A Platform for RFID Security and Privacy Administration

Melanie R. Rieback* Georgi N. Gaydadjiev**
Bruno Crispo, Rutger F.H. Hofman, Andrew S. Tanenbaum***
*Department of Computer Science Vrije University
**Department of Computer Engineering Delft University of Technology
***Department of Computer Science Vrije University, Amsterdam fcrispo, rutger,

Abstract

This paper presents the design, implementation, and evaluation of the RFID Guardian, the first-ever unified platform for RFID security and privacy administration.

The RFID Guardian resembles an "RFID firewall", enabling individuals to monitor and control access to their RFID tags by combining a standard-issue RFID reader with unique RFID tag emulation capabilities.

Our system provides a platform for coordinated usage of RFID security mechanisms, offering fine-grained control over RFID-based auditing, key management, access control, and authentication capabilities. We have prototyped the RFID Guardian using off-the-shelf components, and our experience has shown that active mobile devices are a valuable tool for managing the security of RFID tags in a variety of applications, including protecting low-cost tags that are unable to regulate their own usage.

1. Introduction

Radio Frequency Identification (RFID) tags are remotely-powered computer chips that augment everyday objects with computing capabilities. Corporate executives tout RFID technology as a technological means to achieve cost savings, efficiency gains, and unprecedented visibility into the supply chain. Scientific researchers consider RFID technology as nothing short than an embodiment of the paradigm shift towards low-cost ubiquitous computing. In both cases, RFID tags will blur the boundaries between the online and physical worlds, allowing individuals to manage hundreds of wirelessly interconnected real-world objects, like dendrites in a global digital nervous system. RFID tags may be the size of a grain of rice (or smaller), and have built-in logic (microcontroller or state machine), a coupling element (analog front end with antenna), and memory (pre-masked or EEPROM). Passive tags are powered entirely by their reading devices, while active tags contain auxiliary batteries on board. Passive LF tags (125-135 kHz) can be read up to 30 cm away, HF tags (13.56 MHz) up to 1 m away, UHF tags (2.45GHz) up to 7 m away, and active tags up to 100 m away or more.

Figure 1: Philips I.Code RFID Tags

1.1 RFID Applications and Threats

RFID automation will bring an unfathomable barrage of new applications, forever banishing wires, grocery store cashiers, credit cards, and pocket change from our lives. RFID proponents extol its professional uses for real-time asset management and supply chain management. RFIDbased access passes help to police residential, commercial, and national borders; drivers have embraced RFIDbased retail systems like EZ-Pass, FastPass, IPass, Pay- Pass, and SpeedPass. RFID-based "feel good" personal applications are also proliferating, from "smart" dishwashers, to interactive children's toys, to domestic assistance facilities for the elderly. RFID tags identify lost housepets, and even keep tabs on people; the data carriers have assisted with surgeries, prevented the abduction of infants, and tracked

teenagers on their way to school. Subdermal Verichips are hip accessories for patrons of several European nightclubs, and have been less glamorously deployed for identifying deceased victims of hurricane Katrina [1].

RFID technology thus races on at a pace that surpasses our ability to control it. The same ease-of-use and pervasiveness that makes RFID technology so revolutionary offers less-then-ethical characters unprecedented opportunities for theft, covert tracking, and behavioral profiling. Without the appropriate controls, attackers can perform unauthorized tag reading and clandestine location tracking of people or objects (by correlating RFID tag "sightings"). Snooping is possible by eavesdropping on tag/reader communications. Criminals can also manipulate RFID-based systems (i.e. retail checkout systems) by either cloning RFID tags, modifying existing tag data, or by preventing RFID tags from being read in the first place.

Security and privacy researchers have proposed a wide array of countermeasures against these threats. The simplest solution is deactivating RFID tags; permanently (via "frying"[17], "clipping"[13], or "killing"[4]), or temporarily (using Faraday cages or sleep/wake modes[20]). Cryptographers have created new lowpower algorithms for RFID tags, including stream ciphers [6], block ciphers[5], public-key cryptographic primitives[9], and lightweight protocols for authentication [21]. Additionally, researchers have developed access control mechanisms that are located either on tag (hash locks[22] / pseudonyms[10]) or off (Blocker Tag[11], RFID Enhancer Proxy[12]).

Despite this plethora of countermeasures, neither the threats nor the fears facing RFID have dissipated. The countermeasures have become somewhat of a band-aid that can be slapped onto RFID technology later. Some companies view these results as a desirable way to quiet down the privacy activists. Other companies in RFID standardization committees are actively fighting *against* adding security into RFID protocol design, because it will make their current commercial offerings obsolete. People need a solution that they can physically own and use, not one that relies upon the RFID companies to decide when privacy will become important.

Another missing element is a means to coordinate the myriad of incompatible countermeasures as they trickle onto the market in a piecemeal fashion. Per-tag security policies combined with a lack of automation will form a management nightmare for people, who cannot be expected to know when or how to apply the appropriate countermeasures. There is no unified framework; no systematic means to leverage individual RFID countermeasures to achieve the most important goal of all – the protection of real people.

1.2 RFID Guardian Design Goals

Over the past months, we have designed and prototyped the RFID Guardian, a system that allows people to administer the security of their RFID tags. The design of the RFID Guardian was driven by the following goals, which follow from the nature of RFID applications and deployment considerations:

- **Centralized use and management.**

Most existing RFID countermeasures distribute their security policies across RFID tags, which make them very hard to configure, manage, and use. To address this concern, we designed a single platform to leverage RFID countermeasures in a coordinated fashion. Personalized security policies are centrally enforced by utilizing novel RFID security features (auditing, automatic key managment, tag-reader mediation, off-tag authentication) together with existing ones (kill commands, sleep/wake modes, on-tag cryptography).

- **Context-awareness.**

Different countermeasures have strengths and weaknesses in different application scenarios. Lowcost Electronic Product Code (EPC) tags require different access control mechanisms than expensive crypto-enabled contactless smart cards. Our system maintains both RFID-related context (i.e.RFID tags present, properties and security features, and their ownership status), as well as personal context (i.e. the user is in a non-hostile environment). Context is then used in conjunction with an Access Control List (ACL) to decide how to best protect the RFID tags in question.

- **Ease-of-use.**

People do not want to fuss with an RFID privacy device, so our system must be both physically and operationally unobtrusive. We envision that our system will be eventually integrated into a PDA or mobile phone, so users will not be burdened with carrying an extra physical device. Accordingly, the RFID Guardian uses an XScale processor and simple RFID HW (barely more complex than RFID HW already found in Nokia mobile phones). Also, system operation was designed to be non-interactive for default situations, and offers a user interface for the special cases that require on-site configuration.

- **Real-world useability.**

It is essential that the RFID Guardian work with actual
deployed RFID systems. We chose a single standard as

a proof-of-concept, to prove the technical feasability our ideas. Our RFID Guardian implementation supports 13.56 MHz (HF) RFID, and is compatible with the ISO-15693[2] standard. This frequency and standard is used in a wide array of RFID applications, due to the availablity of relatively inexpensive commodity HW. The ideas in this paper can also be extended to other standards or frequencies, given some extra engineering effort.

The remainder of this paper is organized as follows. Section 2 describes the RFID Guardian's high-level functionality. Section 3 provides implementation details for our RFID Guardian prototype, and Section 4 provides a real-life case study, illustrating the operation of Selective RFID Jamming. Performance results are reported in Section 5. Section 6 presents a discussion of potential attacks, and Section 7 reviews some related work. Our discussion is then concluded in Section 8.

2. System functionality

The RFID Guardian (first introduced in [19]) is a portable battery-powered device that mediates interactions between RFID readers and RFID tags. The RFID Guardian leverages an on-board RFID reader combined with novel tag emulation capabilities to audit and control RFID activity, thus enforcing conformance to a centralized ecurity policy.

The vast majority of RFID readers will not explicitly interact with the RFID Guardian. Eavesdropping and clever tag emulation tactics are necessary to glean information from these readers. However, a small group of RFID readers will have special back-end SW installed, that provides them with an "awareness" of the Guardian.

These RFID readers tend to be in familiar locations (i.e.at home, at the office), and they are intentionally granted more generous access permissions. These RFID readers may explicitly cooperate with the Guardian, sending data containing authentication messages, context updates, or secret keys.

The rest of this section describes the design of the RFID Guardian, focusing on four fundamental issues: (i) auditing, (ii) key management, (iii) access control, and (iv) authentication.

2.1 Auditing

The RFID Guardian monitors RFID scans and tags in its vicinity, serving as a barometer of (unauthorized) RFID activity. RFID auditing is a prerequisite for the enforcement of RFID security policies, plus it furnishes individuals with both the awareness and proof needed to take legal recourse against perpetrators of RFID abuse.

2.1.1 Scan logging

Scan logging audits RFID scans in the vicinity, which are either displayed (using an LCD or screen) or are logged for later retrieval. Tag emulation decodes the RFID reader queries prior to logging the 64-bit UID (tag ID), an 8-bit command code, and annotations (like a 32-bit timestamp). Query data is logged by default, unless the flash memory is almost full.

Audited RFID scans should be filtered to avoid overwhelming the user with uninteresting information. For example, the RFID Guardian might be configured to only log scans targeting tags "owned" by that individual (see next section). Repeatedly polled queries (like inventory queries, which ask tags in range to identify themselves) will also generate a lot of noise, so it is best to have the SW aggregate these queries (e.g. 1000x inventory query from time t1-t2).

2.1.2 Tag logging

The RFID Guardian tracks RFID tag ownership and alerts individuals of newly appearing (possibly clandestine) tags. Ownership of RFID tags can be transferred explicitly via the user interface or an authenticated RFID channel (i.e.while purchasing tagged items at an RFID enabled checkout). Ownership of RFID tags can also be transferred implicitly (i.e. when handing an RFID tagged book to a friend.) The RFID Guardian detects implicit tag acquisition by conducting periodic RFID scans, and then correlating the tags that remain constant across time.

The frequency of RFID tag discovery is adjustable. Given that not all implicit tag acquisitions are desirable, The frequency of scanning/correlation/reporting presents a tradeoff between privacy, accuracy, and battery life. Our opinion is that infrequent correlation in a controlled environment is probably the most useful and least error prone option (i.e comparing RFID tags present at home at the beginning and end of the day).

2.2 Key Management

Modern RFID tags have a variety of security functionality, ranging from tag deactivation commands, to password-protected memory, to industrial-grade cryptography. These security features often require the use of associated key values, which present logistical issues because the keys must be acquired, stored, and available for use at the appropriate times.

The RFID Guardian is well suited to manage RFID

tag keys due to its 2-way RFID communications abilities. Tag key transfer could occur by eavesdropping on the RFID channel when a reader (for example, an RFID tag "deactivation station") issues a query containing the desired key information. Additionally, "Guardian aware" RFID readers can transfer key information explicitly over a secure channel, or key values can be manually entered via the user interface. The RFID Guardian is also an appropriate medium for periodically regenerating tag keys, re-encrypting tag data[8], and refreshing tag pseudonym lists[10].

2.3 Access Control

RFID technologists and privacy activists propose deactivating RFID tags after sale as a means of protecting consumer privacy (and corporate liability). However, if you consider that RFID tags represent the future of computing technology, this proposal becomes as absurd as permanently deactivating desktop PCs to reduce the incidence of computer viruses and phishing. Perhaps RFID tags are in fact too much like modern computers– their default behavior is to indiscriminately transfer data to anyone with compatible equipment. The hope is that modern security technologies like firewalls and proxies can be adapted, to protect hapless RFID tags from themselves via central monitoring and managing of the communications medium.

2.3.1 Coordination of security primitives

The RFID Guardian maintains a centralized security policy that dictates which RFID readers have access to which RFID tags in which situations. This security policy is implemented as an Access Control List (ACL). The ACL resembles one used by a standard packet filter, that allows or denies RFID traffic based upon the querying reader (if known), the targeted tag(s), the attempted command, and the context (if any).

Permitted data types in the ACL are values (i.e.123), text strings (i.e. 'at home', 'in a paranoid mood'), groupings (i.e. assigned groups of tags/readers/context/commands), and wildcards (123*,*). The user configures the ACL, and constructs the groups via the user interface.

2.3.2 Context-awareness

Different situations call for different countermeasures. For example, RFID tagged credit cards require less stringent security at home than at the shopping mall. The RFID Guardian therefore offers context awareness facilities that perceive an individual's situation and then regulate tag access accordingly.

Well defined context like dates and times are easy to infer, but are marginally useful for describing a person's situation, moods, or desires. Alternately, more abstract context information can be represented via "context updates", which are arbitrary textual strings that represent some facet of the user's situation. Context updates could report anything. For example, an RFID reader at the front door of a person's home might inform the RFID Guardian that it is now leaving a protected area. Context updates are provided either by user (via the user interface), or by authenticated "Guardian aware" RFID readers.

2.3.3 Tag-reader mediation

The RFID Guardian acts as a mediator between RFID readers and RFID tags. Just like a packet filter, the Guardian uses Selective RFID Jamming[18] to enforce access control by controlling the communications mediation. The RFID Guardian can therefore control access for low-cost RFID tags that otherwise might not have any access control primitives available to them.

The RFID Guardian's selective jamming scheme is currently optimized for ISO-15693 tags, which use the Slotted Aloha anticollision scheme (as opposed to EPCglobal's 'tree-walking'). Selective RFID Jamming uses tag emulation to decode the incoming RFID reader query, determines if the query is permitted (according to the ACL), and then sends a short jamming signal that precisely blocks the timeslot in which the "protected" RFID tag will give its response.

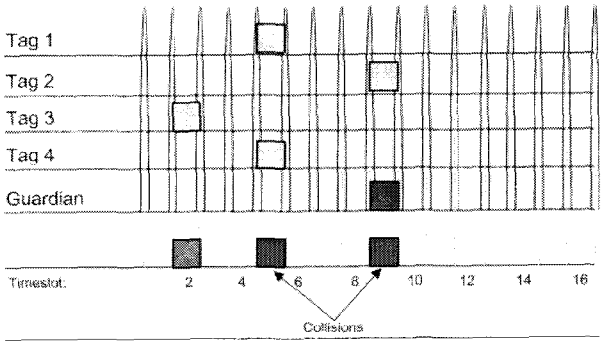

Figure 2: Selectively Jamming Tag # 2

There are 16 timeslots after an inventory query, so during the first round of anticollision, the jamming has a 1 in 16 chance of accidentally interfering any other RFID tag present. During each subsequent round of anticollision, the reader issues another inventory query with a slightly modified mask value, that targets a slightly narrower range of RFID tags than before. Given enough rounds of anticollision, the mask value

will exclude the RFID tag(s) that are being "protected", allowing other tags in the vicinity to get their responses heard by the RFID reader. This means that in practice, our system has a negligible chance of blocking the incorrect RFID tag responses. This makes the RFID Guardian's manner of selectively jamming inventory queries far less-obtrusive than the Blocker Tag's concept of "privacy zones"[11], which block entire ranges of tag identifiers (regardless of who owns the tag.)

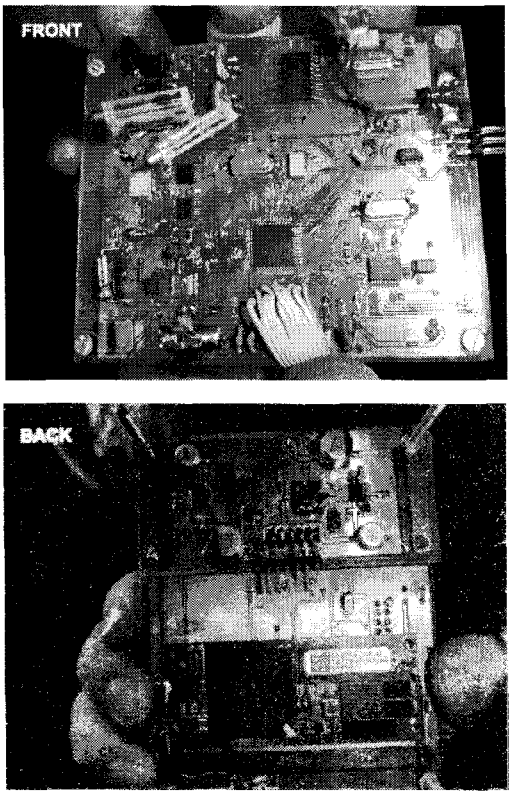

Figure 3: RFID Guardian Prototype

2.4 Authentication

Some high-cost RFID tags can directly authenticate RFID readers, but the majority of RFID tags cannot due to application constraints (i.e. cost or power). The RFID Guardian thus authenticates "Guardian aware" RFID readers on behalf of low-cost RFID tags, adapting the subsequent access control decisions to reflect the permissions of the newly-identified reader. Prior to authentication, the RFID Guardian must also exchange authentication keys with RFID Readers, either ahead of time or using on-the-fly means (ex. user interface, PKI).

After the successful authentication of a reader, the RFID Guardian faces a practical problem: for noncryptographic RFID tags there is no easy way to determine which RFID queries originate from which RFID reader. The best solution would be for RFID standardization committees to add space for authentication information to the RFID air interface. However, until that happens, we are using our own imperfect solution: in the last step of authentication an RFID reader announces which queries it's going to perform, and these queries are noted as part of an "authenticated session" when they occur.

3 . Implementation

The RFID Guardian prototype, shown in Figure 3, is meant to help people solve their RFID privacy problems in a practical way. Therefore, we have tested our system against commonly used RFID equipment – the Philips MIFARE/I.Code Pagoda RFID Reader, with Philips I.Code SLI (ISO-15693) RFID tags. This section will introduce the hardware and software architecture that our prototype uses to monitor and protect the RFID infrastructure.

3.1 Hardware

The RFID Guardian hardware architecture is presented
in Figure 4.

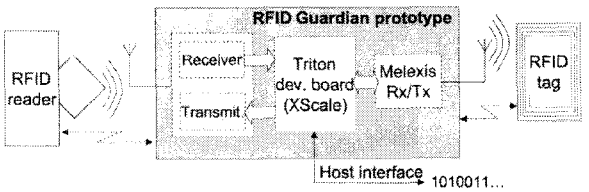

Figure 4: RFID Guardian HW Architecture

Our first salient design decision was to make the RFID Guardian a full-fledged portable computer. We chose a "beast" of a microcontroller – the Intel XScale PXA270 processor, with 64 megabytes of SDRAM and 16 megabytes of Flash memory. We rationalized the use of the XScale by the strict ISO-15693 timing constraints combined with the computational load of authenticating RFID readers. (Section 5 analyzes the extent to which the PXA270 is overkill.) Another benefit of the XScale processor family is its wide deployment in handheld devices, which eases eventual integration of the RFID Guardian into PDAs and mobile phones.

Our prototype has a minimalist User Interface (UI) at the moment – a serial RS-232 interface to the PC host, which contains an attached keyboard and screen. While

this is sufficient for our proof-of-concept, we plan to add a more portable UI to the next version of the RFID Guardian HW.

3.1.1 RF Design Overview

The analog part of our prototype consists of an "RFID reader" front end that uses an RFID reader-on-a-chip, and an "RFID tag" front end which required building our own custom tag emulation HW.

Our *reader transmitter/receiver* was implemented using an ISO-15693 compliant RFID reader IC from Melexis (MLX90121)[16] together with a power stage, based on the application note AN90121 1 [15], that increases the operating range to 30 cm.

Our *tag receiver* is based on an SA605 IC from Philips. The IC is intended for a single chip FM radio, but we used it to implement a high sensitivity AM receiver. Because our receiver is battery powered (as opposed to passively-powered RFID tags), it receives RFID reader signals up to a half meter away.

Our *tag transmitter* implements "active" tag spoofing using an RF power stage and a dedicated digital part that generates and mixes the required sideband frequencies, 13.56 MHz +/- 423 kHz. By actively generating the sideband frequencies, we can transmit fake tag responses up to a half meter.

We also use our tag transmitter as the basic HW primitive to generate the RFID Guardian's randomized jamming signal. (This is described further in the SW section.)

3.1.2 Tag Spoofing Demystified

RFID readers produce an electromagnetic field that powers up RFID tags, and provides them with a reference signal (e.g. 13.56 MHz) that they can use for internal timing purposes. Once an RFID tag decodes a query from an RFID reader (using its internal circuitry), it encodes its response by turning on and off a resistor in synchronization with the reader's clock signal. This so-called "load modulation" of the carrier signal results in two sidebands, which are tiny peaks of radio energy, just higher and lower than the carrier frequency. Tag response information is transmitted solely in these sidebands2, rather than in the carrier signal.

Figure 5 (from the RFID Handbook[6]) illustrates how these sidebands look, in relation to the reader-generated

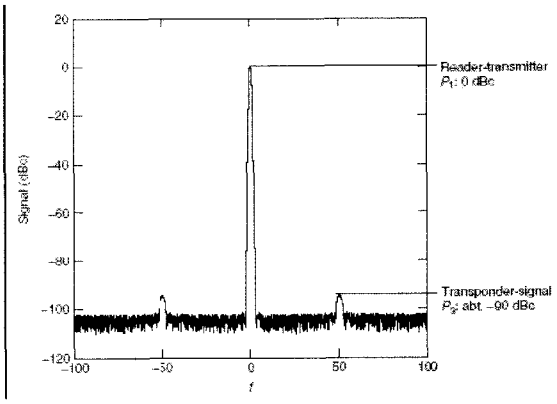

Figure 5: Normal RFID Tag Signal

carrier frequency. The comparatively tiny sidebands have approximately 90 decibels less power than the reader-generated carrier signal, and this is the reason why RFID tag responses often have such a limited transmission range.

The secret to creating fake tag responses is to generate the two sideband frequencies, and use them to send back properly-encoded responses, that are synchronized with the RFID reader's clock signal. The simplest way to generate these sidebands is to imitate an RFID tag, by turning on and off a load resistor with the correct timing. The disadvantage of this approach is that passive modulation of the reader signal will saddle our fake tag response with identical range limitations as real RFID tags (~10 cm for our test setup).

A superior alternative is to use battery power to generate the two sideband frequencies. These super-powerful sidebands are detectable at far greater distances, thus increasing the transmission range of our fake tag response.

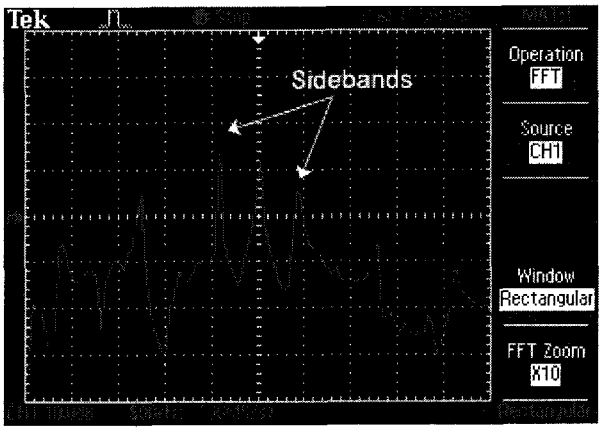

Figure 6: Spoofed RFID Tag Signal

The RFID Guardian prototype utilizes the "active"

tag spoofing approach. Figure 6 shows the signal generated by our tag transmitter. The spoofed "sidebands" are transmitted at a power-level roughly equal to the reader's carrier signal. This has increased the range of our fake tag responses – from 10 cm to a half meter away!

3.2 Software

The RFID Guardian is like a watchdog; it sits with a cocked-ear, waiting for danger to appear. It monitors real-world activity, from unexpected RFID scans to clandestinely located tags, and reacts in real-time lest these dangers remain undetected and undeterred.

The RFID Guardian's SW architecture reflects this event-driven reality. Besides its real-time core, the Guardian's 12694 lines of code provide device drivers (for our RFID HW), a protocol stack (ISO-15693), data storage libraries, high-level system tasks, and application libraries. The result is 254728 bytes of cross-compiled functionality dedicated to RFID security and privacy protection.

3.2.1 Operating System

The RFID Guardian presents a holistic system to users, but lurking below the surface are time-critical SW routines that require central coordination. The e-Cos Real- Time Operating System (RTOS) takes the place of taskmaster; it ensures fast and reliable execution, while simplifying developers' lives by handling threads, basic common interrupt handling, and some device drivers (i.e. RS-232 driver). e-Cos was selected primarily for its availability for the PXA270 microcontroller, but it also proved an excellent choice because it is open-source, free of licensing costs, and has an active developer community.

3.2.2 Libraries

A major portion of the RFID Guardian SW handles intermediate processing steps; e.g. tag spoofing requires ISO-compliant frame modulation and encoding, and scan logging requires a mechanism for caching data in the Flash memory. This section will describe the lowand medium-level libraries that support the main RFID Guardian functionality.

Device Drivers Device drivers are the steering oftware for the RFID Guardian's HW. Driver pairs control the RFID tag device (tag transmitter/receiver), RFID reader device (reader transmitter/receiver), and the jamming signal (random noise generated by the tag transmitter). Device drivers can read/write bytes and RFID markers (EOF, SOF, JAM), and they can also provide timing information. eCos also conveniently provides device drivers for the RS-232 "user interface", which facilitates a connection to the user's keyboard and screen.

Protocol Stack Once the device drivers decode bytes of raw RFID data, the RFID Guardian needs to make further sense out of it; e.g. was it an RFID tag replying to an inventory query, or an RFID reader attempting to read a data block? The ability to understand RFID communications protocols is a prerequisite for making meaningful high-level security decisions (e.g. was the reader's read command authorized?) This is why the RFID Guardian contains an implementation of Part 2 (device drivers) and Part 3 (Communications protocol) of the ISO-15693 standard.

Data Storage Once RFID communications have been interpreted, the internal state of the RFID Guardian is updated by modifying the contents of one or more data structures. Generally, this data is stored in the volatile RAM, but "permanent" data structures are cached into Flash when the processor is idle. The Journaling Flash File System (v2) manages the RFID Guardian's Flash memory, providing filesystem-style access, offline garbage collection, balanced erasing of blocks, and crash resistance.

The data structures themselves collectively reflect the high-level functionality of the RFID Guardian. Transient data structures include the tag presence list, partially-open authentication list, authenticated session list, context list, and timer activity list. Permanent data structures may also include the RFID scan log, access control list, reader authentication key list, tag ownership list, and tag key list.

3.2.3 Tasks

The RFID Guardian's high-level system tasks are little virtual pieces of functionality that take turns controlling the behavior of the system. Each task plays a different role: the tag task acts like a virtual RFID tag, and the reader task like a commodity RFID reader. The timer task is akin to a little alarm clock, that periodically goes off and spurs other system components into action. The user input task primarily relays input from the real-life user input devices to the appropriate SW handler.

Each of these tasks uses a comparable software stack . A main loop at the top level waits for activity on any device, and an interrupt prompts the device driver to decode and store the frame(s). The task then invokes the appropriate high-level application routines.

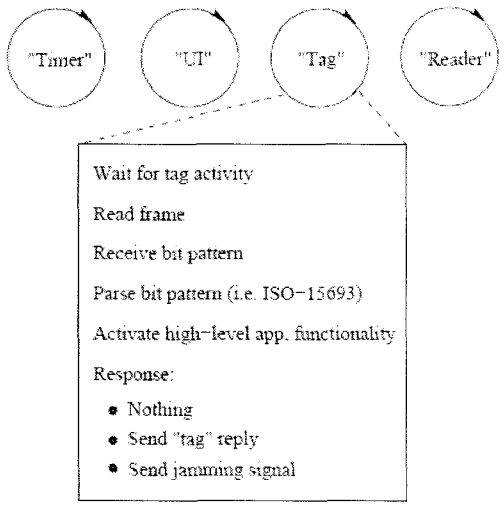

Figure 7: "Tag" Task Functionality

Timer Task The RFID Guardian needs to perform activities at specific times, either periodically (i.e. polling to populate the RFID tag presence list), or on a one-time basis (i.e. timing out a half-opened authentication attempt). The timer task is responsible for keeping track of scheduled activities, and multiplexing the XScale's high-resolution timer interrupts with the corresponding actions that must occur at those times.

User Input Task On rare occasions, users will want to explicitly interact with the RFID Guardian. They may want to configure the ACL, conduct an RFID scan, provide context data, or execute some other kind of system command. The user input task collects these commands from the cornucopia of available input devices, (i.e. RS- 232, keyboard/button/keypad/etc..), and reroutes them to the system components responsible for the desired highlevel functionality.

Tag Task Tag emulation is one of the highlights of the RFID Guardian, being frequently used to achieve the RFID Guardian's high-level goals – RFID scan logging, authentic ating RFID readers, and spoofing one or several RFID tags. The tag task is the entity responsible for coordinating the RFID Guardian's "tag-like" behavior. When activated by an interrupt from the tag receiver, the task calls the device driver to demodulate and decode the incoming RFID queries. This subsequently activates the aforementioned high-level functionality, if needed.

Reader Task The reader task, driven by SW requests from the timer and UI, coordinates use of the Guardian's RFID reader-on-a-chip. The task performs specified queries, (i.e inventory, read/write data), and interprets the tag responses. This is commonly used for detecting (possibly covert) RFID tags, and activating on-tag security mechanisms, if any.

3.2.4 Inter-Device Functionality

Lots of high-level application functionality has been introduced in this paper, but little has been said about the RFID Guardian's interactions with "Guardian aware" RFID infrastructure (introduced in section 2).

RFID Guardian-Reader communications use a metalanguage that we call Guardian Language (GL), which is encapsulated in standard ISO-compliant 'read/write multiple blocks' commands. GL uses an 8-bit Distinctive Starting Block, an 8-bit GL Command and a varying amount of Command Data. The theoretical length limit for command data is 8 kBytes, although the practical limit is 128 bytes, which is the capacity of our I.Code SLI tags.

Here is how GL looks when encapsulated a 'read multiple block' response:

SOF	Flags	DSB	GLC	Command Date	CPC16	EOF
	8bits	8bits	8bits	256bits-64bits	16bits	

Here is a non-exhaustive list of GL commands: Initiate Authentication, Authentication Response, Key Update, Forward Query (proxy mode), Add Tag, Remove Tag, Add Reader, Remove Reader, and Context Update. GL also features non-standard configuration commands, that require some knowledge about the RFID Guardian internal setup.

One caviat is that, because the RFID Guardian is emulating an RFID tag, Guardian-Reader communications are constrained by master-slave interactions. In other words, RFID readers must always initiate communications with the RFID Guardian. Designers must keep this in mind when creating interaction patterns for new RFID security and privacy functionality.

4. Case Study: Selective RFID Jamming

This section will provide a step-by-step demonstration of how Selective RFID Jamming works.

For demonstration purposes, we have given the RFID Guardian a minimal tag ownership list that contains only one tag (UID: 0xe0040100003b0cbd). A single entry in an equally minimal ACL prescribes blocking all tags in the ownership list:

We now generate inventory queries with our Philips

MIFARE/I.Code Pagoda RFID Reader, which is driven from a Windows PC interface. Initially the RFID Guardian is switched off, and the Philips Reader detects three tags in its vicinity: the one tag that is

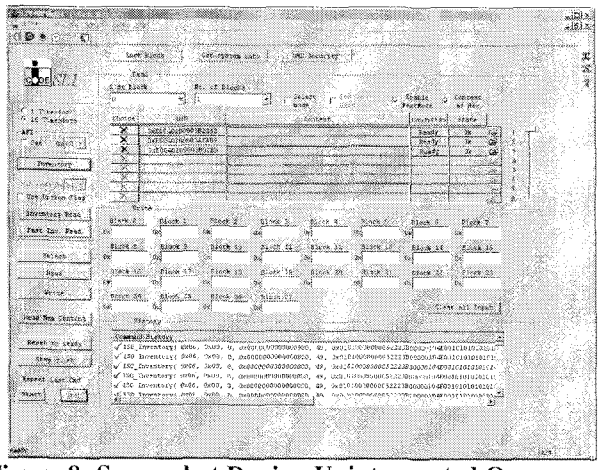

Figure 8: Screenshot During Uninterrupted Query

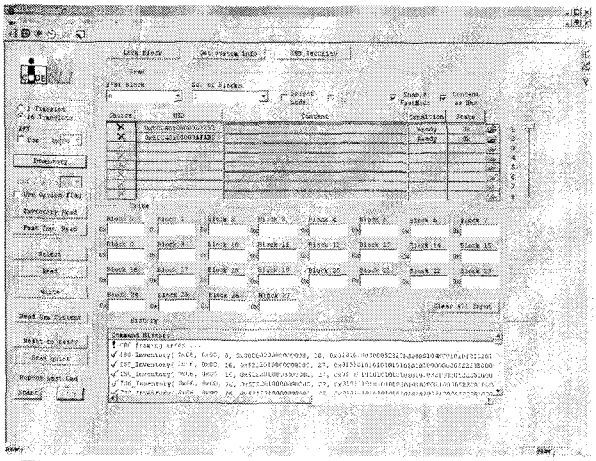

Figure 9: Screenshot During Selective RFID Jamming

Tag	Reader	Command	Comtext
...	...	...	...
<ownership list>	*	*	*

in our ownership list, and two unknown tags (UID: 0xe0040100003b2252 and 0xe0040100003afab9). (See Figure 8 for a screenshot.)

When the RFID Guardian is enabled, the Philips Reader's inventory queries are immediately detected. These requests are decoded, and the RFID Guardian's internal logic determines that the query should be blocked. The Guardian then sends a short (ca. 350/sec) jamming signal at timeslot 13 of the inventory sequence, since that slot corresponds to the protected tag: 0xe0040100003b0cbd.

Only the two unprotected tags are recognized by the Philips reader now, and the jamming caused a CRC error that is reported in the lower central pane of the reader's user interface (see Figure 9).

Debug output from the RFID Guardian illustrates the processing steps, including the decision to jam at timeslot 13:

```
1  Request t_eof 76.877230 RFID_INVENTORY(
1a      flags=RFID_FRAME_DATA_RATE_FLAG|
1b            RFID_FRAME_INVENTORY_FLAG),
1c   masklen=0x00,mask=0x0;
2  Inventory: t_eof 76.877230 s->SN 0 s->NbS 16
3  Inventory: t_eof 76.882010 s->SN 1 s->NbS 16
4  Inventory: t_eof 76.886791 s->SN 2 s->NbS 16
5  Inventory: t_eof 76.888304 s->SN 3 s->NbS 16
6  Inventory: t_eof 76.891568 s->SN 4 s->NbS 16
7  Inventory: t_eof 76.896340 s->SN 5 s->NbS 16
8  Inventory: t_eof 76.901120 s->SN 6 s->NbS 16
9  Inventory: t_eof 76.905893 s->SN 7 s->NbS 16
10 Inventory: t_eof 76.910673 s->SN 8 s->NbS 16
11 Inventory: t_eof 76.915446 s->SN 9 s->NbS 16
12 Inventory: t_eof 76.920225 s->SN 10 s->NbS 16
13 Inventory: t_eof 76.924999 s->SN 11 s->NbS 16
14 Inventory: t_eof 76.929778 s->SN 12 s->NbS 16
15 Inventory: t_eof 76.934552 s->SN 13 s->NbS 16
16 Inventory JAM t 76.934869 on s->SN 13 s->NbS 16
16a    mask len 0 mask 0x0
17 Inventory: t_eof 76.939330 s->SN 14 s->NbS 16
18 Inventory: t_eof 76.944107 s->SN 15 s->NbS 16
```

Lines 1-1c report an Inventory request with a mask length 0, and flags indicating a 16-slot inventory sequence. Lines 2 through 18 report End of Frame (EOF) pulses that mark the start of a new timeslot. (s->SN indicates the current slot number.) Line 16-16a corresponds with timeslot 13, and it indicates the generation of a jamming signal.

5 .Performance Measurements

This section will analyze the performance of the RFID Guardian, under a variety of resource constraints and attack modes.

5.1 Timing Constraints

The RFID Guardian enforces access control decisions on the behalf of RFID tags, so real-time performance is required under both normal and hostile conditions. After all, blocking a tag response *after* it has reached the attacker is not very useful.

In the upper time-line of Figure 10 we show the timing constraints for an inventory request-response sequence as specified by the ISO standard. Like every other RFID message, the request is framed by a start-of-frame marker (SOF) and an end-of-frame marker (EOF). Between these markers, an inventory request carries between 40 (mask size is 0) and 104 (mask size

is 64) data bits. After receiving the request EOF, the tag must wait for 320.9 'sec before starting its answer. This is the time the RFID Guardian has to interpret reader requests and respond to them.

The lower time-line of Figure 10 shows the measured performance of the RFID Guardian. After a complete frame is received (SOF, data, and EOF), it needs 23 'sec to wake up the thread that monitors the receiver and parses the request frame. Immediately before dispatching the response frame, another 5 'sec of overhead is spent in firing up the transmitter. In between these two events, the RFID Guardian has 320.9 - (23 + 5) = 292.9 'sec to consult its ACL (and supporting data structures) and decide whether or not to block the RFID tag response.

How long this decision takes depends on how the RFID Guardian's ACL is organized. To find a coarse upper bound on the ACL length that can be handled by the Guardian prototype, we chose the slowest possible implementation for the ACL: an unsorted array of UIDs that can only be traversed sequentially to locate a specific UID. An RFID request addressed to the last item in the ACL was sent to the Guardian, forcing it to traverse the entire list. With 2600 entries, the Guardian was able to respond in time.

The Guardian prototype is equipped with a powerful XScale processor at high clock speed, 520 MHz. To find out if a Guardian with less processor power would still be feasible, we varied the clock speed of the XScale. The results are shown in Figure 11. The ACL length that the Guardian could still cope with decreases with clock speed, but much less than linearly. This is attributed to two causes: memory speed goes down more slowly and in coarser steps than CPU speed; and parts of the device processing are independent of CPU speed. At 208 MHz, the Guardian prototype can process ACLs of length 1800, even with this suboptimal ACL implementation.

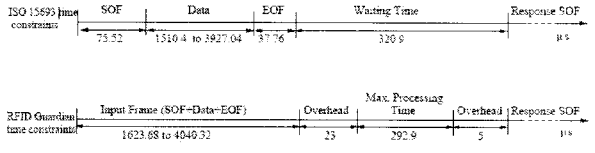

Figure 10: Timing constraints

Of course, with a hash table instead of a linear list, vast numbers of ACLs can be searched in the available 292.9 'sec. In short, ACL length is not likely to be a problem even on a very slow XScale.

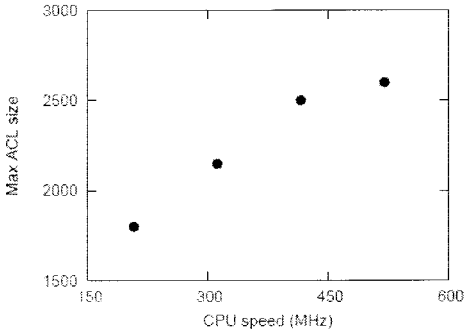

Figure 11: Maximum ACL size that can be processed at a given CPU speed

5.2 DoS Resistance

Now let us consider how attackers will try to defeat the RFID Guardian. They may use malicious readers or fake tags that try to confuse or lock up the RFID Guardian, so that the tags it protects can be read anyway. The primary defense against well-known exploits like buffer overruns must be very careful programming of the RFID Guardian software, which is helped by its limited code size.

Failing that, their next attack is likely to be a DoS (Denial-of-Service) attack to overload the RFID Guardian and prevent it from doing its job. Two RFID Guardian resources are obvious candidates for attack: its limited radio bandwidth and its limited memory. RFID communications always follow the master-slave pattern, where the tag (slave) must respond after a well-defined delay. Attacking during this delay is not feasible: it would immediately alert the RFID Guardian and it would confuse the tags as well. Attacking between reader commands does not constitute a DoS vulnerability of the communication channel: it would be the same as a regular reader action. The attacker could jam the channel, of course, but then he could not read out any tags, which is the presumed reason he wants to cripple the RFID Guardian.

The other potential vulnerability is the limited RFID Guardian flash memory. An attack on the flash memory may target any one of three data structures: the tag ownership list, the tag presence list, or the scan audit log. If an attacker with a battery-powered device simulated thousands of new tags in an attempt to fill up the ownership list or the current list, the RFID Guardian could warn the user about this abnormal activity.

Alternatively, the DoS attacker could try to fill up the audit logs. This does not cause a loss in protection of the owner's tags, but it certainly hampers the RFID Guardian's auditing capabilities. The maximum rate at

which requests can be launched is determined by the bandwidth of the radio channel and the minimum frame size, both of which are specified by the standard. The data rate is 26.48 kbps. The minimum frame is (SOF, 32 data bits, EOF) which takes 1.322 ms followed by a mandatory silence of 320.9 's, which works out to a maximum of 613 requests/sec.

An audit log entry contains the index of the tag being targeted, an index of the context, the command and a timestamp, which results in 2+2+1+4 = 9 bytes bytes. With 613 requests/sec, the attacker can fill up 5517 bytes of flash memory per second. The RFID Guardian prototype has 16MB of flash, of which 14MB is available for logging. Thus a maximum-speed attack would need 42 consecutive minutes of blasting away at full speed to fill the memory. Needless to say, the RFID Guardian should be sounding an alarm long before the memory begins to fill up, thus fulfilling its job of warning the user of an attack. Besides, flash memory is very cheap: another 16 MB might would add less than 2 dollars to the production cost.

To summarize, the RFID Guardian seems immune to the DoS attacks that we can identify, either because they would also disturb regular RFID interaction, or because the RFID Guardian has enough resources to defend itself long enough to alarm its owner after the threat has continued for some while.

6. Discussion

In contrast to the aforementioned Denial of Service attacks, there are a number of attacks that are successful against the RFID Guardian.

The RFID Guardian faces the 'hidden station' problem, which is a geometric problem that depends entirely upon radio ranges. However, we assume that an attacker wouldn't be able to maintain this for long, so we only deal with the "single reader" problem in this paper.

RFID readers could potentially trace the collision space, using collisions to resolve the IDs of RFID Guardian-protected protected RFID tags. We can improve this situation by adding some extra collisions, which will cause the algorithm to traverse a greater part of the ID space, making it look like more than one protected tag is present.

Another weakness of the RFID Guardian is its inability to jam reader queries. Selective RFID jamming only jams tag responses–not queries. However, queries can modify an RFID tag in unauthorized ways, like performing unauthorized data writes, or tag "killing". Other mechanisms can protect RFID tags from this, like temporary tag dectivation PETs (i.e. sleep/wake modes). However, this remains problematic for low cost RFID tags that might not support these other modes.

Finally, attackers can evade RFID Guardian protection by tracking people using tags with pseudonyms. If the RFID Guardian has the pseudonym list (or PRNG seed), it can correlate the IDs, remaining aware that it is dealing with only one tag. If the RFID Guardian doesn't have the list (or seed), it will think that it is dealing with multiple tags that are only observed once. The RFID Guardian also has trouble dealing with tags working with unknown standards/frequencies.

7 . RelatedWork

Given how great the threat of RFID technology is to privacy, it is not surprising that other researchers are also thinking about privacy defenders. Probably the closest work to ours is the RFID Enhancer Proxy[12], which shares some similarities with the RFID Guardian. The REP, too, is an active mobile device that performs RFID tag security managment, using a two-way communications channel between the REP and RFID Readers. However, the REP has some key differences from the RFID Guardian. The most important differences are as follows. First, the REP explicitly "acquires" and "releases" RFID tag activity, which the Guardian does not require. Second, the REP's two-way communications channel is "out-of-band," which requires extra infrastructure. Third, the "tag relabeling" mechanism requires

RFID tags to generate random numbers (or have a sleep mode), which many of them cannot do (or do not have). Fourth, the REP is purely theoretical; in contrast the RFID Guardian has been implemented and tested. RFID tag auditing (and cloning) are supported by several devices. FoeBuD's Data Privatizer[7] will detect RFID scans, find and read RFID tags, and copy data read to new tags. The Mark II ProxCard Cloner, by Jonathan Westhues[23] is a more general-purpose proximity-card cloner, that supports the emulation of several RFID frequencies and standards (the HW is elegant, but the SW is pending). Neither of these perform all the auditing, key management, access control, and authentication functions that the RFID Guardian does.

A less sophisticated approach to privacy protection is to block scans irrespectively of their originating reader. The Blocker Tag (Juels)[11] originated the concept of 'RFID blocking' as a form of off-tag access control. It is designed to abuse the tree-walk anticollision rotocol, and RFID readers are forced to traverse the entire id namespace when trying to locate RFID tags. This approach does not analyze incoming scans, look up

information in an access control list, and depending on what it finds, take action as the RFID Guardian does. Also, it has not been implemented. (A purely SW-based "soft" blocker tag has been implemented, but it expects RFID readers to self-regulate their behavior.)

An active device that can detect RFID scans is the M.I.T. RFID Field Probe[14]. It is a portable device, created by Rich Redemske at MIT Auto-ID Center, that integrates an RFID tag emulator and sensor probe. The HW consists of a semi-passive tag, a power level detector, and a helper battery. The RFID field probe gives audio and visual representations of the field signal strength and signal quality. However, its function is not to protect its owner's privacy, but as a tool to help vendors determine where on their pallets to attach the RFID tag to maximize signal strength for supply-chain management applications. Consequently, it does not have anything like our software, which is the heart of the RFID Guardian's privacy defense.

Several other RFID-based technologies support the concept of two-way RFID communications. Near Field Communications[3] is a peer-to-peer RFID-related communications technology. NFC devices can query RFID tags, and can also communicate with other NFC-enabled devices. However, NFC devices cannot talk with non- NFC enabled RFID readers and do not do privacy protection.

Finally, the RFID countermeasures described in Section 1.1 are all complimentary to the RFID Guardian, in the sense that the RFID Guardian could leverage them as part of its framework, for helping to provide personalized access control. However, none of them are discrete devices that protect privacy.

8. Conclusion

If we are ever immersed in a sea of RFID chips, the RFID Guardian may provide a life raft. This battery-powered device, which could easily be integrated into a cell phone or PDA, can monitor scans and tags in its vicinity, warning the owner of active and passive snooping. It can also do key management, handle access control, and authenticate nearby RFID readers automatically, taking its context and location into account, for example, acting differently at home and on the street. Furthermore, it can manage access to tags with sensitive content using Selective Jamming. No other device in existence or proposed has all of these capabilities. The RFID Guardian thus represents a major step that will allow people to recapture some of their privacy that RFID technology is threatening to take away.

However, what we have described here is only one step. We intend to further develop and improve the RFID Guardian by giving the prototype more capabilities. These capabilities include support for more frequencies and standards, improving the communication range, and simplifying the HW design. We also intend to further develop the security protocols that are needed for the authentication and key management facilities, thinking particularly about interaction requirements with the surrounding RFID infrastructure.

Table 1: RFID Tag Emulators for Security/Privacy

Tool Name	Tag emulation (SW)	Tag emulation (HW)	Scan auditing	Access control	Authentication	Implementation
NFC	√					√
Data Privatizer	√		√			√
Blocker Tag	√			√		√
Field Probe	√	√	√			√
ProxCard Cloner	√	√	√			√
RFID Enhancer Proxy	√		√	√	√	
RFID Guardian	√	√	√	√	√	√