attack could be slowed down if the e-passport had a trustworthy notion of time. The minimalist model [Juels 2005] offered for RFID tags assumes a scheme that enforces a low query-response rate. This model could be implemented using our approaches. Some RFID credit cards have used monotonically increasing transaction counters as a proxy for time, with some cards ceasing to function after the counter rolls over [Heydt-Benjamin et al. 2007].

Secure Timers: To acquire a trustworthy notion of time, multiple sources of time can be used to increase the confidence level an application has in a timer [Rousseau 2001]. This method 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 [Lamport 1978]. This inability to acquire secure time precludes the use of many cryptographic protocols, including timed-release cryptography [Mao 2001; Rivest et al. 1996].

Ultra-Low-Power Clocks and Timers: With the rise of pervasive computing comes 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 [Swanson and Meindl 1972]. Circuits in wristwatches combine analog components and small digital designs to operate at hundreds of nW [Vittoz 1994]. A counter designed for smart cards uses adiabatic logic to operate at 14kHz while consuming 11nW of power [Tessier et al. 2005]. A gate-leakage-based oscillator implements a temperature-invariant clock that operates at sub-Hz frequencies while consuming 1pW at 300mV [Lin et al. 2007]. 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. Most closely related to CusTARD, is a TI-recommended technique [Raju 2000] for the MSP430, that gives a hardware only, wakeup RC-timer. The technique charges a dedicated external capacitor from the microcontroller, then goes into a low-power sleep mode with clocks deactivated; the microcontroller is triggered to wake up when the capacitor voltage surpasses a threshold. CusTARD generalizes this technique to application clocks and intermittently powered, systems with unstable supply voltages. The addition of an ADC allows for precise measurement and fine-grained time. Additionally, this work quantifies the factors that impact measurement accuracy, and details the software and hardware methods for profiling CusTARD to reduce the effect of these factors, and improve overall accuracy. Moreover, we motivate this method as an enabling technology of the future IoT.

Security: Processes with long time constants can also raise security concerns by allowing data to be read from supposedly erased memory cells. Drowsy caches [Flautner et al. 2002] provide a good background on the electrical aspects of data retention.

Table VII. Comparison of Remanence Timekeepers

	TARDIS	**CusTARD**	**CusTARD RTC**
Precision	Low	High	Very High
Accuracy	Low	High	Very High
Cost	None	Low	Very High
Size	None	Small	Small

Gutmann stated that older SRAM cells can retain stored state for days without power [Gutmann 1996]. Gutmann also suggested exposing the device to higher temperatures to decrease the retention time. Anderson and Kuhn first proposed attacks based on low-temperature SRAM data remanence [Anderson and Kuhn 1996]. Experimental data demonstrating low-temperature data remanence on a variety of SRAMs is provided by Skorobogatov [Skorobogatov 2002], 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 [Tuan et al. 2007]. 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 [Saxena and Voris 2009]. Our proposed approaches, in finding a constructive use for remanence and decay, can thus be seen as a counterpoint to the attacks discussed in Section 6.1. They are the first *constructive* method that takes advantage of SRAM remanence to increase the security and privacy of intermittently powered devices.

8. DISCUSSION

Remanence-based timekeepers are mainly applicable to batteryless sensors. This emerging type of device has unique abilities (long lifetime, cheap, small) that enable the future of sensing, but unique constraints that make these platforms difficult to develop for. Remanence-based timekeepers help solve the problem of stable, trustworthy time, in batteryless sensing. Each of the different timekeepers have strengths and weaknesses, however, all are applicable, enabling technologies for the IoT. In the rest of this section, we outline limitations of our approach, and alternatives.

Limitations and Tradeoffs: The major drawback of all the remanence-based timekeepers is the reliance on the developers' knowledge of the application, and the energy harvesting environment in which the timekeeper equipped sensor will be deployed. The developer must be able to make reasonable estimations of the time the sensor will be powered off, because of energy scarcity. Using this information, they can decide on factors like the size of SRAM, the size of the supply capacitor, or the size of the CusTARD capacitor. However, if the developer estimates poorly, applications and timekeeping will suffer. For example, using a CusTARD capacitor that is too large could mean that the capacitor never fills up, and therefore application computation never gets done because of energy wasted by CusTARD. In another scenario, if a developer dedicates too large an SRAM space to TARDIS, system performance may suffer because of the reduced RAM scratch space; also, programming bugs could cause parts of this addressable SRAM to be overwritten, potentially skewing timing information. To overcome these limitations, developers must take a holistic approach to application design, using tools like Ekho [Hester et al. 2014] to gather, understand, and test sensors in different energy environments. Additionally, developers should take into account code size and RAM usage of the final application, and then use the remaining SRAM space for the TARDIS algorithm.

Each of the remanence-based timekeepers has their own set of trade-offs. These are listed in Table VII. As stated, TARDIS can have a negative impact on the user code, because it shares the SRAM space. CusTARD, as a hardware component, requires a small redesign of the sensor board itself. The variation of CusTARD equipped with RTC costs orders of magnitude more than the other techniques, but gives the greatest measurement precision, and in some cases, the longest measurement window. Additionally, the RTC variation has a larger initialization and reset cost than the other techniques.

Alternatives: 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. The EPC Gen2 protocol [EPCglobal 2012] 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 TARDIS has a resolution time from seconds to hours and has a temperature compensation mechanism. Another advantage of TARDIS is that it works on any SRAM-based device regardless of the existence of special circuits to support inventorial flags.

9. CONCLUSIONS

The vision of the IoT, based on the original "smart dust" concept, is slowly taking shape. However, many enabling technologies must first materialize, before the trillions of devices that will make up the IoT can be deployed in an environmentally responsible manner. Batteryless sensing is one of these key enabling technologies that is sustainable, and cost effective. Remanence-based timekeepers provide a trustworthy, reliable source of time on these batteryless devices, enabling applications previously not possible. Using TARDIS, application designers can equip cryptographic protocols for more deliberate defense against semi-invasive attacks such as differential power analysis and brute-force attacks. Using CusTARD, robust sensing applications can be created that provide reliable timestamped sensed data from unreliable sensing devices.

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 TARDIS primarily rely on timers with hourglass-like precision to throttle queries. TARDIS is implemented in software, making the mechanism easy to deploy on devices with SRAM. Unlike TARDIS, CusTARD uses remanence decay of commodity capacitors to provide a finer-grained notion of time with lower energy costs in a compact hardware addition. Without remanence timekeepers, batteryless devices are unlikely to give you the time of day.

ACKNOWLEDGMENTS

The authors would like to thank Jonathan McCune, Gesine Hinterwalter, Karsten Nohl, David Oswald, and Joshua Smith for their feedback on applications; Gildas Avoine for information on passport communication and feedback on applications; Matt Reynolds for information on the EPC gen2 protocol; Charlie McDonald for building temperature control hardware; Quinn Stewart for proofreading; and members of the SPQR lab for reviewing early versions of this article.

REFERENCES

2011. An Introduction to the Architecture of Moo 1.0. (May 2011). https://spqr.cs.umass.edu/moo/Documents/Moo_01242011.pdf.

Abracon Real time clock 2015. Abracon Corporation AB08X5 Real-Time Clock Family. (2015). Retrieved December 10, 2015 from http://abracon.com/Precisiontiming/AB08X5-RTC.PDF.

Ross Anderson and Markus Kuhn. 1996. Tamper resistance: A cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*.

Gildas Avoine. 2012. Personal communication on French passports. (2012).

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

Steve Bono. 2012. Personal communication. (February 2012).

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

Isidor Buchmann. 2001. *Batteries in a Portable World*. Cadex Electronics Richmond.

Michael Buettner, Ben Greenstein, David Wetherall, and Joshua R. Smith. 2008. Revisiting smart dust with RFID sensor networks. In *Proceedings of ACM HotNets 2008*.

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

Ahmad H. Dehwah, Mustafa Mousa, and Christian G. Claudel. 2015. Lessons learned on solar powered wireless sensor network deployments in urban, desert environments. *Ad Hoc Networks* 28 (2015), 52–67.

Conrad Donovan, Alim Dewan, Deukhyoun Heo, and Haluk Beyenal. 2008. Batteryless, wireless sensor powered by a sediment microbial fuel cell. *Environmental Science & Technology* 42, 22 (2008), 8591–8596.

EPCglobal. 2012. *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHZ–960 MHZ, Version 1.2.0*.

Krisztián Flautner, Nam Sung Kim, Steve Martin, David Blaauw, and Trevor Mudge. 2002. Drowsy caches: Simple techniques for reducing leakage power. In *Proceedings of the 29th IEEE/ACM International Symposium on Computer Architecture*. 148–157. DOI:http://dx.doi.org/10.1109/ISCA.2002.1003572

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

Flavio D. Garcia, P. van Rossum, R. Verdult, and R. W. Schreur. 2009. Wirelessly pickpocketing a MIFARE classic card. In *IEEE Symposium on Security and Privacy*. 3–15. DOI:http://dx.doi.org/10.1109/SP.2009.6

Ian Goldberg and Marc Bricenco. 1999. GSM cloning. (1999). Retrieved February 19, 2012 from http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html.

Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems (CHES)*. 86–80.

Peter Gutmann. 1996. Secure deletion of data from magnetic and solid-state memory. In *Proceedings of the 6th USENIX Security Symposium*.

Josiah Hester, Timothy Scott, and Jacob Sorber. 2014. Ekho: Realistic and repeatable experimentation for tiny energy-harvesting sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (SenSys'14)*. ACM, New York, NY, 330–331. DOI:http://dx.doi.org/10.1145/2668332.2668382

Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the Coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*. ACM, New York, NY, 5–16. DOI:http://dx.doi.org/10.1145/2809695.2809707

Thomas S. Heydt-Benjamin, Dan V. Bailey, Kevin Fu, Ari Juels, and Tom O'Hare. 2007. Vulnerabilities in first-generation RFID-enabled credit cards. In *Proceedings of the 11th International Conference on Financial Cryptography and Data Security*, Lecture Notes in Computer Science, Vol. 4886. 2–14.

Daniel E. Holcomb, Wayne P. Burleson, and K. Fu. 2009. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers* 58, 9 (2009), 1198–1210.

Daniel E. Holcomb, Amir Rahmati, Mastooreh Salajegheh, Wayne P. Burleson, and Kevin Fu. 2012. DRV-fingerprinting: Using data retention voltage of SRAM cells for chip identification. In *RFIDSec'12: Proceedings of the 8th International Conference on Radio Frequency Identification: Security and Privacy Issues*. Springer-Verlag. https://spqr.eecs.umich.edu/papers/holcomb-rfidsec12.pdf.

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

Ari Juels. 2006. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications* 24, 2 (February 2006), 381–394. DOI:http://dx.doi.org/10.1109/JSAC.2005.861395

Joseph M. Kahn, Randy Katz, and Kristofer Pister. 1999. Next century challenges: Mobile networking for "smart dust". In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*. ACM, New York, NY, 271–278. DOI:http://dx.doi.org/10.1145/313451.313558

Daniel Hsing Po Kang, Mengjun Chen, and Oladele A. Ogunseitan. 2013. Potential environmental and human health impacts of rechargeable lithium batteries in electronic waste. *Environmental Science & Technology* 47, 10 (2013), 5495–5503.

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

Dominique Larcher and Jean-Marie Tarascon. 2015. Towards greener and more sustainable batteries for electrical energy storage. *Nature Chemistry* 7, 1 (Jan. 2015), 19–29. http://dx.doi.org/10.1038/nchem.2085

Yu-Shiang Lin, Dennis Sylvester, and David Blaauw. 2007. A sub-pW timer using gate leakage for ultra low-power sub-Hz monitoring systems. *Custom Integrated Circuits Conference* (2007).

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

Masateru Minami, Takashi Morito, and Hiroyuki Morikawa. 2005. Biscuit: A battery-less wireless sensor network system for environmental monitoring applications. In *Proceedings of the 2nd International Workshop on Networked Sensing Systems*. Citeseer.

Sydney Newman, C. E. Webber, and Donald Wilson. 1963. Doctor Who. (November 1963). Premiered on British Broadcasting Channel One.

NXP Semiconductors MIFARE Classic. 2012. NXP Semiconductors MIFARE Classic. (2012). Retrieved February 18, 2012 from http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/mifare_classic/.

NXP Semiconductors SPI Real time clock/calendar. 2012. NXP Semiconductors SPI Real time clock/calendar. (2012) Retrieved February 18, 2012 from http://www.nxp.com/documents/data_sheet/PCF2123.pdf.

Inc. Omega Engineering. 2007. *OSXL450 Infrared Non-Contact Thermometer Manual*.

Yossef Oren and Adi Shamir. 2007. Remote password extraction from RFID tags. *IEEE Transactions on Computers* 56, 9 (Sept. 2007), 1292–1296. DOI:http://dx.doi.org/10.1109/TC.2007.1050

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

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

Amir Rahmati, Mastooreh Salajegheh, Dan Holcomb, Jacob Sorber, Wayne P. Burleson, and Kevin Fu. 2012. TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks. In *21st USENIX Security Symposium (USENIX Security 12)*, 221–236. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/rahmati.

Murugavel Raju. 2000. *UltraLow Power RC Timer Implementation Using MSP430*. Texas Instruments Application Report SLAA119.

Damith C. Ranasinghe, Roberto L. Shinmoto Torres, Alanson P. Sample, Joshua R. Smith, Keith Hill, and Renuka Visvanathan. 2012. Towards falls prevention: A wearable wireless and battery-less sensing and automatic identification tag for real time monitoring of human movements. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 6402–6405. DOI:http://dx.doi.org/10.1109/EMBC.2012.6347459

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

Ronald L. Rivest, Adi Shamir, and David A. Wagner. 1996. *Time-Lock Puzzles and Timed-Release Crypto*. Technical Report. Cambridge, MA.

Ludovic Rousseau. 2001. Secure time in a portable device. In *Gemplus Developer Conference*.

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

Nitesh Saxena and Jonathan Voris. 2009. 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*.

Sergei Skorobogatov. 2002. *Low Temperature Data Remanence in Static RAM*. Technical Report UCAM-CL-TR-536. University of Cambridge Computer Laboratory.

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

Sun Electronic Systems Inc. 2011. *Model EC1X Environmental Chamber User and Repair Manual*.

Richard M. Swanson and James D. Meindl. 1972. Ion-implanted complementary MOS transistors in low-voltage circuits. *International Solid-State Circuits Conference* (May 1972).

Robert Szewczyk, Joseph Polastre, Alan Mainwaring, and David Culler. 2004. Lessons from a sensor network expedition. In *Wireless Sensor Networks*, Holger Karl, Adam Wolisz, and Andreas Willig (Eds.). Lecture Notes in Computer Science, Vol. 2920. Springer, Berlin, 307–322. DOI:http://dx.doi.org/10.1007/978-3-540-24606-0_21

Russell Tessier, David Jasinski, Atul Maheshwari, Aiyappan Natarajan, Weifeng Xu, and Wayne Burleson. 2005. An energy-aware active smart card. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems* (2005).

Texas Instruments Inc. 2011. *MSP430F21x1 Mixed Signal Microcontroller*. Texas Instruments Application Report SLAS439F (revised Aug. 2011).

ThingMagic Inc. 2007. *Mercury 4/MERCURY 5 User Guide*.

Tim Tuan, Tom Strader, and Steve Trimberger. 2007. Analysis of data remanence in a 90nm FPGA. *Custom Integrated Circuits Conference*.

Stefan Van Der Walt, S. Chris Colbert, and Gael Varoquaux. 2011. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22–30.

Vishay. 2008. 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.

Eric Vittoz. 1994. Low-power design: Ways to approach the limits. *International Solid-State Circuits Conference*.

Daniel Yeager, Fan Zhang, Azin Zarrasvand, Nicole T. George, Thomas Daniel, and Brian P. Otis. 2010. A 9 μA, addressable gen2 sensor tag for biosignal acquisition. *IEEE Journal of Solid-State Circuits* 45, 10 (Oct. 2010), 2198–2209.

Xianlai Zeng, Lixia Zheng, Henghua Xie, Bin Lu, Kai Xia, Kuoming Chao, Weidong Li, Jianxin Yang, Szuyin Lin, and Jinhui Li. 2012. Current status and future perspective of waste printed circuit boards recycling. *Procedia Environmental Sciences* 16 (2012), 590–597.

Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. *Moo: A Batteryless Computational RFID and Sensing Platform*. Technical Report UM-CS-2011-020. Department of Computer Science, University of Massachusetts Amherst, Amherst, MA.

Received May 2015; revised December 2015; accepted March 2016