CCCP: Secure Remote Storage for Computational RFIDs*

Mastooreh Salajegheh[1] Shane Clark[1] Benjamin Ransford[1] Kevin Fu[1] Ari Juels[2]
[1]*Department of Computer Science, University of Massachusetts Amherst*
[2]*RSA Laboratories, The Security Division of EMC*
{*negin, ssclark, ransford, kevinfu*}*@cs.umass.edu, ajuels@rsa.com*

Abstract

Passive RFID tags harvest their operating energy from an interrogating reader, but constant energy shortfalls severely limit their computational and storage capabilities. We propose *Cryptographic Computational Continuation Passing* (CCCP), a mechanism that amplifies programmable passive RFID tags' capabilities by exploiting an often overlooked, plentiful resource: low-power radio communication. While radio communication is more energy intensive than flash memory writes in many embedded devices, we show that the reverse is true for passive RFID tags. A tag can use CCCP to checkpoint its computational state to an untrusted reader using less energy than an equivalent flash write, thereby allowing it to devote a greater share of its energy to computation.

Security is the major challenge in such remote checkpointing. Using scant and fleeting energy, a tag must enforce confidentiality, authenticity, integrity, and data freshness while communicating with potentially untrustworthy infrastructure. Our contribution synthesizes well-known cryptographic and low-power techniques with a novel flash memory storage strategy, resulting in a secure remote storage facility for an emerging class of devices.

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

1 Introduction

Research involving low-energy computing systems has long treated radio as an energy-hungry resource to be used sparingly. Our work uncovers a key resource in which programmable passive RFID tags differ from higher-powered wireless embedded devices such as motes: *radio communication consumes less energy than persistent local storage.* We exploit radio as a resource to amplify the storage capabilities of an emerging class of batteryless, programmable devices called *computational RFIDs* (CRFIDs) [7, 26, 27].

The main idea of this paper is that a CRFID can securely use radio communication as a less energy-intensive alternative to local, flash-based storage. The smaller energy requirements of radio allow the CRFID either to devote more energy to computation or to accomplish the same tasks using less energy, which may translate into a longer operating range. We use established cryptographic mechanisms to protect against untrustworthy RFID readers that could attempt to violate the confidentiality, authenticity, integrity, and freshness of the data on a CRFID. However, the cryptographic overhead threatens to eliminate the energy advantage of remote storage. Thus, the main challenge is to design an energy-saving remote storage system that provides security under the constraints of passive RFID systems.

This paper uses computational state checkpointing as an example of an application that benefits from our techniques. *Cryptographic Computational Continuation Passing* (CCCP) enables CRFIDs to perform sophisticated computations despite limited energy and continual interruptions of power that lead to complete loss of the contents of RAM. CCCP extends the Mementos architecture [26] for execution checkpointing by securely storing a CRFID's computational state on the untrusted RFID reader infrastructure that powers the CRFID, thereby making program execution on CRFIDs robust against loss of power. The design of CCCP is motivated by (1) a desire to minimize the amount of energy devoted to flash memory writes and (2) the observation that a CRFID's backscatter transmission is surprisingly efficient compared to alternatives such as active radio (like that found

* Proc. of the 18th USENIX Security Symposium, August 2009.

in motes) or flash memory writes.

Our contribution in this paper is the synthesis of several existing ideas with techniques that are specifically applicable to computational RFIDs:

- We describe the design and implementation of CCCP, a *secure remote storage protocol that suits the characteristics and constraints of CRFIDs*, and we show how this protocol can be used in the contexts of execution checkpointing and external data storage on an untrusted RFID reader infrastructure (Sections 3, 4).
- Motivated by a desire to save energy when storing CCCP's numeric counters to nonvolatile memory, we introduce *hole punching* (Section 3.4.4), a unary encoding technique that allows a counter stored in flash memory to be updated economically, minimizing energy- and time-intensive flash erase operations. For a CRFID, less frequent flash erasure means more energy available for computation.

Since CCCP involves communication with a potentially untrustworthy RFID reader, it must ensure the integrity, confidentiality, and data freshness of checkpointed messages. For message integrity, CCCP employs UMAC [4], a Message Authentication Code (MAC) scheme based on universal hash functions (UHF) that involves the application of a cryptographically secure pseudorandom pad. Remotely stored messages in CCCP are encrypted for confidentiality using a simple stream cipher. CCCP's frequent use of key material motivates the use of opportunistic precomputation: when a CRFID is receiving abundant energy, CCCP generates and stores keystream bits in flash memory for later consumption. CCCP maintains a small amount of its own state in local nonvolatile memory, including a counter that must be updated during checkpoint operations when energy may be low. To minimize the energy required to update the counter, CCCP employs hole punching.

Conventional passive RFID tags perform rudimentary computation, often in extremely tight real-time constraints using nonprogrammable finite state machines [1], but CRFIDs offer true general-purpose computational capabilities, broadening the range of their possible applications (Section 6). Although CRFIDs offer more flexibility, they present challenging resource constraints. While sensor motes, which rely on batteries for power, often have an active lifetime measured in weeks or months, a CRFID may be able to compute for less than a second given a burst of energy, and may receive such bursts in quick succession—putting CRFIDs in an entirely different class with regard to energy constraints. Moreover, although CRFIDs have a small amount of flash memory available as nonvolatile storage, writing to this flash memory is energy intensive (Section 2).

Because CRFIDs are new and prototypes are not yet widely available for use in the laboratory, there is little previous work describing their applications or limitations; Section 7 summarizes relevant work that has appeared to date. CCCP extends a recent execution checkpointing system called Mementos [26] by adding remote, rather than local flash-based, storage capabilities to CRFIDs. While systems such as Mementos investigate how to effectively store checkpoints locally in trusted flash memory to achieve computational progress on CRFIDs despite power interruptions, CCCP focuses on using external, *untrusted* resources to increase tag storage capacity in a secure and energy-efficient manner.

2 Computational RFIDs: Background, Observations, Challenges

Consistent with the usage of RFID terminology, the term *Computational RFID* (CRFID) has two meanings: the *model* under which passively powered computers operate in concert with an RFID reader infrastructure, and the passively powered computers themselves. CRFIDs represent a class of programmable, batteryless computers [7, 26, 27]. The small size and low maintenance requirements of CRFIDs make them especially appealing for adding computational capabilities to contexts in which placing or maintaining a conventional computer would be infeasible or impossible. However, CRFID systems require that nearby, actively powered RFID readers provide energy whenever computation is to occur, a requirement that may not suit all applications.

The components of a CRFID are: a low-power microcontroller; onboard RAM; flash memory (on or off the microcontroller); energy harvesting circuitry tuned to a certain frequency (e.g., 913 MHz for EPC Gen 2 RFID); an antenna; a transistor between the antenna and the microcontroller to modulate the antenna's impedance; a capacitor for storage of harvested energy; one or more analog-to-digital converters; and optional sensors for physical phenomena such as acceleration, heat, or light. The first working example of a CRFID is the Wireless Identification and Sensing Platform, or WISP [29], a prototype device slightly smaller than a postage stamp (discounting its inches-long antenna). The WISP is built around an off-the-shelf TI MSP430 microcontroller.

Like passive RFID tags but unlike sensor motes, CRFIDs are powered solely by harvested RF energy and lack active radio components. Instead, such CRFIDs use *backscatter* communication: in the presence of incoming radio waves, a CRFID electrically modulates its antenna's impedance using a transistor, encoding binary information by varying the antenna's reflectivity. While the omission of active radio circuitry saves energy, it gives up the tag's autonomy; a CRFID can send and receive information only at the command of an RFID reader. A CRFID's lack of autonomy is one of the factors that makes it difficult to protect.

2.1 Frequent Power Loss on Tags, but Plentiful External Resources

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

Frequent loss of power may interrupt computation. The CRFID model posits computing devices that are primarily powered by RF energy harvesting, a mechanism that is naturally finicky because of its dependence on physical conditions. Any change to a CRFID's physical situation—such as its position or the introduction of an occluding body—may affect its ability to harvest energy. Existing systems that use RF harvesting typically counteract the effect of physical conditions by placing stringent requirements on use. For example, an RFID transit card reader presented with a card may behave in an undefined way unless the card is within 1 cm for at least 300 ms, parameters designed to ensure that the card's computation finishes while it is still near the reader. CRFID applications may preclude such a strategy: programs on general-purpose CRFIDs may not offer convenient execution time horizons, and communication distances may not be easily controlled. Without any guarantees of energy availability, it may be unreasonable to mandate that programs running on CRFIDs complete within a single energy lifecycle. As an extension of the Mementos system [26], CCCP aims to address the problem of suspending and resuming computations to facilitate spreading work across multiple energy lifecycles.

Storing remotely may require less energy than storing locally. Some amount of onboard nonvolatile memory exists on a CRFID, so an obvious approach to suspension and resumption is simply to use this local memory for state storage. However, to implement nonvolatile storage, current microcontrollers use flash memory, which imports several undesirable properties. While reading from flash consumes energy comparable to reading from volatile RAM, the other two flash operations—writing and erasing—require orders of magnitude more energy per datum (Table 3). Our measurements of a CRFID prototype reveal that the energy consumption of storing a datum locally in flash can in fact exceed the energy consumption of transmitting the same datum via backscatter communication.

To illustrate the difference between flash and radio storage on a CRFID and to show how the relationship is different on a sensor mote, we offer Figure 1. The figure helps explain why designers of mote-based systems choose to minimize radio communication; similarly, it justifies our exploration of radio-based storage as an alternative to flash-based storage on CRFIDs.

It should be noted that CCCP, although its primary data storage mechanism is the communication link between CRFIDs and readers, still requires *some* flash writes during storage operations: CCCP maintains a counter in flash to ensure that key material is not reused. However, because CCCP employs hole punching (Section 3.4.4) to maintain the counter, the amount of data written for counter updates is small compared to the amount of data that can be stored at once—small enough not to obviate the energy advantage of radio-based storage—and counter updates do not frequently necessitate erasures.

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

RFID protocols allow arbitrary payloads. While the EPC Gen 2 protocol imposes constraints on the transmissions between RFID tags and RFID readers—for example, the maximum upstream data rate from tag to reader is 640 Kbps [13]—it also offers sufficient flexibility that CCCP can be implemented on top. In particular, the Gen 2 protocol permits a reader to issue a *Read* command to which a tag can respond with an arbitrary amount of data. Previous versions of the EPC RFID standard mandated a small response size that would have imposed severe communication overhead on large upstream transmissions.

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

2.2 Challenges Due to Energy Scarcity

Several energy-related considerations limit the resources available for computation on CRFIDs, limiting the utility of CRFIDs as a general-purpose computing platform. Ransford et al. [26] discuss the difficulty of effectively utilizing a storage capacitor and enumerate the drawbacks of using capacitors for energy storage; Buettner et al. [7] discuss how energy limitations bear on the de-

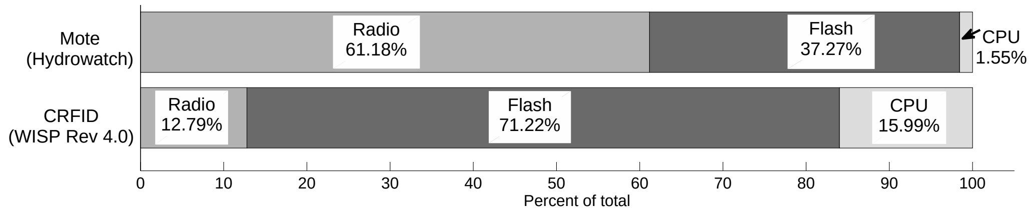

Figure 1: Per-component maximum power consumption of two embedded devices. Radio communication on the WISP requires less power than writes to flash memory. The relative magnitudes of the power requirements means that a sensor mote favors shifting storage workloads to local flash memory instead of remote storage via radio, while a computational RFID favors radio over flash. The numbers for the mote are calculated based on the current consumption numbers given by Fonseca et al. [15]. For the CRFID, we measured three operations (radio transmit, flash write, and register-to-register move) for a 128-byte payload.

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

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

The size of a CRFID's storage capacitor imposes another basic limitation. Flash writes, which owe their durability to a process that effects significant physical changes, require more current and time (and therefore energy) than much simpler RAM or register writes. Per-datum measurements show that, on a WISP's microcontroller, writing to flash consumes roughly 400 times as much energy as writing to a register [26]. Such outflow from the storage capacitor can dramatically shorten the device's energy lifecycles.

Non-autonomous operation. Backscatter communication involves modulating an antenna's impedance to reflect radio waves—an operation that, for the sender, involves merely toggling a transistor to transmit binary data. Such communication cannot occur without a signal to reflect; CRFIDs, like other passive RFIDs, are therefore constrained to communicate only when a reader within range is transmitting. Computation may occur during times of radio silence, but only if sufficient energy remains in the CRFID's storage capacitor. Unlike battery-powered platforms that can operate autonomously between beacon messages from other entities, a CRFID may completely lose power between interactions with an RFID reader. Our experience shows that, lacking a source of harvestable energy, the storage capacitor on a WISP (Revision 4.0) can support roughly one second of steady computation before its voltage falls below the microcontroller's operating threshold. Such limitations constrain the design space of applications that can run on CRFIDs. For example, without autonomy, an application cannot plan to perform an action at a specific time in the future.

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

3 Design of CCCP

CCCP's primary design goal is to furnish computational RFIDs with a mechanism for secure outsourced storage that facilitates the suspension and resumption of programs. This section describes how CCCP is designed to meet that goal and several others. Refer to Section 4 for a discussion of CCCP's implementation, and refer to Section 5 for an evaluation of CCCP's design choices and security; in particular, Section 5.3.1 discusses the overhead imposed by cryptographic operations.

Given a chunk of serialized computational state on a CRFID, CCCP sends the state to the reader infrastructure for storage. (CCCP is designed to work independently of the state serialization method, and does not prescribe

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

Table 1: CCCP's design goals and techniques for accomplishing each of them.

a specific method.) In a subsequent energy lifecycle, an RFID reader that establishes communication with the tag sends back the state, CCCP performs appropriate checks, and the CRFID resumes computation where it left off. CCCP provides several operating modes that allow an application designer to increase security—by adding authentication alone, or authentication and encryption—at the cost of additional per-checkpoint energy consumption. Table 1 describes how CCCP meets each of the goals discussed in this section.

3.1 Design Goal: Computational Progress on CRFIDs

CCCP remotely checkpoints computational state to make long-running operations robust against power loss—i.e., to enable their *computational progress*. We define computational progress as change of computational state toward a goal (e.g., the completion of a loop). While CRFIDs are able to finish short computations in a small number of energy lifecycles (e.g., symmetric-key challenge-response protocols [9, 19]), the challenges described in Section 2 make it difficult for a CRFID to guarantee the computational progress of longer-running computations.

If a CRFID loses power before it completes a computation, all volatile state involved in the computation is lost and must be recomputed in the next cycle. If energy availability is similarly inadequate in subsequent cycles, the CRFID may never obtain enough energy to finish its computation or even to checkpoint its state to flash memory. We refer to such vexatious computations as *Sisyphean tasks*. (Sisyphus was condemned to roll a large stone up a hill, but was doomed to drop the stone and repeat hopelessly forever [20].) A major goal of CCCP is to prevent tasks from becoming Sisyphean by shifting energy use away from flash operations and toward less energy-intensive radio communication.

3.2 Checkpointing Strategies: Local vs. Remote

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

The obvious advantage of flash memory is that its proximity to the CRFID makes it readily accessible. On-chip flash has the further advantage that it may be inaccessible to an attacker. However, the operating requirements of flash are onerous in many situations. With unlimited energy, a CRFID could use flash freely and avoid the complexity of a radio protocol such as CCCP. Unfortunately, energy is limited in ways described elsewhere in this paper, and several disadvantages of flash memory diminish its appeal as a store for checkpointed state. The most obvious disadvantage is an imbalance between the requirements for reading and writing. Write and erase operations require more time and energy per bit than reading (Table 3); additionally, the minimum voltage and current requirements are higher. For example, in the case of the MSP430F2274, read operations are supported at the microcontroller's minimum operating voltage of 1.8 V, but write and erase operations require 2.2 V. Finally, flash memory (both NOR and NAND types) generally imposes the requirement that memory segments be erased before they are written: if a bit acquires a zero value, the entire segment that contains it must be erased for that bit to return to its default value of 1. Aside from burdening the application programmer with inconvenience, erase-before-write semantics complicate considerations of energy requirements. These disadvantages are minor afflictions for higher-powered systems, but they pose serious threats to the utility of flash memory on CRFIDs.

Backscatter transmission, since it involves modulating only a single transistor to encode data, requires significantly less energy than transmission via active radio. In fact, our measurements (Figure 4) show that backscatter transmission of an authenticated, encrypted state checkpoint (plus a small amount of bookkeeping in flash) can require less energy than exclusively writing to flash memory, even after including the energy cost of encrypting and hashing the checkpointed state. Because of its consistent behavior throughout the microcontroller's operating voltage range, backscatter transmission is an especially attractive option when the CRFID receives radio contact frequently but cannot harvest energy efficiently, in which case writing to flash may be infeasible because of insufficient energy in the storage capacitor. These circumstances may occur far from the reader, or in the presence of radio occlusions, or when a computation uses energy quickly as soon as the CRFID wakes up.

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

3.3 Threat Model

We define CCCP's threat model as a superset of the attacks that typically threaten RFID systems [21]. The most obvious way an attacker can disrupt the operation of a CRFID is to starve it of energy by jamming, interrupting, or simply never providing RF energy for the CRFID to harvest. Because they depend entirely on harvestable energy, CRFIDs cannot defend against such denial-of-service (DoS) attacks, so we consider these attacks as a problem to be dealt with at a higher system level. We instead focus on two types of attacks that a CRFID can use its resources to address: (1) active and passive radio attacks and (2) attacks by an untrusted storage facility.

An adversary may attempt to:

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

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

3.4 Secure Storage in CCCP

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

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

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

3.4.1 Keystream Precomputation

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

CCCP stores keystream material on the CRFID because we assume that the CRFID trusts only itself; a CRFID cannot extract trustworthy keystream material from a reader it does not trust, nor from any observable external phenomenon (which, in our threat model, an attacker would be able to observe equally well). Because a CRFID can reserve only finite storage for storage of keystream material, the material must be periodically refreshed. CCCP opportunistically refreshes the keystream material with pseudorandom bits, following Algorithm 3.

To provide unique keystream bits to cryptographic operations (encryption and MAC), CCCP uses an existing implementation [9] of the RC5 block cipher [28] in counter mode to generate pseudorandom bits and store them to flash. The choice of a block cipher in counter mode means that the resulting MAC and ciphertext are secure against a computationally bounded adversary [6]. A stream cipher would work equally well in principle, but in implementing CCCP, we found that those under consideration required a large amount of internal read-write state. For example, the stream cipher ARC4 requires at least 256 bytes of RAM [30], whereas RC5 requires only an 8-byte counter. The RC5 key schedule is preloaded into flash memory the first time the device is programmed, and the keystream materials are generated during periods of excess energy (or *power seasons*; see § 3.5). One such period of excess energy is the CRFID's initial programming, at which time the entire keystream buffer is filled with keystream bits. To avoid reusing keystream bits, CCCP maintains several variables in nonvolatile memory. Table 2 summarizes the variables CCCP stores in nonvolatile memory.

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

Table 2: Variables CCCP stores in nonvolatile memory.

3.4.2 UHF-based MAC for Authentication and Integrity

CCCP uses a MAC scheme based on universal hash functions (UHF) [8] to provide authentication and integrity. CCCP constructs the MAC by first hashing the message and then XORing the 80-bit hash with a precomputed cryptographic keystream. Because of the resource constraints of CRFIDs, it is critical to use a scheme that consumes minimal energy, and according to recent literature [4, 14], UHF-based MACs are potentially an order of magnitude faster than MACs based on cryptographic hash functions. We chose UMAC [4] as the MAC function after evaluating several alternatives. Our experiments on WISP (Revision 4.0) CRFIDs determined that UMAC takes on average 18.38 ms and requires 28.79 μJ of energy given a 64-byte input.

3.4.3 Stream Cipher for Confidentiality

To provide confidentiality, a CRFID simply XORs its computational state with a precomputed cryptographic keystream. This encryption scheme is low-cost in terms of computation and energy, but it relies on using each keystream bit at most once. CCCP ensures that the encryption and MAC functions never reuse keystream bits by keeping track of the beginning and end of fresh keystream material in flash memory. The keystream pool is represented as a circular buffer. The address of the first unused keystream material is derived from the value of *chkpt_counter* (Table 2), and the last unused keystream material ends just before the address pointed to by *kstr_end*.

If the application using CCCP demands confidentiality at all times, then if CCCP cannot satisfy a request for unused keystream bits, it pauses its work to generate more keystream bits. This behavior is inspired by that of the blocking `random` device in Linux [17].

3.4.4 Hole Punching for Counters Stored in Flash

To avoid reusing keystream material, CCCP maintains a counter (*chkpt_counter*) from which the address of the first unused keystream bits can be derived. The counter is stored in flash memory because it is used for state restoration after power loss. However, incrementing a counter stored in binary representation always requires changing a 0 bit into a 1 bit (Figure 2(a)). On segmented flash memories, changing a single bit to 1 requires the erasure (setting to 1) of the entire segment that contains it—at least 128 bytes on the MSP430F2274—before the new value can be written. An additional cost that varies among flash cells is that they wear out with repeated erasure and writing [18].

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

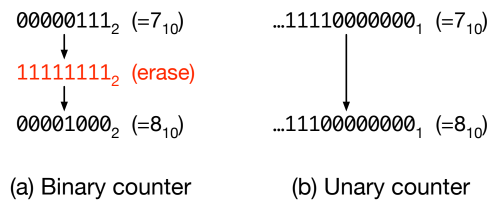

Figure 2: Illustration of hole punching. While incrementing a binary counter (a) in flash memory may require an energy-intensive erase operation, complemented unary representation ((b), with the number of zeros, or "holes," representing the counter value) allows for incrementing without erasure at a cost of space efficiency.

word. Table 3 illustrates the energy cost of erasing an entire segment and the energy cost of writing a single word.

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

Table 3: Comparison of energy required for flash operations on an MSP430F2274. Hole punching often allows CCCP to use a single-word write (2 bytes on the MSP430) instead of a segment erase when incrementing a complemented unary counter.

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

3.4.5 Extension for Long-Term Storage

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

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

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

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

3.5 Power Seasons

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

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

4 Implementation

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

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

The RFID reader side of CCCP consists only of code to drive the reader appropriately for communication events. Because of the Gen 2 protocol's complexity, we have not completely implemented the reader side of the CCCP protocol. Rather than write a large amount of code for the reader, we chose to use simple control programs for the reader and inspect the exchanged messages manually, a strategy that allowed us to concentrate on the more resource-constrained CRFID side of the system while avoiding porting applications from one proprietary reader to another. (The WISP [Revision 4.0] is nominally compatible with only the Alien ALR-9800 and Impinj Speedway readers; we chose to use a desktop PC to program these readers for the sake of simplicity and portability.) A full implementation of the reader side would properly parse each message received from the CRFID and manage storage for checkpointed state.

4.1 Communication Protocol

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

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

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

- A reader issues a *Read* command to a specific tag to request an arbitrary amount of data from an RFID tag's memory. A CRFID can respond to a coordinated *Read* command with a chunk of checkpointed state.

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

Figure 3 gives an overview of CCCP's message types and their ordering. CCCP does not require protocol changes to the EPC Gen 2 standard, but it requires that an RFID reader be controlled by an application that understands CCCP. While a proprietary radio protocol for CCCP could be more efficient than one built atop an existing RFID protocol, a goal of CCCP—inherited from the design goals of the WISP CRFID—is to maintain compatibility with existing RFID readers.

5 System Evaluation

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

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

```
CHECKPOINT($state, keystream, chkpt\_counter$)
≻ Compute the (constant) amount of keystream material that will be used in this invocation
$chkpt\_size = STATE\_SIZE + \text{LENGTH}(\langle state, chkpt\_counter \rangle) + 80$ bits

$k \leftarrow chkpt\_counter \times chkpt\_size$                    ≻ $keystream[k]$ holds unused keystream material
$chkpt\_counter \leftarrow chkpt\_counter + 1$                       ≻ Update $chkpt\_counter$ in nonvolatile memory

$C \leftarrow state \oplus keystream[k \ldots k + STATE\_SIZE - 1]$    ≻ Encrypt $state$ by XORing with keystream material
$k \leftarrow k + STATE\_SIZE$                                         ≻ ... and advance $k$

$H \leftarrow \text{NH}(\langle C, k \rangle, keystream[k \ldots k + \text{LENGTH}(\langle C, k \rangle) - 1])$    ≻ Hash the encrypted state
$k \leftarrow k + \text{LENGTH}(\langle C, k \rangle)$                ≻ ... and advance $k$

$M \leftarrow H \oplus keystream[k \ldots k + \text{LENGTH}(H) - 1]$  ≻ Construct an 80-bit MAC

TRANSMIT($C, M$)                                                       ≻ Note: TRANSMIT blocks until a reader is detected
```

5.1 Security Semantics

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

- Under CCCP's threat model, storing checkpointed state only in local flash memory is the most secure option, since it involves no radio transmission at all. However, for reasons detailed elsewhere in this paper, writing to flash memory is not always possible or desirable. We call the flash-only approach *Mementos* after the system [26] that inspired CCCP.
- In a mode called *CCCP/NoSec*, a CRFID sends computational state in plaintext. Under CCCP's threat model, CCCP/NoSec allows an attacker to intercept computational state and trivially recover the information it contains.
- In a mode called *CCCP/Auth*, the CRFID computes a message authentication code (MAC), attaches it to plaintext computational state, and transmits both. To trick a CRFID into accepting illegitimate state, an attacker must craft a message that incorporates a MAC that the CRFID can verify. However, since CCCP's MAC routine incorporates keystream material that is local to the CRFID, the attacker must guess the contents of a chunk of the CRFID's keystream memory, which requires brute force under our threat model.
- In a mode called *CCCP/AuthConf*, CCCP encrypts computational state, computes a MAC, and transmits both (Algorithm 1). As with CCCP/Auth, an attacker who wants to trick a CRFID into accepting illegitimate state must find a hash collision; however, part of her colliding input must be a valid *encrypted* computational state from which the CRFID would be able to resume. Since CCCP does not reuse keystream material, the attacker is limited to brute-force search to find such an encrypted state.

5.2 Experimental Setup & Methods

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

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

```
RESUME($C, M, keystream, chkpt\_counter$)
1  ▷ Find the first unused keystream material, then backtrack to find the keystream material CHECKPOINT used to
     hash and MAC the ciphertext
2  $chkpt\_size = STATE\_SIZE + \text{LENGTH}(\langle C, chkpt\_counter\rangle) + 80$ bits        ▷ N.b.: $STATE\_SIZE = \text{LENGTH}(C)$
3  $k \leftarrow chkpt\_counter \times chkpt\_size$
4  $k \leftarrow k - (\text{LENGTH}(\langle C,k\rangle) + 80$ bits$)$
5
6  $H \leftarrow \text{NH}(\langle C,k\rangle, keystream[k \ldots k + \text{LENGTH}(\langle C,k\rangle) - 1])$        ▷ Compute the ciphertext's hash
7  $k \leftarrow k + \text{LENGTH}(\langle C,k\rangle)$        ▷ ...and advance $k$ to point to the MAC
8
9  if $M = H \oplus keystream[k \ldots k + \text{LENGTH}(H) - 1]$        ▷ If the MAC is OK, then...
10    then $k \leftarrow k - (\text{LENGTH}(C) + \text{LENGTH}(\langle C,k\rangle))$        ▷ backtrack further ...
11         $state = C \oplus keystream[k \ldots k + \text{LENGTH}(C) - 1]$        ▷ and decrypt $C$ to yield $state$
12         RESTORE-STATE($state$)
13    else ▷ Do nothing
```

charge accumulation, making it difficult to shut off the energy supply at a precise capacitor voltage.

After watching the GPIO pin signal the beginning and end of the task, we calculated the task's duration and the corresponding change in the storage capacitor's voltage. When an operation completed too quickly to observe clearly on the oscilloscope, we repeated it in an unrolled loop and divided our measurements by the number of repetitions. Finally, we calculated per-bit energy values by subtracting the baseline energy consumption of the WISP with its MSP430 microcontroller in the LPM3 low-power (sleep) mode. We subtract the WISP's baseline energy consumption in order to discount the effects of omnipresent consumers such as RAM and CPU clocks. For all measurements that we present, we give the average of five trials.

5.3 Performance

Figure 4 shows that, for data sizes greater than 16 bytes, a checkpoint operation under CCCP/NoSec requires less energy than a checkpoint to flash. Under CCCP/AuthConf, which adds encryption and MAC operations, a similar threshold exists between 64 and 128 bytes. Checkpointing via flash has an additional cost: if the checkpointing mechanism needs to overwrite existing data (e.g., old checkpoints) in flash memory, it must erase the corresponding flash segments and potentially replace whatever data it did not overwrite. Even if a flash write does not necessitate an immediate erasure, it makes less space available in the flash memory and therefore increases the probability that a long-running application will eventually need to erase the data it wrote—that is, it incurs an *energy debt*. In the ideal case, an application can pay its energy debt easily if erasures happen to occur only when energy is abundant—i.e., in summer power seasons. However, since CCCP is designed to address scenarios in which energy availability fluctuates, we consider the case in which each write incurs an energy debt. Factoring in debt, we characterize the energy cost of a write of size *dsize* as

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

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

The energy measurements we present in this paper (e.g., in Figure 4) fail in some cases to strongly support the hypothesis that radio-based checkpointing is consistently less energy intensive than flash-based checkpointing. The imbalance is due to a missed opportunity for optimization on the WISP prototype. The transistor used

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

```
KEY-REFRESH($keystream, kstr\_end, chkpt\_counter, rc5counter$)
 1  ≻ Find the first unused keystream material in the circular keystream buffer
 2  $chkpt\_size = STATE\_SIZE + (STATE\_SIZE + \text{LENGTH}(\langle null, chkpt\_counter \rangle)) + 80$ bits
 3  $k \leftarrow chkpt\_counter \times chkpt\_size \pmod{\text{LENGTH}(keystream) / chkpt\_size}$
 4
 5  ≻ Erase all used keystream memory, then write pseudorandom data to it
 6  ERASE-MEMORY-RANGE($keystream[kstr\_end \ldots k]$)
 7  $i \leftarrow kstr\_end$
 8  while ($i < k$)
 9      do $rc5counter \leftarrow rc5counter + 1$              ≻ Update counter in nonvolatile memory
10         $keystream[i] \leftarrow \text{RC5}(rc5counter - 1)$     ≻ Write keystream material into nonvolatile memory
11         $kstr\_end \leftarrow i + 1$                        ≻ Update kstr_end in nonvolatile memory
12         $i \leftarrow i + 1$
```

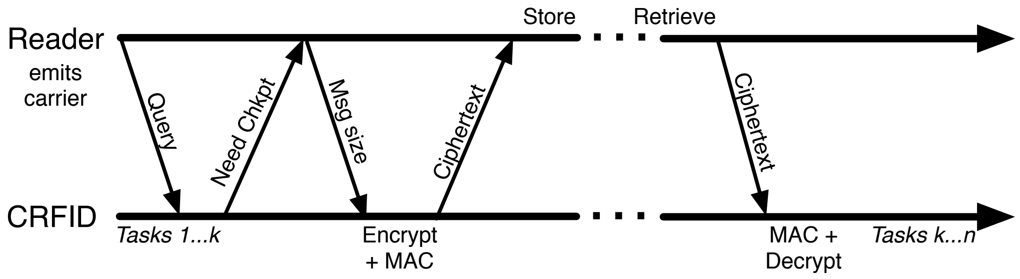

Figure 3: Application-level view of the CCCP protocol. The CRFID sends a request to checkpoint state while in the presence of a reader, and the reader specifies the maximum size of each message. The CRFID then prepares the checkpoint and transmits it in a series of appropriately sized messages. The reader stores the checkpoint data for later retrieval by the CRFID. All messages from the reader to the CRFID also supply power to the CRFID if the latter is within range.

for backscatter modulation on the WISP (Revision 4.0) draws 500 μW of power, far more than is typical of a comparable mechanism on a conventional RFID tag. Alien's Higgs 3, a conventional RFID tag, draws only 15.8 μW of power [2] (total) during operation—an order of magnitude difference that supports an alternative design choice for future CRFIDs.

5.3.1 System Overhead

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

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

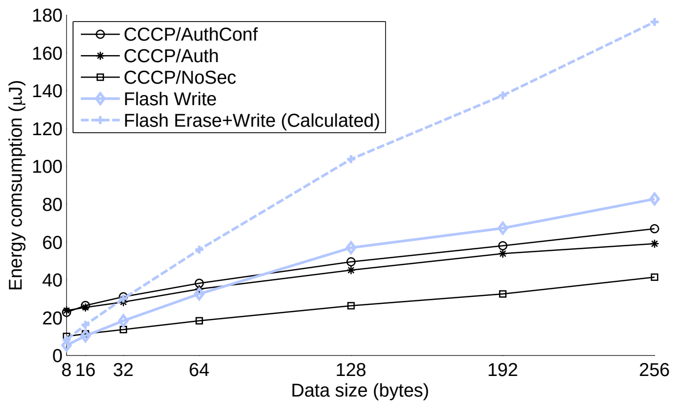

Figure 4: Energy consumption measurements from a WISP (Revision 4.0) prototype for all considered checkpointing strategies. Under our experimental method, we are unable to execute flash writes larger than 256 bytes on current hardware because larger data sizes exhaust the maximum amount of energy available in a single energy lifecycle. The average and maximum percent error of the measurements are 5.85% and 14.08% respectively.

transmitting both. It requires as much keystream material as the size of the state plus a constant amount for authentication.

6 Applications

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

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

RFID sensor networks. Recent work [7] describes *RFID sensor networks* (RSNs) that combine RFID reader infrastructure with sensor-equipped computational RFIDs. RSNs do not simply replace traditional sensor networks because of several limitations. First, they require an infrastructure of readers that provide power to sensor nodes. Second, they are constrained by the distances (several meters) at which CRFIDs currently operate. Third, because RFID communication is asymmetric, the nodes of an RSN cannot exchange information with each other except through a more powerful reader. However, there are applications for which short-range networks of batteryless sensors would be appropriate; Yeager et al. offer several examples [34].

Computational RFIDs as smartcards. Some passive RFID tags are capable of executing strong cryptographic primitives. For example, various models of the Mifare DESfire can perform triple-DES or AES, while other RFID devices can compute elliptic-curve and RSA signatures, such as the RF360 introduced by Texas Instruments [32]. The RF360 is designed to allow public-key authentication in RFID-enabled identification documents, such as e-passports.

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

Some smartcards are capable of performing biometric authentication—generally fingerprint verification. Match-on-card, i.e., verification of the validity of a fingerprint through computation exclusively within the smartcard, has long stood as a technical challenge. The U.S. National Institute of Standards and Technology (NIST) recently conducted an evaluation of a range of such algorithms in contactless cards [11]. CCCP is a promising tool for expanding the class of radio devices for which match-on-card is feasible. While CCCP does not follow a strict match-in-device paradigm—given that it outsources data to a reader—it nonetheless provides comparable security assurances.

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

Trusted platform modules (TPMs) [3, 33] offer support for such outsourcing. A TPM is a hardware device, standard in the CPUs of modern PCs and servers, that can provide a secure attestation to the software configuration of the computing platform on which it operates. Briefly stated, an attestation takes the form of a digital signature on a digest of the software components loaded onto the device. (An attestation does not provide assurance against hardware tampering or subversion of running software.)

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

7 Related Work

CCCP is closely related to Mementos [26] in that both systems provide checkpointing of program execution on CRFIDs. Whereas Mementos relies purely on flash memory and focuses on finding optimal checkpoint frequencies via static and dynamic analysis, CCCP relies primarily on untrusted remote storage via radio and focuses on low-power cryptographic protections to ensure that remotely stored data is as secure as if it were stored locally.

Several systems share CCCP's goal of exploiting properties of RFID systems to enhance security and privacy. For instance, Shamir's SQUASH hash algorithm [31] exploits the underutilized radio link between a tag and a reader to reduce the amount of cryptographic computation necessary on a tag. While number-theoretic hash functions typically require significant computational resources for modular arithmetic, the SQUASH function eliminates costly modular reductions and produces large (unreduced) hash outputs that a tag can send directly to a reader. Tags can thus use the SQUASH function to engage in secure challenge-response protocols with minimal computational resources on the tag. The scheme is provably as one-way as Rabin encryption. Like SQUASH, CCCP exploits the relatively low cost of radio communication between a tag and a reader to increase security. While SQUASH increases radio communication to reduce computation, CCCP increases radio communication to reduce writes to flash memory.

CCCP uses cryptographic techniques from past work on secure file systems and secure content distribution. CFS [5], the SFS read-only file system [16], and Plutus [22] investigated how to provide secure storage layered on various degrees of untrusted infrastructure. The key generation techniques in secure file systems help CCCP to precompute keystream materials during power seasons. While scalability and throughput are the main challenges in such file systems, CCCP primarily addresses energy and memory constraints. The semantics of CCCP storage are similar to the semantics of secure file systems. None of the systems explicitly and directly prevent denial of service. Storing information on untrusted RFID readers trades off the gain in storage capacity and energy conservation versus the risk of losing data due to compromise or destruction of the external storage. To mitigate the risk against denial of service, CCCP could choose to replicate data as do secure file systems.

CCCP shares some goals with power-aware encryption systems such as that proposed by Chandramouli et al. [10]. Both systems are designed to consume little energy while offering the security of well-known cryptographic primitives and both are motivated by a study of power profiling results, but they have different goals. Chandramouli et al. focus on deriving an energy consumption model and establishing a relationship between energy consumption and security, and they offer an encryption scheme that might allow CCCP to consume less energy during its precomputation of keystream bits. However, CCCP's opportunistic precomputation occurs during periods of abundant energy, when the choice of encryption scheme is not of the utmost importance. CCCP's precomputation allows it to use time- and energy-efficient XOR operations at checkpoint time, when energy is low; an alternative encryption scheme would have to save time or energy over simple XOR operations to be useful when energy consumption matters.

CCCP shares a number of properties with systems built for sensor networks. Storage-centric sensor networks [12, 24] have focused on reducing radio communication and increasing writes to flash memory to conserve energy. One of our motivating observations is that this relationship is inverted in the CRFID model: CCCP reduces writes to flash memory in favor of increasing radio communication. Performing cryptography is hard on both a CRFID and its elder cousin the sensor mote. Previous systems, such as SPINS [25] and TinySec [23] for sensor networks, have faced design choices similar to CCCP's. SPINS and TinySec use RC5 because of its small code size and efficiency, but the battery-powered platform underlying these systems differs in fundamental ways from batteryless computational RFIDs. For a side-by-side comparison of such embedded systems, see Table 1 of Chae et al. [9].

CCCP provides secure storage for CRFIDs, and CRFIDs are closely related to existing passively powered RFID tags conforming to the EPC Gen 2 standard [13]. At times the RFID and sensor world fuse together. Buettner et al. [7] propose *RFID sensor networks* (RSNs) as a replacement for wireless sensor networks in applications where batteries are inconvenient, and the authors describe RSNs built on WISP CRFIDs. However, the RSN work does not consider remote storage options for CRFIDs.

8 Future Work

Our future work includes enhancements to the CCCP protocol. Most pressingly, the protocol currently suffers from a potential atomicity problem. In CHECKPOINT (Algorithm 1), *chkpt_counter* is updated before the checkpointed state is transmitted, so that even if the transmission fails, *chkpt_counter* will point to unused keystream material the next time CHECKPOINT runs. However, if CHECKPOINT updates the offset but terminates before transmission succeeds, then the next RESUME operation will see a value of *chkpt_counter* from which its normal backtracking operation will not find the correct keystream material. CCCP cannot currently recover from such a mismatch.

An unacceptable solution is for CHECKPOINT to update *chkpt_counter after* a successful transmission; such a strategy opens the possibility that, if power loss occurred between the transmission and the counter update, CCCP would reuse keystream material. A more reasonable solution (which we have not implemented) is to use a separate *commit bit* that is set in nonvolatile memory after both the *chkpt_counter* update and the transmission; this solution avoids both problems mentioned above. Minimizing the energy cost of maintaining a commit bit is an opportunity for hardware optimization.

A number of implementation enhancements are also future work. For instance, shortfalls in over-the-air RFID protocols and a lack of drivers on the WISP make the restore procedure unnecessarily complicated and difficult to implement. We also plan to extend the borders of CCCP from checkpointing towards long-term storage as described in Section 3.4.5. Key management makes long-term storage more challenging than checkpointing. Another area for further investigation is modifying the checkpoint function to operate at lower voltages. Writing the counter value to flash memory restricts checkpoints to periods where the available energy can support at least one write to flash memory. Our future work seeks to circumvent these minimum voltages in order to accomplish secure remote storage for CRFIDs whenever their processors have sufficient energy to compute. Finally, for simplicity, CCCP's communication protocol currently addresses only the scenario in which a single tag communicates with a single reader. We plan to discard that simplifying assumption during further testing in multi-reader infrastructures.

9 Conclusion

CRFIDs enable pervasive computing in places where batteries are difficult to maintain. However, the high energy necessary to erase and write to flash memory makes storage difficult without a constant energy source. CCCP extends Mementos [26] by exploiting the backscatter transmission common on passive RFID systems to remotely store checkpoints on an untrusted RFID reader infrastructure. CCCP protects data with UHF-based MACs, opportunistic precomputation of keystream material for symmetric cryptography, and hole punching to store a counter used to enforce data freshness. Our measurements of a prototype implementation of CCCP on the WISP tag shows that radio-based, remote checkpoints require less energy than local, flash-based checkpoints—despite the overhead of the cryptography to restore the security semantics of local, trusted storage. CCCP gives a CRFID increased storage capacity at low energy cost and enables long-running computations to make progress despite continual power interruptions that destroy the contents of RAM. Moreover, the abstraction provided by CCCP allows application developers to focus on computation rather than space, energy, and security management. Flash memory generally requires a coarse-grained, high-power erase operation before writing a new value. Our hole punching technique allows CCCP to partially reuse unerased flash memory, thus reducing the frequency with which flash memory must be erased.

10 Acknowledgments

We thank our shepherd Ian Goldberg and the anonymous reviewers for their detailed feedback and discussion; Robert Jackson for guidance on RF power consumption; Berk Sunar and Christof Paar for advice on UHF-based MACs; Mike Todd for help with circuit-level aspects of flash memory; Mankin Yuen for assisting with measurements; and John Brattin for identifying algorithmic flaws. We also thank the members of the SPQR group from UMass Amherst CS and ECE for reviewing early drafts of this work. This research was supported by NSF grants CNS-0520729, CNS-0627529, and the NSF REU program. This research is supported in part by UMass through the CVIP Technology Development Fund. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] AHSON, S. A., AND ILYAS, M., Eds. *RFID Handbook: Applications, Technology, Security, and Privacy*. CRC Press, 2008.

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

[3] BERGER, S., CÁCERES, R., GOLDMAN, K. A., PEREZ, R., SAILER, R., AND VAN DOORN, L. vTPM: virtualizing the trusted platform module. In *Proceedings of the 15th USENIX Security Symposium* (2006), USENIX Association.

[4] BLACK, J., HALEVI, S., KRAWCZYK, H., KROVETZ, T., AND ROGAWAY, P. UMAC: Fast and secure message authentication. In *CRYPTO* (1999), Springer-Verlag, pp. 216–233.

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

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

[7] BUETTNER, M., GREENSTEIN, B., SAMPLE, A., SMITH, J. R., AND WETHERALL, D. Revisiting smart dust with RFID sensor networks. In *Proceedings of the 7th ACM Workshop on Hot Topics in Networks (HotNets-VII)* (October 2008).

[8] CARTER, L., AND WEGMAN, M. Universal hash functions. In *Journal of Computer and System Sciences* (1979), Elsevier, pp. 143–154.

[9] CHAE, H.-J., YEAGER, D. J., SMITH, J. R., AND FU, K. Maximalist cryptography and computation on the WISP UHF RFID tag. In *Proceedings of the Conference on RFID Security* (July 2007).

[10] CHANDRAMOULI, R., BAPATLA, S., SUBBALAKSHMI, K. P., AND UMA, R. N. Battery power-aware encryption. *ACM Trans. Inf. Syst. Secur. 9*, 2 (2006), 162–180.

[11] COOPER, D., DANG, H., LEE, P., MACGREGOR, W., AND MEHTA, K. *Secure Biometric Match-on-Card Feasibility Report*, 2007.

[12] DIAO, Y., GANESAN, D., MATHUR, G., AND SHENOY, P. Rethinking data management for storage-centric sensor networks. In *Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR)* (January 2007).

[13] EPCGLOBAL. EPC Radio-Frequency Identity Protocols, Class-1 Generation-2 UHF RFID. http://www.epcglobalinc.org/standards/uhfc1g2/, 2008.

[14] ETZEL, M., PATEL, S., AND RAMZAN, Z. Square hash: Fast message authentication via optimized universal hash functions. In *In Proc. CRYPTO 99, Lecture Notes in Computer Science* (1999), Springer-Verlag, pp. 234–251.

[15] FONSECA, R., DUTTA, P., LEVIS, P., AND STOICA, I. Quanto: Tracking energy in networked embedded systems. In *8th USENIX Symposium of Operating Systems Design and Implementation (OSDI'08)* (2008), pp. 323–328.

[16] FU, K., KAASHOEK, M. F., AND MAZIÈRES, D. Fast and secure distributed read-only file system. *ACM Transactions on Computer Systems 20*, 1 (February 2002), 1–24.

[17] GUTTERMAN, Z., PINKAS, B., AND REINMAN, T. Analysis of the Linux random number generator. In *IEEE Symposium on Security and Privacy* (2006), IEEE Computer Society, pp. 371–385.

[18] HADDAD, S., CHANG, C., SWAMINATHAN, B., AND LIEN, J. Degradations due to hole trapping in flash memory cells. In *Electron Device Letters* (March 1989), IEEE, pp. 117–119.

[19] HALPERIN, D., HEYDT-BENJAMIN, T. S., RANSFORD, B., CLARK, S. S., DEFEND, B., MORGAN, W., FU, K., KOHNO, T., AND MAISEL, W. H. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy* (May 2008), IEEE Computer Society, pp. 129–142.

[20] HOMER. *Odyssey*, vol. XI. ca. 750 B.C.

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

[22] KALLAHALLA, M., RIEDEL, E., SWAMINATHAN, R., WANG, Q., AND FU, K. Plutus: Scalable secure file sharing on untrusted storage. In *Proc. USENIX Conference on File and Storage Technologies* (San Francisco, CA, December 2003).

[23] KARLOF, C., SASTRY, N., AND WAGNER, D. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)* (November 2004).

[24] MATHUR, G., DESNOYERS, P., GANESAN, D., AND SHENOY, P. CAPSULE: An energy-optimized object storage system for memory-constrained sensor devices. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys)* (November 2006).

[25] PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, J. D. SPINS: Security protocols for sensor networks. *Wireless Networks 8*, 5 (Sept. 2002), 521–534.

[26] RANSFORD, B., CLARK, S., SALAJEGHEH, M., AND FU, K. Getting things done on computational RFIDs with energy-aware checkpointing and voltage-aware scheduling. In *Proceedings of USENIX Workshop on Power Aware Computing and Systems (HotPower)* (December 2008).

[27] RANSFORD, B., AND FU, K. Mementos: A secure platform for batteryless pervasive computing, August 2008. USENIX Security Works-in-Progress Presentation.

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

[29] SAMPLE, A. P., YEAGER, D. J., POWLEDGE, P. S., MAMISHEV, A. V., AND SMITH, J. R. Design of an RFID-based battery-free programmable sensing platform. In *IEEE Transactions on Instrumentation and Measurement* (2008).

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

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

[32] TEXAS INSTRUMENTS INCORPORATED. http://www.ti.com/rfid/shtml/news-releases-11-12-07.shtml.

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

[34] YEAGER, D., POWLEDGE, P., PRASAD, R., WETHERALL, D., AND SMITH, J. Wirelessly-Charged UHF Tags for Sensor Data Collection. In *IEEE International Conference on RFID* (2008), pp. 320–327.