Robust, Anonymous RFID Authentication with Constant Key-Lookup

Mike Burmester*
Florida State University
Dept. of Computer Science
Tallahassee, FL 32306
United States of America
burmester@cs.fsu.edu

Breno de Medeiros
Florida State University
Dept. of Computer Science
Tallahassee, FL 32306
United States of America
breno@cs.fsu.edu

Rossana Motta
Florida State University
Dept. of Computer Science
Tallahassee, FL 32306
United States of America
motta@cs.fsu.edu

ABSTRACT

A considerable number of anonymous RFID authentication schemes have been proposed. However, current proposals either do not provide robust security guarantees, or suffer from scalability issues when the number of tags issued by the system is very large. In this paper, we focus on approaches that reconcile these important requirements. In particular, we seek to reduce the complexity of identifying tags by the back-end server in anonymous RFID authentication protocols—what we term the *key-lookup* problem.

We propose a compiler that transforms a generic RFID authentication protocol (supporting anonymity) into one that achieves the same guarantees with constant key-lookup cost even when the number of tags is very large. This approach uses a lightweight one-way trapdoor function and produces protocols that are suitable for deployment into current tag architectures. We then explore the issue of minimal assumptions required, and show that one-way trapdoor functions are necessary to achieve highly scalable, robustly secure solutions. We then relax the requirement of unlinkable anonymity, and consider scalable solutions that are provably secure and for which the loss of privacy is minimal.

Categories and Subject Descriptors

K.6 [**Security & Protection**]: Authentication; K.6 [**Miscellaneous**]: Security; D.2 [**Software/Program verification**]: Formal methods; Reliability; Validation; D.4 [**Security and Protection**]: Authentication; Cryptographic controls; Information flow controls; C.3 [**Special-purpose and Application-based Systems**]: Smartcards; C.4 [**Performance of Systems**]: Reliability, availability, and serviceability.

General Terms

Algorithms, Design, Reliability, Security, Theory.

*This material is based on work supported by the DoD IASP program under grant number 022317.

ASIACCS '08, March 18-20, Tokyo, Japan

Keywords

RFID, privacy, availability, scalability, provably secure protocols, unlinkability.

1. INTRODUCTION

Radio-Frequency Identification (RFID) enables objects to be identified by radio waves, without physical contact and without need for line-of-sight alignment. The flexibility of this technology holds great promise for novel applications, and increasingly RFID tags are being deployed in situations where their proper operation must be assured to a medium or high level of confidence. Well-known examples are the use of RFIDs to harden identification documents against forgery, or to provide access control to physical resources and/or secure locations. In addition to the authenticiy and integrity requirements, it is often desirable, and sometimes required, that RFIDs provide anonymized identification services, to preserve the privacy of the persons that carry them.

A considerable body of research has been developed to provide solutions to the anonymous authentication problem in RFID [1, 3, 14, 18, 22, 23]. However, currently available solutions either do not provide robust security guarantees, or suffer from scalability issues when the number of tags issued by the system is very large. The principal reason leading to this conflict between requirements is the small circuit footprint available on RFID tags, which has so far limited implementation of cryptography in tags to symmetric-key algorithms. In the anonymous setting, symmetric-key approaches introduces the difficulty that the server must first decide which tag (and corresponding key) should be used to validate a tag's authentication transcript. This difficulty is worsened if the system has a large number of tags, creating vulnerabilities to denial-of-service attacks, and potentially raising threats to privacy through timing attacks.

In this paper, we focus on the worst-case complexity (time and computation) of identifying tags, by searching for matches in the symmetric-key database of the back-end server. More specifically, we consider the ratio between the costs of (1) authenticating the response of a tag against a single tag identity, and (2) authenticating the response of a tag in an anonymous interaction (when the identity of the tag is not known a priori). This ratio we call the *key-lookup cost*. In the worst case, for anonymous RFID authentication, the key-lookup cost is linear in the number of tags (the server has to exhaust the symmetric-key database to find a match). Molnar–Soppera–Wagner [18] presented an anonymous RFID protocol that achieves logarithmic key-lookup,

by using a binary tree of symmetric-keys (the tree of secrets), and assigning to each tag the keys of a root-to-leaf path: the response of a tag is then linked to this path, and this link is used to identify the tag (only $2 \log T$ checks are needed, where T is the number of tags). Burmester–van Le–de Medeiros [3] use a different approach, in which the key-lookup is constant for tags that have not been previously interrogated by rogue readers (invoked by the adversary), but otherwise it is linear.

Organization of this paper. After this introduction we discuss, in Section 2, the conflict of privacy and availability and the impact of privacy on the scalability of key-lookup. We then describe in Section 3 a compiler that transforms a generic RFID authentication protocol satisfying privacy requirements into one that achieves strong security with constant key-lookup when the number of tags is large. This approach uses a lightweight one-way trapdoor function—described recently by Shamir [21], and produces protocols that are suitable for deployment into current tag architectures. In Section 4 we show that one-way trapdoor functions are necessary for strongly privacy-preserving RFID authentication that supports constant-cost key-lookup, even for large numbers of tags. Finally, in Section 5 we relax the requirement for unlinkable anonymity and consider provably secure solutions for which the loss of privacy is minimal.

Our main contributions

- A compiler that transforms any RFID authentication protocol of a certain form into one that achieves scalability for the back-end server, providing for constant key-lookup cost (Section 3.2).
  - This improves on the worst-case cost achieved by the most efficient key-lookup scheme to date—$O(\log n)$ by [18].

  The compiler produces new protocols from existing ones, in such a way as not to weaken any privacy and authenticity guarantees enjoyed by the compiled ones.
  - In particular, it can be used to construct schemes with constant key-lookup cost that achieve higher security than the scheme in [18], for instance by not requiring the use of keys that are shared by many tags.
- A lightweight implementation of the compiler based on the Shamir adaptation of Rabin's one-way function (Section 3.3).
- A security proof for the compiler (Section 3.5).
- A proof that any RFID authentication protocol that is strongly privacy-preserving with constant key-lookup must also employ public-key obfuscation, when the number of tags is large (Section 4).
- A more efficient, alternative approach for provably secure RFID authentication that supports constant key-lookup, under a minimal relaxation of privacy that allows for limited linkability (Section 5).

2. PRIVACY VS AVAILABILITY

Support of privacy in RFID tends to conflict with fundamental requirements, such as availability, i.e., the ability of the system to function correctly and continuously, through its projected lifetime. Attacks against availability work by forcing components of the RFID system into temporary or permanent states from where they are no longer capable of fulfilling their proper roles.

There are several attack strategies against RFID systems that target availability. For instance, jamming attacks seek to overwhelm the communication medium with noise; such attacks can be detected and mitigated by mechanisms at the physical layer [22]. In this paper, we focus instead on mechanisms that support availability at the protocol level (RFID application layer).

Storms in wireless systems are caused when the number of transmissions exceeds the capacity of the system to process them, thus restricting availability. Typically storms are linked to design and protocol failures, and are not caused directly by the adversary. For example, network flooding[1] in a wireless network may cause a storm, if the local node density is high. The same applies for RFID systems: RFID readers may not be able to process all tag responses, when the number of tags is large. This may result in some tags not being authenticated.

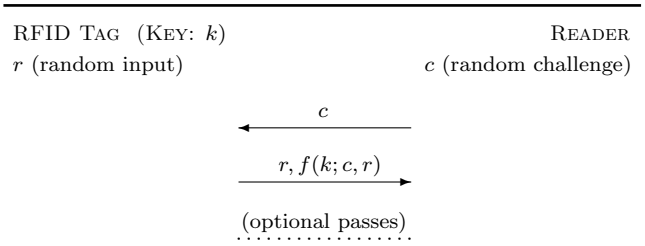
The adversary may exploit the lookup complexity by having a rogue tag make faulty responses on behalf of several "virtual" tags. It may not be easy to detect such attacks, because they cannot be distinguished from non-adversarial faulty responses. If the back-end server spends more resources trying to disambiguate fake responses than the adversary spends on generating them, then the RFID system is inherently flawed. Note that the cost of triggering such an RFID storm is restricted to the cost of selecting EPC channels [10], one for each response, since the faulty response can be generated by simply updating a counter (or some similar mechanism). Furthermore the rogue tag is not subject to the usual tag constraints (e.g., it can have its own power supply). It is therefore important to design RFID systems with scalable key-lookup, as systems with large number of tags may require a constant cost of key-lookup to achieve resilience against attacks that target availability (denial-of-service attacks).

At the protocol level, one must also deal with a class of attacks against availability, termed *disabling attacks*, that target state synchronization requirements. More precisely, strong authentication (in a symmetric-key setting) requires that RFID tags and the back-end server share secrets (e.g., keys and other state information). In the case of privacy-preserving protocols, mutable information (e.g., a changing pseudonym) must be used by the back-end server to recognize the tag in the absence of fixed identification values. These represent shared dynamic states that must be maintained in synchrony by tags and the back-end server. Disabling attacks seek to break this synchronicity.

Among RFID protocols that provide strong privacy guarantees, some have limited ability to tolerate attacks against availability. For instance, in the Ohkubo–Suzuki–Kinoshita protocol [19], the attacker may use invalid timestamps to disable tags temporarily or permanently. Other solutions

[1] In network flooding all neighbors of a node will re-broadcast a newly received message.

Figure 1: A generic challenge-response RFID protocol.



use hierarchic key structures to speed up lookup time, but are consequently more vulnerable to key-exposure threats as the higher-level keys are shared among many tags. For instance, if the keys of a tag in the Molnar–Soppera–Wagner protocol [18] are compromised, then its sibling tag can be identified (from the key of their parent), and more generally, cousin tags at distance t can be identified as a group (from the key of their t-ancestor).

Yet other protocols require linear search among all issued keys to authenticate a response, an approach that is infeasible for large numbers of tags. An improvement over always requiring an exhaustive search is to employ an optimistic approach [3, 24]. In this case, the server is normally able to recognize the tag in constant time based on a pseudonym value, but when this value becomes de-synchronized, the server can recover the tag identity through linear search among the valid (issued) keys. All protocols that (in some circumstances) require a linear search suffer from scalability issues as the number of tags in the system increases.

In what follows, we describe how to systematically modify RFID protocols to achieve constant-time effort to authenticate a tag, resolving scalability issues and reconciling privacy and availability requirements of RFID applications.

3. A SCALABILITY-PROVIDING PROTOCOL COMPILER

In this section, we provide a high-level description of our approach, a compiler that can transform many challenge-response RFID protocols to achieve scalability, supporting constant cost for RFID key-lookup.

3.1 A Generic Challenge-Response RFID Protocol

Fig. 1 illustrates a typical challenge-response RFID authentication protocol. In the first pass, the reader produces a challenge c that could include a timestamp, a random nonce, or other information as specified by the protocol. In the second pass, the tag evaluates and broadcasts the result of computing a function $f(k; \cdot, \cdot)$ on the challenge and (possibly) on additional input r generated by the tag. The value r could embed a tag nonce and either an identifier (if privacy is not a concern), or a (mutable) pseudonym to facilitate tag recognition without leaking its identity. The dotted line in Fig. 1 indicates optional passes for added functionality, such as mutual authentication, key-update for forward-security, etc.

The security and efficiency provided by such generic protocols are highly related to the choice of the function f, which is keyed with a symmetric-key k unique to the tag and known to the back-end server (not depicted above). Even restricting our attention only to protocols that provide privacy, there are many possibilities for implementation of the above protocol, providing different security guarantees. For instance, if unlinkable privacy is desired, the outputs of the function f must be indistinguishable from pseudo-random. Protocols may further differ on the method for pseudonym update, and may provide for additional features—for instance, protocols may only support authentication when the reader has on-line access to the back-end server, or conversely, may be suitable for reader-server batch communication.[2]

In the following, we assume that we deal with a generic RFID protocol as in Fig. 1 that suffers from lack of scalability of key-lookup. We also assume that the protocol must achieve privacy, since otherwise scalability may be easily achieved by having the tag reveal its identifier to the server, thus allowing for immediate key-lookup.

3.2 The Compiler

Let $h(\cdot)$ stand for a trapdoor one-way function, where the trapdoor t is known only to the back-end server (the Verifier). We fix the protocol in Fig. 1 to achieve key-lookup scalability by changing the form of the tag's response to

$$r, h(id, r'), f(k; c, r \oplus r'), \tag{1}$$

where f is as in Fig. 1, id is a tag identifier, r' is a random nonce, selected from $R = \{0,1\}^n$ uniformly at random, and "$\oplus$" denotes bitwise xor. Here n is a security parameter. When receiving the above response (relayed by the Reader), the back-end server (the Verifier) can use its knowledge of the trapdoor to recover id. It may then apply the steps of the (non-scalable) version of the protocol.

In order for the compiler to preserve the security properties of the original protocol, the function $h(\cdot, \cdot)$ must satisfy some properties.

1. The probability ensembles

   $$\{h(id_1, r')\}_{r' \in \{0,1\}^n} \text{ and } \{h(id_2, r')\}_{r' \in \{0,1\}^n}$$

   must be computationally indistinguishable, for any pair of identities id_1, id_2. (To preserve unlinkable anonymity.)

2. The function $h(id, \cdot)$ is one-way (i.e., in the variable r'), for any choice of id. (To preserve transcript integrity.)

The need for condition (1) should be obvious, as otherwise it allows for distinguishing between two identities, breaking unlikable anonymity of the underlying protocol. Note that (1) does NOT imply (2), because the identities are not known to the adversary. For instance, (1) is satisfied by the function $g(id, r') = (r', \eta(id; r'))$, where η is a pseudo-random function (PRF) with key id. Now, for the necessity of condition (2), consider the following attack when (2) does not hold: The adversary inverts $h(id, r')$ to obtain the value r' in each of two sets of responses from (potentially distinct) tags: $V_1 : r_1$, $h(id_1, r'_1)$, $f(k_1; c_1, r_1 \oplus r'_1)$ and $V_2 : r_2$, $h(id_2, r'_2)$, $f(k_2; c_2, r_2 \oplus r'_2)$, constructing the response $V : r_3$,

[2] In this case, the tag validation may be delayed until the next batch interaction (e.g., [23]), or may be immediate with (limited) delegation by the back-end server to the reader [18].

$h(id_1, r'_1)$, $f(k_2; c_2, r_2 \oplus r'_2)$, with $r_3 = r'_1 \oplus r_2 \oplus r'_2$. Then the adversary substitutes V_2 by V: V will be accepted by the back-end server if $id_1 = id_2$, but only with negligible probability if $id_1 \neq id_2$. This violates the anonymity of the protocol, endowing the adversary with an effective mechanism to link tag transcripts (through active attacks).

Under the assumptions (1) and (2) above, we can show that the compiled protocol provides the same security guarantees as achieved by the initial protocol while guaranteeing constant key-lookup cost. In the following we shall first describe a lightweight implementation of a one-way trapdoor function (Section 3.3), and then demonstrate the explicit efficiencies (Section 3.4) and security guarantees (Section 3.5) achieved by the compiler when applied to a family of protocols [3, 24] that is secure under a very robust security model (universal composability).

3.3 A Lightweight One-way Trapdoor Function

The greatest challenge in making the above compiler practical is to design very efficient one-way trapdoor functions with the required properties. First, we point out that since the RFID tags never need to perform operations using the trapdoor—from the perspective of a tag h is simply a one-way function, this asymmetry can be exploited to obtain more efficient schemes. There are several alternatives that could be used to implement the function efficiently. The most interesting approach uses a recent construction SQUASH (from SQUaring haSH), proposed by Shamir in [21], that shows how to implement modular squaring (where the modulus N is reasonably large, say 1024 bits) while using just a few hundred gate-equivalents (GEs) for computation and another several hundred GEs for read-only storage. Because only arithmetical operations are used, this approach can be implemented very efficiently, while from a security point of view it is as hard as integer factorization [21].

3.4 Applications and Implementations

The compiler can be applied to practically any anonymous RFID protocol to establish constant key-lookup. In particular, to the lightweight RFID protocols O-TRAP and O-FRAP presented in [3, 24], for strong UC security with constant key-lookup. We next discuss efficiency aspects of implementing the protocols that result from application of the compiler to the O-TRAP scheme.

The authenticator f. In the O-TRAP/O-FRAP family of protocols, f is realized by a PRF, which in practice can be implemented using a variety of well-known constructions. Efficiency vs. security trade-offs in this architecture are easily achieved, as key-size and pseudo-randomness (estimated as the logarithm of the PRF cycle) can be chosen to the granularity of individual bits. Here we discuss two implementation strategies based on different PRF instantiations.

Using a well-known technique by Goldreich et. al. [13], it is possible to build a PRF that makes a call to a pseudo-random number generator (PRNG) per bit of input processed. In turn, a very efficient PRNG implementation can be achieved using linear feedback shift registers, such as the self-shrinking generator [7]. This results in a small number of bit operations per input and output bits. The entire footprint of this implementation has been estimated to require only 1435 logic gates (within 517 clock cycles and 64B memory), achieving 128-bit security [16].

Block ciphers can similarly be used to implement PRFs through a number of standard constructions—their concrete security was analyzed in [2]. Recently, highly optimized implementations of the Advanced Encryption Standard (AES) block cipher [8] have been achieved, and these are suitable for RFID architectures [12]. An RFID architecture using this implementation was proposed in [11], with footprint equal to 3400 GEs and mean current consumption equal to 8μA, assuming a clock rate of 100kHz, and within 1032 clock cycles.

The obfuscator h. The Rabin cryptosystem [20] is a public-key encryption scheme that uses modular squaring with composite modulus $N = pq$, p, q primes, to encrypt data. The public key is N and the private key is the factorization of N. In particular, if x is the plaintext then the ciphertext is $y = x^2 \bmod N$. To decrypt y the factors of N are used: there are four quadratic residues of y, one of which is x. In Shamir's adaptation, modular squaring is replaced with integer squaring: $h(x) = x^2 + rN$, where r a random number slightly larger than N [21]. It is not difficult to show that inverting h is as hard as factoring composite numbers [21].

Cost of obfuscating. Let $x = id||r'$. We need to compute $h(x) = x^2 + kN$, for a 1024-bit wide N. The square x^2 and the product kN, are computed separately on-the-fly, using a 128 bit register, and then combined (with carries buffered for the next iteration). To evaluate individual bits of x^2 and kN, we convolute x with itself, and k with N, using the 128 bit register, and invoke a PRNG to generate the bits of x and k on-the-fly. 8 invocations will be needed. The cost of implementing h is then: (i) 512 NOT gates for read-only storage of the 1024-bit modulus N and, (ii) a PRNG and buffers for the computations. Since f and h are computed sequentially, we can re-use the PRNG and buffers of f. So the circuit complexity of h is < 1000 GEs.

Total cost of scheme. In total, assuming a PRG-based implementation of f, the cost is $\simeq 1435 + 1000 = 2435$ GEs.

3.5 The Security Proof for the Compiler

In this section, we prove the result for the case where the generic challenge-response RFID protocol in Fig. 1 realizes one-way authentication with strong (unlinkable) privacy in the UC framework. Our goal is to show that the compiled protocol realizes the same security level, while providing constant key-lookup cost.

We now present the essential ideas to construct a security proof in the UC framework. We are informal in dealing with some aspects of the UC security formalization, e.g., omitting references to *session identifiers*. Our interest is to convey the general ideas behind the proof, leaving rigorous analysis to the full version of the paper, where we shall also consider cases when the original protocol achieves only weaker guarantees, or only supports security in frameworks weaker than the UC model.

The UC framework defines the security of a protocol π in terms of the interactive indistinguishability between real- and ideal-world simulations of executions of π [4, 5, 6]. In the real-world executions, honest parties are represented by probabilistic polynomial-time Turing (PPT) machines that

execute the protocol as specified, and adversarial parties, also represented by PPTs, that can deviate from the protocol in an arbitrary fashion. The adversarial parties are controlled by a PPT adversary that has full knowledge of their state, controls the communication channels of all parties (honest and adversarial), and interacts with the environment in an arbitrary way, and in particular eavesdrops on communications. The ideal-world executions are controlled by an ideal functionality $\mathcal{F}$, a trusted party that guarantees the correct execution of the protocol—in particular, $\mathcal{F}$ emulates all honest parties. The ideal-world adversary is controlled by $\mathcal{F}$ to reproduce as faithfully as possible the behavior of the real adversary. We say that π realizes $\mathcal{F}$ in the UC framework, if no PPT environment $\mathcal{Z}$ can distinguish (with better than negligible probability) real- from ideal-world protocol runs.

In our case π is the compiled protocol, whereas the original protocol $\hat{\pi}$ is of the form in Fig. 1, and achieves UC anonymous authentication, as defined for instance, in [24]. In particular, $\mathcal{F}$ takes the following actions in the idealized protocol executions:

- Generates the challenges for honest READERs.
- Receives challenges on behalf of honest TAGs, or from the adversary.
- Generates responses on behalf of honest TAGs.
- Decides which TAG responses are authentic on behalf of the Verifier.

Note that because we are assuming that the generic RFID protocol is UC secure, the function $f(\cdot;\cdot,\cdot)$ is pseudo-random, while $h(\cdot,\cdot)$ satisfies the conditions (1) and (2) in Section 3. To prove that π is secure we show that each behavior securely provided by $\mathcal{F}$ can also be achieved in the real-world through π. That is, we simulate the operation of π with access to $\mathcal{F}$ by the real-world operation of the protocol that does not rely on $\mathcal{F}$. We summarize the key features of π, that represent the real-world protocol runs:

- The challenges of the Verifier are received through a READER.
- The responses of the TAGs are mediated by a READER, that may be adversarial.
- The adversary controls the adversarial READERs and may, modify or interrupt any channels at will—but cannot tamper with the contents of the channels connecting honest READERs to the Verifier.

The main difference between the real- and ideal-world is that the values produced by $\mathcal{F}$ are generated as truly random, as opposed to pseudo-random. More precisely, at the beginning of the simulation, $\mathcal{F}$ choses a special identity id^* among the set of all possible identities. Later, whenever π produces a response on behalf of the tag with identity id, and on input the challenge c, the functionality $\mathcal{F}$ generates a random value r, and checks if it has an entry $(\mathsf{function\_value}; id, c, (r, F))$ in its database, for any F in the output space of f. If so, $\mathcal{F}$ sets $\rho \leftarrow (r, F)$. If not, it selects a new value F at random (in the output space of f), and sets $\rho \leftarrow (r, F)$, entering $(function\_value; id, c, \rho)$ into its database. It then selects H according to the probabilistic ensemble $\{h(id^*;\cdot)\}$, and enters the record $(\mathsf{identity}, id, r, H)$ in its database. Finally, it returns the values ρ, H as the tag's response in the ideal-world.

A value (ρ, H) is authentic (against challenge c) if there is a record of it in the database; more precisely, if there are entries $(\mathsf{identity}, id'', r'', H'')$ and $(\mathsf{function\_value}; id', c', (r', F'))$, with $id'' = id'$, $r'' = r'$, $H = H''$, $c = c'$, and $\rho = (r', F')$. Observe that this specification of $\mathcal{F}$ makes several security guarantees obvious: unforgeability, freedom from replays and substitutions, privacy (unlinkable anonymity), etc. Indeed, violating such properties requires the adversary to guess randomly produced values.

In order to show that ideal world protocol executions are indistinguishable from real ones, we proceed in stages. First, we consider a real-world simulation of protocol π^*, which is distinguished from protocol π only in that tags send values $h(id^*, r)$ in the last flow, for a fixed tag identity id^*, regardless of the actual identity of the user. The server uses the trapdoor only to recover the value r, discarding the tag identity token id^*, and tries to validate the response against each of the issued (and valid) tag keys.

Real-world executions of protocol π^* are not distinguishable from protocol π, under the assumption that the server implementing protocol π^* has enough processing power to exhaust its database at roughly the same throughput that the server implementing π can invert the trapdoor function $h(\cdot,\cdot)$ (using the trapdoor information), so that no delays are noticeable. This is so because of our assumption on the indistinguishability of distributions of outputs of the function h on different identities.

Note that, while in π the value $h(id, r)$ "binds" the identity id to the argument of the function $f()$, the loss of this binding does not noticeably affect the security of π^*. The reason is that any attack to π^* that changes the inputs to $f()$ and allows for the creation of a novel, valid transcript would also represent an attack on the original (uncompiled) protocol $\hat{\pi}$, violating the assumption that $\hat{\pi}$ is UC-secure.

Now that we have shown the indistinguishability between real-world executions of π and π^*, we can invoke the composition theorem of the UC framework to reduce the security of π^* to that of the original (uncompiled) protocol $\hat{\pi}$. Indeed, consider the following simulation of π^* by $\hat{\pi}$: The simulator $\mathcal{S}$ chooses a trapdoor function $h()$—e.g., generates a new one. (For the purposes of this simulation, it is not necessary that $\mathcal{S}$ learn the value of the trapdoor, since it does not need to invert it.) $\mathcal{S}$ impersonates the server interacting with π^*, simply relaying the messages of the trusted server that freely interacts with other parties in a simulation of $\hat{\pi}$. However, when in the simulation of $\hat{\pi}$ some tag produces an authentication response $(w, z) = (r, f(k; c, r))$, then S chooses a random value $\tilde{r}$ and computes $y = h(id^*, \tilde{w})$, $x = \tilde{r} \oplus w$, and outputs (y, x, z) on behalf of some other honest tag in the simulation of π^*. It is now clear that an adversary that succeeds in violating the security of π^* also produces a strategy (through the simulation) to defeat the security of $\hat{\pi}$, contradicting the assumed security of the later.

It follows that when the challenge-response protocol UC-realizes one-way authentication with strong privacy, then the compiled protocol will maintain this security level. We get scalable key-lookup because the server can efficiently invert the function h. □

4. STRONG PRIVACY WITH CONSTANT KEY-LOOKUP IMPLIES PUBLIC-KEY OBFUSCATION

We show that when the number of tags T is large, anonymous, unlinkable RFID authentication with constant key-lookup implies public-key obfuscation. To achieve this, we clarify in greater detail what we mean by constant cost of key-lookup.

The security parameter n serves as a natural constraint on algorithmic efficiency. More specifically, a feasible algorithm is characterized by having its cost factor dominated by some polynomial $p(n)$. Let $DB(n)$ be the number of tags in the server database, in some instance of the RFID scheme, with security parameter n. We require that the algorithm that lists all entries in $DB(n)$ be feasible, since the database must be constructible—hence, the size of $DB(n)$ is bound above by some polynomial $p_{DB}(n)$. Note that the cost to the (honest) server to invalidate an answer from the adversary, by exhaustion in the database, is $O(DB(n) \times val(n))$, where $val(n)$ is the cost to authenticate an honest tag, if the server knows its identity in advance. Therefore, if the strategy (in the worst-case) is to use exhaustive search for the key, then the key-lookup cost $lookup(n)$ is $O(DB(n))$ and it is only constant when $DB(n)$ is constant. By contrast, our definition of scalable key-lookup requires that the lookup cost $lookup(n)$ is constant whenever the size of $DB(n)$ is bounded by a polynomial in n.

To simplify our argument, we make the following assumptions on the obfuscator h and the authenticator f:

1. $h(id, r')$, $r' \in_R \{0,1\}^a$, is as in Section 3.2,

2. $f(k; c, r \oplus r')$, $r \in_R \{0,1\}^b$, is pseudo-random,

with a, b linear in n. We require the number of tags T to be an increasing function of n, otherwise the key-lookup cost would be constant simply be exhausting all the keys.

The proof is straightforward. Suppose that a tag's response $r, h(k, r'), f(k; c, r \oplus r')$, to the server's challenge c identifies the tag to the server with overwhelming probability, say $1 - \varepsilon$, ε negligible (in n). Then it is easy to see that the obfuscator h will identify the tag to the server with non-negligible probability. Indeed, the contribution of the authenticator f to the identification of the tag in constant key-lookup time is asymptotically smaller than 1 by a non-negligible amount. More specifically, if the server can only check the authenticator of a constant number ℓ of tags for a possible match, then it will succeed with probability bounded by $\varepsilon' = \ell/T$. It follows that the obfuscator h will identify the tag to the server independently of f with probability bounded below by $1 - \varepsilon - \varepsilon'$, or $1 - \ell/T - \varepsilon$, which is non-negligible if n is large enough, since ℓ/T must eventually approach 0—as ℓ is constant, and T is not, as functions of n.

Since we are assuming that every RFID tag in our challenge-response protocol can obfuscate its identifier, but only the back-end server can disambiguate it, h must be a public-key one-way function (it must have a trapdoor that only the back-end server possesses).

5. MITIGATING PRIVACY IN SUPPORT OF AVAILABILITY

In this section we weaken the requirement for unlinkable privacy while maintaining scalability for the back-end server. We first observe that for tag responses to be linked certain patterns must be detectable. This can happen in different ways. For example, the adversary may succeed in detecting patterns after having corrupted some tags. Alternatively the adversary may destabilize tags so that they cannot be recognized by the back-end server in scalable time, thus forcing them into using responses with detectable patterns (e.g., re-using pseudonyms), or forcing the server into a linear key-lookup search.

The Molnar-Soppera-Wagner [18] protocol discussed earlier is an example of an anonymous RFID protocol for which tag responses may be linked if some tags get corrupted. Most of the other anonymous RFID protocols proposed in the literature rely on state synchronization—see e.g., [3, 9, 15, 14, 24, 19, 23]. State synchronization protocols require an extra pass to confirm state changes, and are subject to the well known "two generals' attack" problem: the server (tag) can never be certain of the next state of the tag (server) when the adversary controls the communication channels. These protocols are prone to disabling attacks. In the worst case, tags cannot be directly recognized by the server, which must then run a linear search through the key-lookup database for each disabled tag. To mitigate this DoS attack, tags may reuse earlier pseudonyms (if these are still recognizable), or as a last resort, reveal their identity (not their secret key).

A desirable privacy compromise is to minimize the loss of privacy. For example, to restrict linkability to those periods when the tag is attacked. One of the most effective disabling attack is the *entrapment* attack, in which the tag is prevented from communicating with authorized readers and can only be interrogated by the adversary. Entrapment is not necessarily physical although it does imply the ability to locate or track, since tags communicate autonomously in wireless mode.

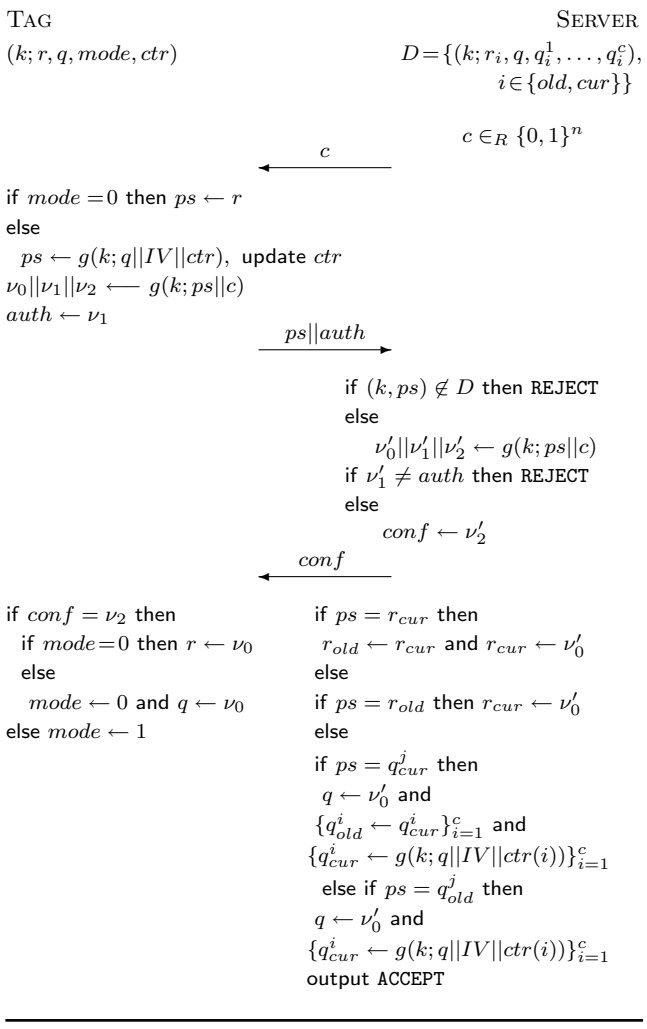
Attacks of this kind on privacy prompt us to revisit the definition of identification. The traditional approach in cryptography is to define identification as entity authentication [17]. However in more general applications it is clear that identification has a broader scope and interpretation. Entities are typically identified by their attributes, which may not involve authentication. For example, for a criminal anonymity means not being visually identified by a witness: if the criminal is identified then he/she will most likely be found guilty in a court of law even though the witness may not have checked the criminal's ID while the crime was committed. It is therefore necessary to interpret RFID privacy in the context of its application.

5.1 An Anonymous RFID Authentication Protocol with Constant Key-Lookup

We present a scalable solution for RFID authentication in which anonymity is established by synchronizing states. Our solution will allow a determined adversary to link tags during an entrapment attack, but this will not extend beyond such attacks. More specifically, although the adversary may succeed in linking tags during an entrapment session, this information will be independent for each entrapment session, thus minimizing the loss of privacy.

The protocol is described in Fig. 2, and has three passes. It is based on O-FRAP, a protocol proposed by van Le–Burmester–de Medeiros [24], which is an optimistic

Figure 2: An anonymous RFID authentication protocol that supports constant key-lookup.



forward-secure RFID authentication protocol. To simplify the description of our protocol in this paper, we shall drop the requirement for forward-secrecy: however the required changes to recapture this functionality are straightforward (and are discussed in Section 5.5).

5.2 Trusted Setup and Key-Lookup Database

Each tag is initially assigned a unique tuple $(k; r, q)$ of random values of bit-length n, the security parameter, that is stored in non-volatile memory: k is its secret key, r a one-time pseudonym and q a seed for generating entrapment pseudonyms. The tag is also given a boolean variable *mode* and a cyclic counter *ctr* that takes c distinct values, c a small constant. The protocol uses an appropriate PRF g to generate values for the pseudonyms, the authenticators and for confirmation. For each tag, the server stores in a key-lookup database DB a tuple: $(k; r_i; q; q_i^1, \ldots, q_i^c), i \in_R \{old, curr\}$, with the q a seed and the q_i^j pseudonyms used during entrapment attacks. Initially: $q = q_0$, $r = r_i = r_0$, and $q_i^j = g(k; q_0||IV||ctr(j))$, $i \in \{old, cur\}$, $1 \leq j \leq c$, where q_0, r_0 are random values, IV is an initial vector and $ctr(j)$ is the j-th value of *ctr*. The server stores pairs (old, cur) of values in DB to maintain state coordination with the tags. We assume that DB is indexed by each of the $2c + 2$ pseudonyms so that the cost of disambiguating a pseudonym is constant.

The state of each tag is controlled by the variable *mode*: if the tag is subject to an attack $mode = 1$, otherwise $mode = 0$. More specifically, $mode \leftarrow 1$ whenever the tag fails to receive confirmation to its response from a challenge, while $mode \leftarrow 0$ when confirmation has been received. When in $mode = 1$, the tag employs entrapment pseudonyms computed on-the-fly by evaluating $g(k; q||IV||ctr)$. Since there is only a constant number of such values (eventually they recycle) the tag has to defend itself against fly-by attacks by rogue readers that seek to exhaust these values. The simplest defense is to use a *time-delay* mechanism as described in [24]. This will extend the recycle time by a few orders and thwart such attacks, but may fail to deal with entrapment attacks, for which eventually the tag responses will be linked. However, unlinkability will be restored the moment the tag gets mutually authenticated by the server: on receiving confirmation from the back-end server the tag will update its seed q.

5.3 Protocol Description

We refer to Fig. 2. In the first pass the tag is challenged by the server with a random c. If the tag received confirmation in its previous interaction ($mode = 0$) then it will update its pseudonym $ps \leftarrow r$ and compute three values ν_0, ν_1, ν_2, of equal length, by inputting (k, ps, c) to the PRF g: ν_0 is kept for later use as a pseudonym update; $\nu_1 = auth$ is an authenticator and $\nu_2 = conf$ is used for confirmation. The tag responds with $ps||auth$. If the tag has not previously received confirmation ($mode = 1$) then it uses a different pseudonym in its response, computed on-the-fly with seed q.

The server uses the key-lookup database DB to disambiguate ps, and then checks $auth$. If correct, it sends $conf$ to the tag. The server then proceeds with pseudonym updates, that have to be synchronized with those of the tag: $ps = r_{cur}$ corresponds to the case when the tag is not under attack; $ps = r_{old}$ and $ps = q_i^j$, $i \in \{old, cur\}$, correspond to cases when the tag did not receive confirmation, with the last one indicating that the tag was (also) previously interrogated by an unauthorized reader (an entrapment attack). In this case the server will use a new seed $q \leftarrow \nu_0$ to update the pseudonyms q_{cur}, to support unlinkability between entrapment attacks.

If the tag receives confirmation then it will update the pseudo-nym r if in $mode = 0$, otherwise it updates the seed q. What distinguishes this protocol from O-FRAP [24] is that, at all times in this protocol, the values r, q stored by the tag in its non-volatile memory are synchronized with those stored by the server in DB. Consequently, the tag can be identified with constant key-lookup.

5.4 Security Considerations

The protocol in Fig. 2 addresses disabling attacks by weakening the requirement for unlinkable privacy. However linkability is restricted to entrapment attacks in which the tag is either physically restricted or closely tracked. During such attacks it is reasonable to assume that the tag is monitored and therefore, to some extend, already identified or located.

Our protocol is based on O-FRAP [24] that is proven se-

cure in the UC framework. From a security point of view the main difference with our protocol is its functionality: it uses entrapment pseudonyms that will eventually recycle. However these pseudonyms remain pseudo-random until they get exhausted. In the full version of this paper we shall detail the changes that have to be made to the security proof of O-FRAP to get a proof. We shall show:

THEOREM 1. *Let m be the number of uninterrupted interrogations that the adversary can make to a tag. (During an entrapment session, where the tag does not have an opportunity to interact with an authorized reader.)*

1. *If m is bounded by a constant then: the protocol in Fig. 2 realizes one-way authentication with* strong privacy *in the UC framework and constant key-lookup.*

2. *If m is not bounded then: the protocol in Fig. 2 realizes one-way authentication with* linkable privacy *in the UC framework and constant key-lookup.*

5.5 Implementation and extensions

Our protocol requires only the use of a PRF which as pointed out in Section 3.4 can be implemented with a PRNG. This allows for very efficient implementations. In particular the protocol can be adapted to conform with the EPC Gen2 standards [10]. However this protocol does not support forward-secrecy. To capture this functionality we can adapt it so that as in O-FRAP [24] the key is updated whenever the pseudonym is. The tag will require additional non-volatile memory for key update storage, but otherwise the same basic circuit can be used. One can also capture key-exchange, by using O-FRAKE [24], which is a key-exchange extension of O-FRAP.

6. CONCLUSION

In order for RFID systems that support strong security and privacy to become a reality, a well-rounded practical solution that also considers threats to availability, and which supports scalability, is needed. In this paper, we have introduced a scalability compiler that transforms a challenge-response RFID authentication protocol into a similar RFID protocol that shares the same functionality and security as well as provides scalability for the back-end server (constant lookup time even in the presence of active adversaries).

We have described a particular instantiation of the compiler and illustrated its application to a family of authentication protocols with strong security features. In particular, we have shown how to achieve security and privacy with constant lookup cost within the universally composable security model. The compiler requires only several hundred additional GEs of circuit area. Moreover, the compiler preserves other properties, such as suitability for batch authentication with delayed verification by the trusted server.

We have also proven that one-way trapdoor functions have to be used to obfuscate identifiers, in RFID authentication protocols that support anonymity with constant lookup cost. Finally by weakening the restriction on unlinkable privacy, we have described a provably secure anonymous RFID authentication protocol that supports scalable lookup and minimizes the loss of privacy due to linkability.

7. REFERENCES

[1] G. Avoine and P. Oechslin. A scalable and provably secure hash-based RFID protocol. In *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2005)*, pages 110–114. IEEE Press, 2005.

[2] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS 1997)*. IEEE Computer Society Press, 1997.

[3] M. Burmester, T. van Le, and B. de Medeiros. Provably secure ubiquitous systems: Universally composable RFID authentication protocols. In *Proc. of the 2nd IEEE/CreateNet International Conference on Security and Privacy in Communication Networks (SECURECOMM 2006)*. IEEE Press, 2006.

[4] R. Canetti. *Studies in Secure Multiparty Computation and Application*. PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.

[5] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13:1:143–202, 2000.

[6] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS 2001)*, pages 136–145. IEEE Press, 2001.

[7] D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In *Proc. Advances in Cryptology (CRYPTO 1993)*, LNCS, pages 22–39. Springer, 1994.

[8] J. Daemen and V. Rijmen. *The design of Rijndael*. Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2002.

[9] T. Dimitriou. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Proc. IEEE Intern. Conf. on Security and Privacy in Communication Networks (SECURECOMM 2005)*. IEEE Press, 2005.

[10] EPC Global. EPC tag data standards, vs. 1.3. `http://www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.3.pdf`.

[11] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In *Proc. of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, volume 3156 of *LNCS*. Springer, 2004.

[12] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES implementation on a grain of sand. In *IEE Proc. on Information Security*, volume 152(1), pages 13–20, 2005.

[13] O. Goldreich, S. Goldwasser, and S. Micali. How to construct pseudorandom functions. *Journal of the ACM*, 33(4), 1986.

[14] D. Henrici and P. M. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications*, pages 149–153, 2004.

[15] A. Juels. Minimalist cryptography for low-cost RFID tags. In *Proc. Intern. Conf. on Security in Communication Networks (SCN 2004)*, volume 3352 of *LNCS*, pages 149–164. Springer, 2004.

[16] H. Lee and D. Hong. The tag authentication scheme

using self-shrinking generator on rfid system. In *Transactions on Engineering, Computing, and Technology*, volume 18, pages 52–57, 2006.

[17] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[18] D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In *Proc. Workshop on Selected Areas in Cryptography (SAC 2005)*, volume 3897 of *LNCS*. Springer, 2006.

[19] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In *Proc. RFID Privacy Workshop*, 2003.

[20] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report TR-212, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979.

[21] A. Shamir. Squash: A new one-way hash function with provable security properties for highly constrained devices such as RFID tags. In *Invited Talk, Internat. Conf. on RFID Security (RFIDSec'07)*, 2007.

[22] S. Sharma, S. Weis, and D. Engels. RFID systems and security and privacy implications. In *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, LNCS, pages 454–470. Springer, 2003.

[23] G. Tsudik. YA-TRAP: Yet another trivial RFID authentication protocol. In *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2006)*. IEEE Press, 2006.

[24] T. van Le, M. Burmester, and B. de Medeiros. Universally composable and forward-secure RFID authentication and authenticated key exchange. In *Proc. of the ACM Symp. on Information, Computer, and Communications Security (ASIACCS 2007)*. ACM Press, 2007.