Privacy For RFID Through Trusted Computing

David Molnar* Andrea Soppera† David Wagner‡

ABSTRACT

Radio Frequency Identification (RFID) technology raises significant privacy issues because it enables tracking of items and people possibly without their knowledge or consent. One of the biggest challenges for RFID technology is to provide privacy protection without raising tag production and management cost. We introduce a new architecture that uses trusted computing primitives to solve this problem. Our design splits the RFID reader into three software modules: a Reader Core with basic functionality, a Policy Engine that controls the use of RFID-derived data, and a Consumer Agent that performs privacy audits on the RFID reader and exports audit results to third party auditors. Readers use *remote attestation* to prove they are running a specific Reader Core, Policy Engine, and Consumer Agent. As a result, remote attestation allows concerned individuals to verify that RFID readers comply with privacy regulations, while also allowing the reader owner to verify that the reader has not been compromised.

Furthermore, industry standards bodies have suggested several mechanisms to protect privacy in which authorized readers use a shared secret to authenticate themselves to the tag. These standards have not fully addressed issues of key management. First, how is the shared secret securely provided to the legitimate reader? Second, how do we guarantee that the reader will comply with a specific privacy policy? We show how, with remote attestation, the key-issuing authority can demand such a proof before releasing shared secrets to the reader. We also show how *sealed storage* can protect secrets even if the reader is compromised. Finally, we sketch how our design could be implemented today using existing RFID reader hardware.

1. INTRODUCTION

Radio Frequency Identification (RFID) Tags consist of a small integrated circuit capable of transmitting an identifier to a reading device in response to a query. The widespread deployment of RFID technology poses significant privacy risks not present with prior technology. These privacy issues arise because RFID tags enable tracking of items and people, possibly without their knowledge or consent. Unnoticed by consumers, embedded tags could enable hidden surveillance.

We propose a new architecture for a *Trustworthy Reader* that makes RFID systems privacy friendly while retaining many of the legitimate benefits of current RFID technology. In our scheme, a RFID reader contains a Trusted Platform Module (TPM) chip. Our threat model is that an adversary may compromise the reader, but not the TPM. We believe this threat model is realistic because readers will be installed in many different places, some of them not physically secure. The TPM, in contrast, is a tamper-resistant hardware module with a narrow interface. In Section 3.3 we discuss existing and near-future RFID reader hardware that includes a TPM.

Using the TPM, we show how the Trusted Computing primitive of *remote attestation* enables any party to check that the reader will respect a particular privacy policy. We envision that privacy-friendly merchants will choose a privacy policy, post their policy, then use readers built according to our architecture. Privacy watchdogs and other interested parties can then perform random audits of readers to ensure that the reader matches the posted privacy policy. We also introduce a special piece of software, a *Consumer Agent* (CA) that enables automatic monitoring of privacy policies.

Furthermore, our architecture addresses an open problem common to most previous proposals involving RFID privacy, that of *key management*. Researchers and industrial standard bodies have recognized the privacy issue for some time and have suggested a set of solutions, such as the EPCglobal "kill" command, Infineon's proprietary mutual authentication extension to their ISO-18000-3 tags, or research on RFID pseudonyms [15, 2]. The common factor among all these approaches is the use of a secret key, shared between the reader and the tag and different for each tag. The reader uses this shared secret to prove that it is authorized to read the tag or to send other commands, such as a kill command.

The use of a shared secret raises questions about how key management should be performed. First, how do we provide the right password to the reader? In many applications, a

*dmolnar@eecs.berkeley.edu
†andrea.2.soppera@bt.com
‡daw@eecs.berkeley.edu

WPES'05 November 7, 2005, Alexandria, Virginia, USA.

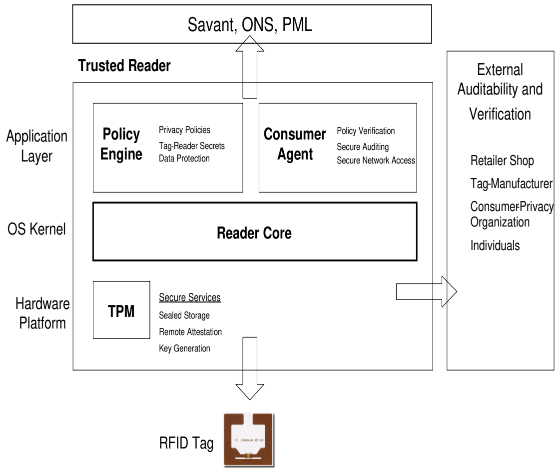

Figure 1: Block Diagram for a Trusted RFID Reader.

large number of tags will pass in front of a reader in a small amount of time, so any scheme must scale. Second, how do we determine which readers should be authorized? For instance, can we guarantee that the reader will comply with a specific privacy policy? From a consumer point of view, this would be very valuable. In a world of hidden tags, we would like to create a mechanism for accountability that enables us to verify the usage of specific privacy policies.

In practice, we expect it will be important to support offline readers. If we could be sure that every reader would have a low-latency high-reliability connection to some central key-issuing authority (a Trusted Center), then readers could obtain secrets from this center as they are needed. This does not work for offline readers, because readers have to contact the Trusted Center for every transaction. While there is precedent for such a thing in credit card transaction processing, we would like to do better. Therefore, we seek solutions that will work even when readers are mostly or wholly offline.

In the following section, we outline our design for a trusted RFID reader. We describe a way to split the reader software to allow for maximum flexibility in changing privacy policies on the fly, while also allowing for automatic auditing capability. We show examples of privacy policies that can be supported by our architecture. Finally, we show how Trusted Computing primitives can be used to remotely audit the privacy and security guarantees provided by our trusted RFID reader.

2. DESIGN

2.1 Splitting the Reader

In this section, we briefly outline the trusted reader architecture details (see Figure 1). We split the reader into three parts: a Reader Core, a Policy Engine, and a Consumer Agent.

Reader Core (RC) The Reader Core contains the operating system, radio interface, network capability, and other parts of the basic functionality of the RFID reader. The Reader Core must interface with the Trusted Platform Module to ensure that the TPM reflects the configuration of the machine at all times. A trusted root runs as part of the Reader Core and monitors everything that is launched on the reader. We assume that applications, such as the reader business logic, cannot modify the Reader Core and that the TPM is not compromised by the Reader Core. Recent work in this area includes the Enforcer system of Smith et al. [12] and the Trusted Computing Linux initiative of IBM [7]. We note that the business logic of the RFID reader is not part of the Reader Core; this logic can be updated at will, so long as the Reader Core can still guarantee the privacy properties we need.

Policy Engine (PE) The Policy Engine is a software module that enables the Trusted Reader to operate in a privacy-friendly manner. The Policy Engine is responsible for two main components. First, it has a machine-readable *policy file* that determines which tags an RFID reader is permitted to scan and the permitted uses of the resulting data. Second, the Policy Engine controls tag-reader secrets. For example, if a secret is needed to decrypt an RFID pseudonym or authenticate to an RFID tag, the Policy Engine will provide this secret to the Reader Core. We place the secrets in the Policy Engine to allow for the Policy Engine to be updated or migrated from reader to reader without needing to change the Reader Core.

Consumer Agent (CA) The Consumer Agent introduces a means for organizations and individuals to actively monitor and enforce privacy policies. This control is exerted by a Consumer Agent on a Trusted Reader that runs a trusted Reader Core and Policy Engine. The Consumer Agent interacts dynamically with the Reader Core and Policy Engine. In particular, it logs reading operations that have been performed or denied. Policy details and the reading log can then be reported to remote privacy auditors, such as consumer organizations, at regular intervals or on demand. Again, if the configuration of the system is compromised the Consumer Agent will report it, or even halt the operation of the reader. The owner of the RFID reader can also use remote attestation to verify that a particular Consumer Agent is running; this allows the set of Consumer Agents to be limited to those mutually acceptable to privacy auditors and the reader owner. We can even update the Consumer Agent on the fly as audit requirements change.

Our motivation for this split of the reader is to limit the amount of human audit required for each change in privacy policy. Because the trusted core is long-lived, we can afford to have human audit of the core, obtaining assurance that it will limit RFID secrets only to those uses specified in the policy file. In contrast, the machine readability of the policy file allows policies to be updated and readers retasked with new policies.

Our vision is that the Consumer Agent transforms privacy audits from an expensive, periodic process to a continuous, relatively cheap process. With the Consumer Agent, compliance monitoring is continuous and automated. While we expect that not all parts of a privacy audit can be mechanized in this way, we believe that many can be, including the policies we discuss in the next section. Therefore our architecture will lower the barrier to conducting a privacy audit, directly affecting an organization's cost for compliance. In particular, small organizations that do not have the infrastructure for a full privacy-audit may use a simple off-the-shelf Consumer Agent that implements one of a set of standard policies.

2.2 Possible Policies

We sketch some possible privacy policies that can be enforced by our architecture. This is not an exhaustive list; it merely illustrates the kind of policies that might be implemented and checked by a trusted RFID reader. Some of these policies might be embodied in off-the-shelf Consumer Agents for the use of small organizations.

- Do not retain any data on read tags for more than 5 minutes after the last known reading.
- If there is a "soft blocker" tag present, do not read any tags covered by the soft blocker [9].
- If the tag has a "privacy bit" set, do not retain any readings from that tag.
- Only retain data on tags in a certain set S (e.g. tags owned by the store, possibly specified by a common prefix or manufacturer ID).

  Do not retain any data on tags in a certain set S.
- Information about tags never leaves the reader in cleartext; such information is always encrypted with a specific public key before being exported.
- The history of tags is anonymized: no timestamp is associated with each tag reading and the log is shuffled before being exported.

In general, any policy that specifies reader behavior can, in principle be supported by our architecture. Again, the Consumer Agent can perform continuous auditing of this policy and transmit its audit results to authorized third party auditors at regular intervals. For a concrete example, consider a retail store. At point of sale, the retail store can rewrite the RFID tag identifier to set a "privacy bit," as suggested by Juels [8]. Then the privacy policy of compliant readers will prevent them from retaining readings from tags with the privacy bit set.

2.3 Limiting Secrets to Compliant Readers

We now show how trusted computing can limit secrets to RFID readers that will enforce a specific privacy policy. Our technical tool is *remote attestation,* a trusted computing primitive that enables a machine to prove to a remote party that it is running a particular configuration. In remote attestation, the Trusted Platform Module signs a list of hashes of software and configuration files currently run on the computer [5]. In our case, this is a hash of the Reader Core, a hash of the Policy Engine, a hash of the current policy file, and a hash of the Consumer Agent. This can be sent to the Trusted Center, which can then determine if the configuration is allowed to use a set of secrets or not.

2.4 Reader Compromise

We assume that the adversary may compromise a reader at any time. This is because the reader may be under the control of untrusted parties. For example, a shop owner has physical posession of the RFID reader, but customers may not want to trust the shop owner. Furthermore, as RFID takes off, there will be tens of thousands of RFID readers deployed in many different locations, some of which may not be physically secure. We do not want a single compromised reader to reveal all secrets or all privacy sensitive data.

Consequently, it is not enough to limit secrets to RFID readers that are running the correct Reader Core, Policy Engine, and Consumer Agent at time of attestation. We need some method for protecting these secrets after they have been provisioned to the reader. Here we use the trusted computing primitive of *sealed storage.* Sealed storage is storage encrypted with a key managed by the TPM; the TPM will decrypt this storage only for a certain configuration of the machine. In our case, this configuration is the Core, Engine, and Agent authorized to use these RFID secrets. Therefore, if the configuration changes later, such as through adversary compromise, the TPM will deny access to the key and hence access to the sealed storage.

3. PRACTICAL CONSIDERATIONS

3.1 Reader Core

The Reader Core is responsible for three main security tasks. First, it must interface with the Trusted Platform Module. Second, it must ensure that RFID secrets are used only as specified by the Policy Engine. Third, it must allow the Consumer Agent to carry out its auditing duties. We expect that these requirements will be verified by hand audit in the short term; automatically verifying them would be an interesting area for future research.

3.2 Policy Engine and Consumer Agent

Building a fully general Policy Engine and Consumer Agent for RFID secrets is likely to be difficult. First implementations, however, can fix a single application and "hard-code" a policy for that application. We expect that after the first few such applications, it will become more clear what features need to be supported by a policy language.

3.3 Building It Today

All of the elements of this architecture are easily achievable with existing products. In fact, these ideas could be implemented today, using only RFID readers and tags that are already shipping. The ThingMagic Mercury 4 uses the XScale2 processor family; the XScale2 includes a TPM chip. This reader runs Linux and has a software defined radio compatible with a wide range of tags. The trusted core could be built on top of a TPM-aware Linux kernel, such as Enforcer [12] or IBM's Trusted Computing Linux. We can then install, in sequence, the PE and CA software modules via the TPM endorsement key.

When a tag arrives in the proximity of the reader, a request to read secrets is sent to the PE module. This request is forwarded to the TPM, which verifies that the reader core is still valid. The TPM then sends back this data to the trusted root and the PE is executed. A similar process is performed for the CA module; the only difference is that a log is created for each reading operation and the information can be distributed to remote third parties. As noted, an initial implementation could hard-code the desired privacy policy. While significant engineering challenges remain before this can become a shipping product, there are no technology barriers: the main pieces are all available today.

4. RELATED WORK

Karlof et al. on used a TCG-compliant TPM to protect privacy secrets in the context of reputation logs maintained by a software agents network [11]. Privacy and civil liberties applications of Trusted Computing are also pointed out in the context of the Terra secure virtual machine monitor by Garfinkel et al [4]. The notion of a Consumer Agent is similar to the external auditing "observer" introduced by Chaum in the context of digital cash [1]. Our notion of attesting to a configuration file together with an application is also seen in the work of Sailer et al. [18], but while they focus on a general-purpose system, our focus is on special-purpose RFID readers. Nakamura et al. have begun an investigation of trusted computing applied to the RFID setting and have a prototype implementation using the IBM Trusted Platform on Demand architecture [14, 13]. IBM's Arcom division has a demonstration of an XScale2 board with an RFID application, although details are not available [17]. Karjoth et al. show how privacy intentions can be formalized in machine-parsed privacy policies; this is the kind of work that would be needed for our Policy Engine [10]. Floerkemier et al. also discuss adapting the Fair Information Principles to RFID readers with a machine-readable policy and show how auditing can be achieved with a special "watchdog tag" that looks for noncompliant readers [3]. Smith et al.'s Enforcer system [12] and IBM's Trusted Computing Linux initiative [7] have shown how to use a TPM to prove properties about the software running on a TPM-enabled platform. Finally, we note that Intel's XScale2 processors contain a TPM, and Intel has announced that their future desktop motherboards will contain a TPM chip as well [6, 16]. While there is a great deal of work on Trusted Computing, and some of it concerns RFID readers, our design provides for isolation between business logic, flexible update of audit logic, and automatic checking of privacy policies, features which do not appear to be in previous work.

5. CONCLUSIONS

We have shown that Trusted Computing can aid privacy in the RFID setting by allowing anyone to verify that readers handle RFID-derived data according to a specific privacy policy. Furthermore, provisioning secrets to a large network of RFID readers raises significant practical issues in protecting these secrets. Our design shows that Trusted Computing primitives are effective at addressing these issues. Because an RFID reader is typically an embedded device, we can use a much smaller operating system than is possible in the general computing case. While implementing a general-purpose policy language is likely to be difficult, application-specific first steps can still make use of these primitives for early deployments. Therefore, trusted computing primitives should be seriously considered for the design of any large-scale RFID deployment.

6. ACKNOWLEDGEMENTS

We thank Guenter Karjoth for helpful discussions on mechanizing privacy policies and for pointing us to the work of Nakamura et al. We thank Megumi Nakamura for providing us with an overview of IBM work on RFID readers and trusted computing. We also thank Trevor Burbridge and Jeff Farr from BT for helpful comments. This work was supported by NSF grants CCR-0093337 and by a generous donation from BT. David Molnar is supported by an NSF Graduate Research Fellowship.

7. REFERENCES

[1] D. Chaum and T. Pedersen. Wallets databases with observers. In *CRYPTO 1992*, 1992.

[2] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In *CHES*, 2004.

[3] Christian Floerkemeier, Roland Schneider, and Marc Langheinrich. Scanning with a purpose – supporting the fair information principles in RFID protocols. In Hitomi Murakami, Hideyuki Nakashima, Hideyuki Tokuda, and Michiaki Yasumura, editors, *Ubiquitious Computing Systems. Revised Selected Papers from the 2nd International Symposium on Ubiquitous Computing Systems (UCS 2004), November 8-9, 2004, Tokyo, Japan*, volume 3598 of *Lecture Notes in Computer Science*, Berlin, Germany, June 2005. Springer-Verlag.

[4] T. Garfinkel, M. Rosenblum, and D. Boneh. Flexible OS support and applications for trusted computing. In *HotOS-IX*, 2003.

[5] Trusted Computing Group. Trusted computing platform module specification v1.1, 2005.

[6] Ed Hardy. Intel unveils next-generation XScale processors, 2004. `http://www.brighthand.com/article/Intel_Unveils_PXA270_XScale_Processor%s`.

[7] IBM. IBM Trusted Linux, 2005. `http://www.research.ibm.com/gsal/tcpa/`.

[8] Ari Juels. A bit of privacy, 2005. `http://www.rfidjournal.com/article/articleview/1536/1/133/`.

[9] Ari Juels and J. Brainard. Soft blocking: Flexible blocker tags on the cheap. In *WPES 2004*, 2004.

[10] G. Karjoth, M. Schunter, and E. Van Herreweghen. Translating privacy policies into privacy promises – how to promise what you can keep. In *Workshop on Policies for Distributed Systems and Networks*, 2003.

[11] C. Karlof, Y. Li, and E. Ong. Using trustworthy computing to enhance privacy, 2002. `http://www.cs.berkeley.edu/~daw/teaching/cs261-f02/reports/karlof.ps`.

[12] J. Marchesini, S.W. Smith, O. Wild, J. Stabiner, and A. Barsamian. Open-source applications of TCPA hardware. In *Applied Computer Security Applications Conference*, 2004.

[13] H. Maruyama, F. Seliger, N. Nagaratnam, T. Ebringer, S. Munetoh, S. Yoshihama, and T. Nakamura. Trusted platform on demand, 2004. IBM Research Report RT0564.

[14] M. Nakamura, T. Mishina, and S. Munetoh. Integrity validation infrastructure for RFID edge controllers. In *SCIS2005*, 2005. In Japanese.

[15] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to a privacy friendly tag. In *RFID Privacy Workshop, MIT*, 2003.

[16] Desktop Pipeline. Intel introduces new business, home platforms, 2005. May 26, 2005. `http://www.desktoppipeline.com/163701495`.

[17] Security ProNews. Embedded systems designers to see trusted computing components in action, 2004. `http://securitypronews.com/articles/security/spn-23-20040908EmbeddedSys%temsDesignerstoSeeTrustedComputingComponentsinAction.html`.

[18] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Usenix Security*, 2004.