Universally Composable RFID Identification and Authentication Protocols

MIKE BURMESTER
Florida State University, Tallahassee
TRI VAN LE and BRENO DE MEDEIROS
Google, Inc. 1600 Amphitheatre, Parkway Mountain View and
GENE TSUDIK
University of California, Irvine

As the number of RFID applications grows and RFID tags begin to enter many aspects of everyday life, concerns about their security and privacy become greatly amplified. At the same time, the acutely restricted and cost-sensitive nature of RFID tags rules out simple re-use of existing traditional security/privacy solutions and calls for a new generation of extremely lightweight identification and authentication protocols.

This paper describes a universally composable security framework tuned especially for RFID applications. By making RFID-specific setup, communication, and concurrency assumptions, we arrive at a model that guarantees strong security, privacy and availability properties, while permitting the design of practical RFID protocols. The framework supports modular deployment, which is most appropriate for ubiquitous applications. As an instantiation of the proposed framework, this paper describes a set of simple, efficient, secure and anonymous (untraceable) RFID identification and authentication protocols. These protocols involve minimal interaction between a tag and a reader and place only a small computational load on the tag. They also impose little computational burden on the back-end server. We show that our protocols are provably secure within the proposed universally composable framework.


1. INTRODUCTION

Radio-Frequency Identification Devices (RFIDs) are rapidly becoming ubiquitous, moving to replace barcodes as the means of product or item identification and being deployed for a variety of applications. Among other advantages RFIDs, unlike barcodes, do not require line-of-sight alignment with readers for proper scanning, and their smaller form factor occupies less physical space on the packaging of products. Increasingly, RFIDs are also being deployed for reasons that far extend the role played by traditional barcodes, and where the (limited) programmable characteristics of RFIDs are essential.

The proliferation of RFIDs into many spheres of everyday life raises numerous privacy- and security-related concerns. In particular, RFIDs could be used to fa-
cilitate automatic aggregation of data about people’s movements and/or shopping preferences through the use of covert readers to track RFID-equipped items carried by people. Moreover, as they are increasingly being adopted for security-, safety- and payment-related functions (such as bearer identification, medical information indexing and electronic payment) it becomes necessary to focus on both privacy and security aspects of RFID tags.

The RFID security and privacy challenge is formidable mainly because of the extremely limited resources available on a typical tag. While not all RFID tags are resource-poor (e.g., passport-borne tags are expected to offer public key primitives), most retail-style tags, because of their tremendous scale, need to be as cheap as possible which results in severe resource constraints. Furthermore, since security and privacy features are usually not within the main purpose of RFID tags, available resources are even fewer.

In this paper, we focus on the systematic design of RFID security/privacy protocols which yields solutions that are efficient, provably secure as well as very flexible and modular. First, we use lightweight LFSR-based pseudo-random generators to construct all cryptographic primitives, resulting in independently variable key-length and output-length pseudo-random functions that allow for full optimization of security/efficiency trade-offs. Second, provable security extends to simultaneously achieving availability (resilience against DoS attacks), untraceability (tag anonymity) and unforgeability (cloning resistance). Third, we introduce a universally composable security framework for new functionalities especially tuned for RFID applications. This allows for flexible, modular and secure re-usability of our protocols in unanticipated contexts.

2. CONTEXT
The initial purpose and application of RFID tags was to supply, upon request, an encoded identifier. An identifier might be unique, (as in passports or badges) or clustered, i.e., it identifies a type of merchandize. The most common RFID type is a passive tag that has no power source of its own and is thus incapable of any autonomous activity. Such a device is powered by the radio waves of the reader and its antenna doubles as a source of inductive power. We concentrate mainly on these weak devices, more specifically on EPC Global class 2 devices [EPC Global], since any solution developed for them can be easily generalized for less restricted settings.

Entities and Scale. A typical RFID system involves three types of legitimate entities: tags, readers and back-end servers. The latter two are sometimes treated as one entity. However, we are interested in scalable systems and consider that a practical modern RFID system must be capable of supporting tens of millions of tags. Replicating the security functionality on all readers would be insecure (since compromise of a single reader would be catastrophic) and would also pose a management nightmare (since changing any security-related parameter would require modifying all readers).

RFID Tags. The tags are attached to, or embedded in, host objects to be identified. Each tag is a transponder consisting of an RF element and a micropro-
The RF coupling element has an antenna coil to capture RF power and data. The microprocessor has small amounts of ROM for storing, among other information, the tag’s identification number, volatile RAM and (potentially) non-volatile EEPROM. A typical low-cost unable to perform public key cryptographic operations; however, it is assumed capable of computing inexpensive conventional operations such as pseudo-random primitives (PRG/PRF), and en/decryption of short amounts of data.

Readers. A tag reader is a (often portable) computing device that has resources at least comparable to those of a powerful PDA. Each reader is equipped with a transceiver consisting of an RF module, a control unit and a coupling element to interrogate the tags. A reader incorporates a radio interface to the tags and a high-level interface to the back-end server that eventually processes captured data. Note that the interface to the back-end server might not be real-time; more on this below.

Back-End Server. A back-end server is a trusted entity that maintains a database containing all information needed to identify tags, including their identification numbers. Since the integrity of an RFID system is entirely dependent on the proper (secure) behavior of the server, we assume that the server is physically secure and not attackable. It is certainly legitimate to consider privacy mechanisms that reduce the trust on the server (e.g., to mitigate the server’s ability to collect user behavior information) or to make the server’s functions auditable. In this paper however, we consider a server to be entirely trusted and do not investigate such issues. For an overview of mechanisms that can be used to deal with privacy issues concerning back-end servers of RFID systems we refer to [Sharma et al. 2003]. As far as resources, we consider the server to be a powerful computing device with ample disk, memory, communication and other resources.

Server-Reader Interaction. Multiple readers are assigned to a single server. For simplicity we assume a simple logical server that might physically be represented by multiple replicated servers. All communication between server and readers is over private and authenticated channels. Whenever necessary, we make the assumption that servers and readers maintain loosely synchronized clocks, though, in some cases, readers may not be able to communicate with servers in real time.

Reader-Tag Interaction. A reader supplies a passive tag with power through by means of electromagnetic induction, along with clock pulses and data. The energy supplied to the tag in this fashion is sufficient to power a modest amount of computation and enable the tag to commit any changes to its persistent state—of small constant length—to non-volatile storage. Once a tag is within the range of a reader, it gets powered and is ready to communicate with the reader.

Focus/Scope. This paper’s focus is on security issues at the protocol layer. We are not concerned with physical or link layer issues, such as the coupling design, the power-up and collision arbitration processes and the air-RFID interface. For details on such issues and, more generally, on standards for RFID systems, we refer to Electronic Protocol Code [EPC Global] and ISO 18000 standard [ISO/IEC]. We recognize that physical attacks, such as jamming and collision, are a major security
issue for RFID applications, and discuss some of these attacks in Section 9.

Adversarial Model. RFIDs are a challenging platform from an information assurance standpoint. Their extremely limited computational and storage capabilities imply that many techniques for securing communication protocols are not feasible, and that only lightweight approaches must be considered. Yet, the privacy and security requirements of RFID applications can be quite significant. Ultimately, security solutions for RFID applications must take as rigorous a view of security as other types of applications. Accordingly, our threat model assumes a Byzantine adversary: It controls the delivery schedule of all communication channels, and may eavesdrop into, or modify, their contents. The adversary may also instantiate new communication channels and directly interact with honest parties. However, since the reader-server channels are assumed secure, and any assumptions about reader-server time synchronization are made explicit, it is unnecessary to model adversarial interactions with reader-server channels. In addition, the adversary may interact with the environment and with (polynomially many) sessions of arbitrary protocols, executed concurrently (or in composition) with the protocol under analysis.

2.1 Priorities, Constraints and Optimizations

In the context of RFID applications, nearly every factor having impact on tag resources and capabilities is important and even crucial. To this end, we aim to minimize requirements for: (1) non-volatile RAM on the tag, (2) tag code (gate count) complexity, (3) tag computation requirements, (4) number of rounds in reader-tag interaction\(^1\), (5) message size in reader-tag interaction, (6) server real-time computation load, and (7) server storage requirements. It is easy to see that the first three items directly influence tag cost. Also, the fourth item (number of rounds and messages) is important since more rounds imply more protocol logic and, hence, higher complexity and gate count. In fact, having more than two rounds in reader-tag interaction implies that the tag must keep temporary state while the protocol executes. This necessitates either bigger non-volatile RAM and/or continuous power from the reader (while the protocol executes) to store the state in volatile RAM.

Finally, we need to avoid features currently not considered realistic for most low-cost RFID tags, such as public key cryptography, tamper-resistant shielding and an on-board clock.

2.2 Modes of Operation

We consider two modes of tag identification: real-time and batch.\(^2\) The real-time mode is the one typically considered in the literature: it involves on-line contact

---

\(^1\)The authentication protocol may involve interaction between a reader and a server, in addition to that between a tag and a reader. We are understandably less concerned about the complexity of the former.

\(^2\)We assume that the back-end server is necessary, as a reader is unable to identify/authenticate tags on its own. In situations where this assumption is false, the discussion in this section does not apply. For example, one could imagine an RFID-equipped driver’s license reader, carried by police officers, capable of storing information about all locally-issued driver’s licenses, on the order of tens of millions. Also, recent work [Tan et al. 2007] shows how to perform, under some circumstances, serverless RFID authentication.
between the reader and the server, in order to quickly identify (and optionally authenticate) the tag in question. If immediate feedback about a tag is needed — e.g., in facility access, retail or library check-out scenarios — the server must be contacted in real time.

In batch mode, a reader scans numerous tags, collects replies and sometime later performs their identification (and optionally, authentication) in bulk. From the security perspective, the batch mode seems relevant wherever immediate detection of fraudulent/counterfeit tags is not the highest-priority issue and, instead, emphasis is on security against fraudulent readers. In practical terms, however, the batch mode is appropriate when circumstances prevent or inhibit contacting the back-end server in real time. For example, consider an inventory control application, where readers are deployed in a remote warehouse and have no means of contacting a back-end server in real time. More generally, some of the following factors might prompt the use of the batch mode: (1) The server is not available in real time, either because it is down, disconnected or because readers do not have sufficient means of communication; (2) The server is available, but is over-loaded with requests, causing response time to be jittery, thus making each tag interrogation instance unacceptably slow; (3) The server is available and not over-loaded but is located too far away, causing response time to be too long; (4) The network is congested, which causes unacceptable delays; (5) A mobile/wireless reader has limited resources and, in order to conserve battery power, simply cannot afford to contact the server for each scanned tag.

3. DESIGN REQUIREMENTS

In designing our RFID protocols, we set to achieve the following goals:

—Efficiency: protocols must be as lightweight as possible. Many RFID platforms can only implement highly optimized symmetric-key cryptographic techniques.

—Optimistic Performance: The overhead should be minimal, and there should be no additional cost to the tags when the system is under attack — any additional cost should be born by those components of the system that can afford it (in our case the server).

—Privacy: most proposed RFID applications inherently require anonymity and untraceability of individual tags (or families/batches of tags). The need for privacy is particularly acute with tags used for medical purposes (e.g., embedded in human bodies) or authorization/identification purposes (e.g., passports or badges). Thus, we believe that anonymity should be treated as a core requirement. (We note that regular authentication protocols usually do not offer any anonymity support.)

—Security: more traditional security features are also very important. They include: authentication (implicit or explicit) of readers to tags, tags to readers, replay prevention, timing attack prevention, etc.

—Availability: RFID systems are vulnerable not only to classical attacks on authentication: impersonation, man-in-the-middle, etc., but also to attacks that aim to incapacitate a tag, i.e., force it into a state which it cannot recover from. Such vulnerabilities are often exacerbated by the wireless and human-imperceptible
nature of RFID tags, allowing them to be manipulated at a distance by covert readers.

—**Concurrent Security:** RFID systems are nearly always highly concurrent.\(^3\) Therefore, it is important to address security of the overall protocol (involving the RFIDs and other system entities) in concurrent environments, where it is assumed that adversary can adaptively modify communications.

—**Modularity and Reusability:** security protocols are often deployed in a variety of contexts with similar security characteristics, i.e., where adversaries are subject to similar computational and communication resource bounds. This widespread practice can nonetheless introduce vulnerabilities. For instance, protocols are often analyzed under the implicit assumption of operating in isolation, and therefore may fail in unexpected ways when used in combination with other protocols. Since RFID tags are components of larger systems, it is preferable to pursue security analysis techniques that guarantee preservation of security properties when the protocols are executed in arbitrary composition with other (secure) protocols. This type of security is provided by formalizing and analyzing the security of protocols within the universal composability (UC) framework. UC-secure protocols support modularity; they can be safely re-used in other contexts and are suitable for a variety of constrained environments.

4. **SECURITY APPROACH**

Our security approach is guided by the objective of specifying provably anonymous authentication protocols for secure RFID applications. In particular, considering that RFID applications are often components of more comprehensive systems, we focus on security frameworks that provide for security under universal composition with arbitrary applications. The choice of cryptographic primitives to implement the protocols takes into consideration the need for computationally lightweight solutions that adhere to the hardware-imposed constraints of the platform, and the protocols are designed to be scalable to a large volume of devices.

4.1 **UC Security Framework Overview**

The universal composability (UC) framework specifies a particular approach to security proofs, and guarantees that proofs that follow that approach remain valid if the protocol, say \(\pi\), is composed with others (modularity) and under arbitrary concurrent protocol executions (including with itself). The UC framework defines a real-world simulation, an ideal-world simulation, an emulation that translates protocol runs of \(\pi\) from the real-world to the ideal-world, and an interactive environment \(Z\) that captures whatever is external to the current protocol execution (Figure 1).

In more detail, the components of a UC security formalization are:

1. A mathematical model of real executions of the protocol \(\pi\). In this model, honest parties are represented by probabilistic, polynomial-time (PPT) Turing machines that correctly execute \(\pi\) as specified, and adversarial parties that can

---

\(^3\)EPC Global Gen2 systems have the theoretical potential to simultaneously identify over 1,000 tags/second [EPC Global ], an important economic factor that makes RFID deployment cost-effective when compared with systems that scan barcodes.

deviate from \( \pi \) in an arbitrary fashion. The adversarial parties are controlled by a single PPT adversary \( A \) that (1) has full knowledge of the state of adversarial parties, (2) can arbitrarily schedule the communication channels and activation periods of all parties, both honest and adversarial, and (3) interacts with the environment in arbitrary ways, in particular can eavesdrop on all communications. In the UC framework there is also an additional PPT adversarial entity \( Z \), called the environment, that generates the initial inputs of all parties, reads their final outputs and interacts in an arbitrary way with the adversary during the execution of \( \pi \).

(2) An idealized model of executions of \( \pi \), where the security properties do not depend on the correct use of cryptography, but instead on the behavior of an ideal functionality \( F_\pi \), a trusted party that all parties may invoke to guarantee correct execution of particular protocol steps. The ideal-world adversary \( \hat{A} \) is controlled by \( F_\pi \), to reproduce as faithfully as possible the behavior of the real-world adversary \( A \).

(3) A proof that, for each PPT adversary \( A \) for \( \pi \), there is a PPT simulator \( S \) that translates real-world runs of \( \pi \) in the presence of \( A \) into ideal-world protocol runs of \( \pi \) in the presence of a simulated adversary \( \hat{A} = S(A) \) such that, no environment \( Z \) can distinguish (with better than negligible accuracy) whether it is communicating with a instance of \( \pi \) and \( A \) in the real-world or with \( F_\pi \) and \( \hat{A} \) in the ideal-world. Formally,

\[
\forall A \exists S : \text{View}^{\text{real}}_Z(A, \pi) \approx \text{View}^{\text{ideal}}_Z(\hat{A}, F_\pi).
\]

The main feature of the UC framework is that the UC security of a composite system can be derived from the UC security of its components without need for holistic re-assessment.

---

\(^4\)While the UC framework can accommodate unconditional security settings, we focus in this paper on computational security that is more appropriate for lightweight ubiquitous applications.

4.2 Previous work

The UC framework formulates protocol security in terms of indistinguishability between real and ideal protocol simulations. This strategy was informally proposed by Goldwasser et al. [Goldreich et al. 1987], and more properly specified by Beaver et al. [Beaver and Goldwasser 1989; Beaver 1991b; 1991a]. Canetti was the first to consider computationally bounded adversaries in this setting, establishing the universal composability framework [Canetti 1995; 2000; 2001]. An alternative modular, formal models-type approach called reactive systems, emphasizes independent analysis of cryptography and communication layers, was proposed by Pfitzmann and Waidner [Pfitzmann and Waidner 2000; 2001].

Formal modeling via real vs. ideal simulations is being increasingly used in the analysis of cryptographic protocols, including for authentication and key-exchange [Canetti and Krawczyk 2001; Hofheinz et al. 2003; Canetti and Herzog. 2004], for zero-knowledge proofs [Canetti and Fischlin 2001; Canetti et al. 2002], and for the universe of cryptographic primitives [Laud 2005]. More recently, an RFID privacy-oriented protocol has been proven secure in a strong real/ideal setting [Ateniese et al. 2005].

Comprehensive security models such as the UC framework are relevant in the context of ubiquitous computing applications, and in particular RFID applications. For instance, Bono et al. [Bono et al. 2005] have shown that realistic, simple attacks can compromise tags that use encryption with small keys through brief interactions between attackers and the tags. Man-in-the-middle attacks—that could be implemented as brief interaction attacks—have been shown to compromise [Gilbert et al. 2005] the HB+ authentication protocol introduced in [Juels and Weis 2005], although this protocol is known to be provably secure, even in a concurrent execution setting [Katz and S.Shin 2006]. The reason is that the security analysis in [Juels and Weis 2005; Katz and S.Shin 2006] does not consider a Byzantine attacker (in particular, it does not model man-in-the-middle attacks). Other proposed protocols [Tsudik 2006; Avoine and Oechslin 2005] are vulnerable to denial-of-service attacks that can permanently invalidate the tags. In this case the protocols have been shown to provably provide anonymous secure authentication within a security model that does not consider availability threats.

In contrast, the protocols described in this work achieve anonymity, authenticity, and availability in the UC security framework.

5. UC FORMALIZATION

As noted in Section 4.1, the UC model requires both a model of real protocol executions (familiar from traditional Byzantine security models) as well as a model of ideal protocol executions. The real-world model of protocol executions simply has the honest parties execute the protocol, while adversarial parties are centrally controlled by an adversary. As in other Byzantine settings, all real-world parties, including the adversary \( A \), are PPTs. \( A \) can eavesdrop into and schedule all communication channels. It can moreover schedule the activation order of parties.

In both the real- and ideal-world simulations, the adversary interacts with the environment \( Z \), a PPT. In the UC framework, the context of a protocol execution is captured by a session identifier \( sid \). The \( sid \) is controlled by \( Z \), and reflects external

**Functionality \( F_{\text{com}} \)**

\( F_{\text{com}} \) has session identifier \( \text{sid} \), and only acts upon messages with the same \( \text{sid} \).

- **Upon input** Channel from party \( p \): Generate a unique channel identification \( c \), a record \( \text{channel}(c, p) \) and output \( c \) to party \( p \).

- **Upon input** Listen\((c)\) from party \( p \): If there is a record \( \text{channel}(c, p) \) then record \( \text{listen}(c, p) \) and send message \( \text{listen}(c) \) to \( A \).

- **Upon input** Broadcast\((c, m)\) from party \( p \): Send message \( \text{broadcast}(c, m) \) to \( A \).

- **Upon request** Deliver\((c, m)\) from \( A \): If there is a record \( \text{listen}(c, p) \) then remove this record and output \( m \) to party \( p \).

Fig. 2. Idea anonymous communication

aspects of execution, as for example, temporal and/or locational issues, shared attributes and/or keys, etc. All parties involved in a protocol execution instance share the same \( \text{sid} \). In particular, the security proof cannot make any assumptions about extraneous knowledge that may or not be available to \( Z \) through interactions with other entities (including other instances of the protocol). The environment \( Z \) is the first party to become active in any simulation, and it activates the adversary next. If the adversary \( A \) (and all other parties) become inactive, control passes to \( Z \). \( A \) and \( Z \) may interact in arbitrary ways, and the real-world simulation halts when the environment halts. \( Z \) may read the output tapes of the tags and server at any moment. \( Z \) may also monitor other arbitrary protocol sessions, thus allowing our protocol to start and run concurrently with arbitrary others.

The ideal-world, however, departs considerably from the real-world, in that honest parties are controlled by an ideal functionality. We now describe the ideal functionalities corresponding to **anonymous authentication** \( F_{\text{auth}} \) and **anonymous key exchange** \( F_{\text{ake}} \), respectively. We also describe an extra functionality, that we call **anonymous wireless communication** \( F_{\text{com}} \). This last functionality captures an (implicit) assumption in all protocols for anonymous RFID authentication, namely that the RFID communication layers provide for anonymous communication channels. In the following, each of these functionalities is described in detail.

Observe that the ideal functionality security is unconditional, and does not rely on any cryptographically primitives that are computationally secure. This is because, in the UC framework, the security has to support concurrent executions.

5.1 Wireless communication

RFIDs are transponders that communicate in a wireless medium. In such a medium, communication has the potential of being anonymous, as location, network topology, and routing strategies do not disclose the identity of the communicating parties. Accordingly, our protocols require that only the type of a communicating party—server or transponder (tag)—is revealed through the use of communication.

Any RFID security protocol that provides anonymity must assume the existence of anonymous channels. To model this requirement in the UC framework, we introduce the ideal anonymous communication functionality \( F_{\text{com}} \), see Figure 2. For clar-
ity of presentation, in the description of our functionalities, we distinguish between messages coming from protocol parties—we call these inputs, and messages coming from the adversary—we call these instructions. As the communication anonymity requirement applies to both the real and idealized protocols, our description of the real protocol in Section 6 also makes use of \( F_{\text{com}} \).

The functionality \( F_{\text{com}} \) is an incorruptible, anonymous register of available channels. Every time a party \( p \) engages in a new subsession (via input \( \text{CHANNEL} \)), \( F_{\text{com}} \) generates a new and unique channel identifier \( c \) and records the relationship between the requesting party and the channel as \( \text{channel}(c, p) \). Parties may request to use their assigned channels for receiving messages (via \( \text{LISTEN}(c) \) inputs) or to send messages (via \( \text{BROADCAST}(c, m) \)). The channels do not connect parties per se. Instead, \( F_{\text{com}} \) reports to the adversary which channels \( c \) have been created, and registered for receiving or sending, and the messages sent through the channels. Note that the adversary learns from \( F_{\text{com}} \) the type of the communicating party (server or tag) but not its identity. The adversary then instructs \( F_{\text{com}} \) (via request \( \text{DELIVER}(c, m) \)) which messages to deliver on listened channels.

5.2 Anonymous client authentication

Client authentication is a process in which one party, the server, is assured of the identity of another party, the client, by acquiring corroborative evidence. Anonymous client authentication is a special type of authentication where the identity of the client remains private to third parties that may eavesdrop on their communication or even invoke the protocol and interact with the parties.\(^5\) In the UC framework, this is captured by the parties having ideal access to an anonymous client authentication functionality, which we denote by \( F_{\text{acauth}} \). This functionality is presented in Figure 3. We first describe its basic components and attributes and then its behavior.

**Parties.** There are two types of protocol parties, server and tag. In each session, there is a single instance of a party of type server and arbitrarily many instances of type tag. The function \( \text{type}(p) \) returns the type of party \( p \) in the current session. The UC entities, such as the adversary \( A \) and the environment \( Z \), are not parties per se, though \( A \) may control several protocol parties. Upon successful completion of a subsession, the server accepts the tag as authenticated.

**Sessions.** A single session spans the complete life-time (simulation instance) of our authentication scheme. It consists of several concurrent subsessions, which are initiated by protocol parties upon receiving input \( \text{INITIATE} \) from the environment \( Z \). While the server and tags initiate subsessions, the adversary controls the concurrency and interaction between these subsessions. All parties involved in a subsession of the authentication scheme are given a unique session identifier \( sid \) by the environment \( Z \).

**Client authentication.** Successful authentication in the real-world is a result of sharing common secrets—one party can corroborate the values produced by another

\(^5\)Anonymous authentication is also used to assert that the identity of a party is hidden even from the authenticating party (the server, in our case), and only its legitimacy is established. Here we assume that the server is trusted and do not use the term anonymity in this sense.
Functionality $\mathcal{F}_{\text{Facauth}}$

$\mathcal{F}_{\text{Facauth}}$ has session identifier $\text{sid}$ and only admits messages with the same $\text{sid}$.

**Upon input** $\text{Initiate}$ from $\text{server}$: Delete existing records $\text{init}(s, \cdot)$, for all parties. Generate a unique subsession identifier $s$ and record $\text{init}(s, \text{server})$. Send $\text{init}(s, \text{server})$ to $\mathcal{A}$.

**Upon input** $\text{Initiate}$ from $\text{tag}_i$: If $\text{tag}_i$ is adversarially controlled, ignore. Else generate a unique subsession identifier $s$ and record $\text{init}(s, \text{tag}_i)$. Send $\text{init}(s, \text{tag})$ to $\mathcal{A}$.

**Upon request** $\text{Accept}(s, s')$ from $\mathcal{A}$: If there are records $\text{init}(s, \text{server})$ and $\text{init}(s', \text{tag}_i)$, then remove $\text{init}(s', \text{tag}_i)$. Output $\text{ACCEPT}(\text{tag}_i)$ to server.

**Upon request** $\text{Impersonate}(s', \text{tag}_i)$ from $\mathcal{A}$: If there is a record $\text{init}(s, \text{server})$ and $\text{tag}_i$ is adversarially-controlled output $\text{ACCEPT}(\text{tag}_i)$ to server.

Fig. 3. Ideal anonymous client authentication

as functions of the shared secrets. The choice of authentication partners is decided by the real adversary, who has full control of the network. In the ideal-world, this is emulated by invocations of the command $\text{Accept}$ at the server. The true identity of the client is given to the server, regardless of the action of the adversary. This limits the adversary to only invoke and schedule the protocols at each party.

**Anonymity.** The only information revealed to the adversary by the functionality $\mathcal{F}_{\text{Facauth}}$ is the type $\text{type}(p)$ of the party $p$, whether it is a $\text{tag}$ or $\text{server}$. The difference between $\text{tag}$ and $\text{server}$ is observable in the real-world since the server always starts the protocol. In the ideal-world it is also observable because this information is explicitly disclosed by $\mathcal{F}_{\text{com}}$.

**Activation sequence.** The environment $\mathcal{Z}$ is the first entity to be activated, and the last to halt. $\mathcal{Z}$ activates the adversary $\mathcal{A}$, and initializes all the protocol parties (server and tags). The protocol parties instantiate the protocol in the real-world. In our protocols and functionalities, the receiving party of any message or subroutine output is activated next. If no outgoing message or subroutine output is produced in the processing of an incoming message, then by convention the environment $\mathcal{Z}$ is activated.

In Figure 3, the functionality $\mathcal{F}_{\text{Facauth}}$ is activated by an $\text{Initiate}$ input from party $p$ (server or tag). If $p$ is not adversarially controlled, the message $\text{init}(s, \text{type}(p))$ is released to the adversary, where $s$ is a newly created subsession identification label, and the record $\text{init}(s, p)$ is stored locally. Successful authentication in the ideal-world is achieved by invocations of the command $\text{Accept}$ by the server. Authentication only succeeds if both parties are requesting authentication. Finally, the adversary can impersonate tags in the ideal-world by invoking the command $\text{Impersonate}(s, \text{tag}_i)$, which only succeeds if the impersonated party $\text{tag}_i$ is controlled by the adversary.

**Forward-security.** Our models and protocols can be adapted to support corruption and to provide for forward-security in that context. (See [van Le et al. 2007] for more details.) This requires the use of a larger amount of re-writable mem-
Functionality $F_{auth}$

$F_{auth}$ has session identifier $sid$ and only admits messages with the same $sid$.

Upon input $\text{Initiate from server}$: Delete existing records $\text{init}(s, \cdot)$ for all parties. Generate a unique subsession identifier $s$ and record $\text{init}(s, \text{server})$. Send $\text{init}(s, \text{server})$ to $A$.

Upon input $\text{Initiate from tag}_i$: If $\text{tag}_i$ is adversarially controlled, ignore. Else generate a unique subsession identifier $s$ and record $\text{init}(s, \text{tag}_i)$. Send $\text{init}(s, \text{tag}_i)$ to $A$.

Upon request $\text{Accept}(s, s')$ from $A$: If there are records $\text{init}(s, \text{server})$ and $\text{init}(s', \text{tag}_i)$, remove $\text{init}(s', \text{tag}_i)$, and record $\text{partner}(s, \text{server}, s', \text{tag}_i)$. Output $\text{ACCEPT}(\text{tag}_i)$ to server. Else if there is a record $\text{partner}(s', \text{server}, s, \text{tag}_i)$ then remove it and output $\text{ACCEPT}(\text{server})$ to party $\text{tag}_i$.

Upon request $\text{Impersonate}(s, \text{tag}_i)$ from $A$: If there is a record $\text{init}(s, \text{server})$ and $\text{tag}_i$ is adversarially-controlled, output $\text{ACCEPT}(\text{tag}_i)$ to server.

Upon request $\text{Kill}(s, s')$ from $A$: If there is a record $\text{partner}(s, \text{tag}_i, s', \text{server})$ then remove this record. Output $\text{DIE}$ to $\text{tag}_i$. Remove $\text{tag}_i$ from the session $sid$.

Fig. 4. Ideal anonymous mutual entity authentication (with a disabling mechanism)

ory, however, which is expensive in the context of RFIDs. In practice, it may be more convenient to discard (destroy) tags at frequent intervals than to provide for forward-security. Therefore, for simplicity of exposition, we do not consider security issues in the presence of key-compromise and tag corruption in this paper.

5.3 Anonymous mutual entity authentication

Mutual entity authentication is a process by which two parties are assured of the identity of each other. Anonymous mutual entity authentication refers to mutual authentication in which the identities of the authenticating parties remain anonymous to third parties that may eavesdrop on their communication, or even invoke the protocol and interact with the parties. (The remark in footnote 5 applies here as well.) In the UC framework, this is captured by the anonymous mutual authentication functionality $F_{auth}$, presented in Figure 4.

As in the case of client authentication, mutual authentication involves activation of the functionality $F_{auth}$ by an INITIATE input from a party $p$ (server or tag). Successful mutual authentication in the ideal-world is achieved by invoking the command ACCEPT—for each of the inputs $(s, s')$ and $(s', s)$—by a tag and the server. Impersonation, via IMPERSONATE$(s', \text{tag}_i)$ is the same.

Secure disabling mechanism. A common functionality of RFID protocols is the kill service, through which a server can permanently disable a tag. Securing this service requires the server to authenticate itself, so it is available only in the mutual authentication setting. $F_{auth}$ supports this functionality via a $\text{Kill}(s, s')$ command.
5.4 Anonymous authenticated key-exchange

The functionality for anonymous key-exchange \( F_{aake} \) is presented in Figure 5. This is a fairly straightforward extension of \( F_{auth} \). Authentic keys are computed as an additional, private output as the result of a successful subsession. Since authenticated key-exchange implies mutual authentication, it is straightforward to extend \( F_{aake} \) to incorporate a kill command, which we omit for the sake of conciseness.

**Session-key indistinguishability.** \( F_{aake} \) provides for session-key indistinguishability, in addition to all the security properties provided by \( F_{auth} \). More specifically, if the adversary \( A \) were to be given either (i) a recently exchanged session key corresponding to a fresh authentication key, or (ii) a random value of equal length, \( A \) could not distinguish the two cases. This is so because \( F_{aake} \) generates session keys at random when the authentication key is fresh—i.e., being used for the first time since the last successful authentication session completed.

6. EXAMPLE PROTOCOLS

In this section we describe four optimistic RFID authentication protocols: RIP, RAP, O-RAP, and O-RAKE. Each protocol offers anonymity under an optimistic approach that involves minimal overhead as long as the system is not under attack. Our protocols rely on a trusted setup and on the wireless communication functionality \( F_{com} \) described earlier. The protocols are lightweight enough for realistic RFID deployment scenarios; yet, they provide strong UC security and therefore suitable as modular units in other ubiquitous application contexts, such as sensor networks. The only restriction is that each component playing the role of a single tag must use separate independent keys when performing parallel authentication/key-exchange sessions.

**Trusted Setup.** This procedure is performed in a physically secure environment. For each \( \text{tag}_i \), a fresh, unique set of values \( \text{mem}(i) \) is generated (according to speci-
fications that depend on the protocol) and stored at the non-volatile section of the tag memory. A corresponding entry \( \langle i, \text{mem}(i) \rangle \) is stored at the server.

**Server Database.** In addition to tag key entries, the server maintains one or more databases (tables) to increase protocol efficiency in cases when the adversary is passive (optimistic authentication). At the beginning of a new server sub-session, a brand new table is created. Retrieving an entry with index \( idx \) in a database table \( D \) is denoted by \( D.\text{retrieve}(idx) \). This is a very efficient operation; indeed, since the maximum size of database tables is known in advance, the indexing scheme can be implemented on the basis of a hash-table lookup operation, which takes constant time for a realistic size assumption. On the other hand, in some cases, e.g., if the adversary tampers with communication, it may be necessary for the server to evaluate a function \( f \) (which is defined dynamically and therefore cannot have been pre-computed) against each tag key to try and match against a particular value \( r \). This operation is relatively inefficient costing \( O(|D|) \).

### 6.1 YA-TRAP protocol family

We first discuss the protocol YA-TRAP—Yet Another Trivial RFID Identification Protocol, re-named here RIP, that was originally proposed in [Tsudik 2006]. Next, we extend this protocol to offer tag authentication (in addition to identification); the resultant protocol is called RIP+. However, this client-authentication version is still highly vulnerable to tag disabling attacks. We discuss two avenues to address this: (1) An empirical approach that mitigates the effects of DoS attacks by constraining their effectiveness to pre-defined time intervals, while maintaining the characteristics of a 2-pass protocol and batch operation; (2) and an alternative solution, fully resistant to tag disabling attacks, called RAP—RFID Authentication Protocol. In Section 7 we show that this latter extension achieves UC-security. More precisely, for tag authentication, RAP can operate as a (batch) 2-pass protocol, UC-realizing the Anonymous Client Authentication functionality \( F_{\text{auth}} \); for supporting the kill-key functionality, or to deal with large-scale DoS attacks, RAP can work with an optional third pass, achieving the Anonymous (Mutual) Authentication functionality \( F_{\text{auth}} \) (with disabling mechanism).

In RIP, each RFID tag (or simply tag, when its identity is not relevant) is initialized with: \( \text{mem}(i) = \{ k_i, t_0, t_{max} \} \), where \( k_i \) is a tag-specific value that serves two purposes: (1) a tag identifier, and (2) a cryptographic key. Thus, its bit-size must be the greater of: that required to uniquely identify a tag (i.e., a function of the total number of tags) and that required to serve as sufficiently strong cryptographic key for the purposes of a Message Authentication Code (MAC) computation. In Section 8, we provide estimates for circuit size and power requirements to implement all required protocol primitives, demonstrating the practicality of our protocols.

The component \( t_0 \) is the initial timestamp assigned to the tag, e.g., the timestamp of manufacture. It need not be tag-unique; an entire batch of tags can be initialized with the same \( t_0 \) value. The bit-size of \( t_0 \) depends on the desired granularity of time and the number of times a tag can be authenticated. The value \( t_{max} \) can be viewed as the highest possible time-stamp. Like \( t_0 \), \( t_{max} \) does not need to be unique, e.g., a batch of tags can share this value.

Each tag is further equipped with a sufficiently strong, uniquely seeded pseudo-
random number generator (PRG) $G_{\text{tag}}$. For $\text{tag}_i$, $G_j^i$ denotes the $j$-th invocation of the unique PRG of that tag. No synchronization whatsoever is assumed as far as PRGs on the tags and either readers or servers. In other words, given a value $G_j^i$, no entity (including a server) is able to recover $k_i$ nor any other information identifying $\text{tag}_i$. Similarly, given two values $G_j^i$ and $G_k^\ell$, deciding whether $i = \ell$ should be computationally hard for any entity. Note that the circuitry needed for implementation of the PRG can be shared by the PRF, resulting in very compact implementations. Again, see Section 8 for details.

6.2 Overview of RIP

The main idea behind RIP is the use of monotonically increasing time-stamps to provide tracking-resistant anonymous tag identification. The use of timestamps is motivated by the old result of Herzberg, et al. [Herzberg et al. 1994], which we briefly summarize next.

The work in [Herzberg et al. 1994] considered anonymous authentication of mobile users who move between domains, e.g., in a GSM [Redl et al. 1998] cellular network or a wired Kerberos-secured [Steiner et al. 1988] internetwork. The technique in [Herzberg et al. 1994] involves a remote user identifying itself to the host domain by means of an ephemeral userid. The userid is computed as a (collision-resistant, one-way) hash of current time and a secret permanent userid.

A trusted server in the user’s “home” domain maintains a periodically updated hash table where each row corresponds to a traveling user. The length of the update interval is a system-wide parameter, e.g., one hour. The table can be either pre-computed or computed on-the-fly, as needed. Each row contains a permanent userid and a corresponding ephemeral userid. When a request from a foreign agent—e.g., Kerberos authentications server (AS) or ticket-granting server (TGS)—in a remote domain—or Visitor Location Registry (VLR) in a GSM setting—comes in, the home domain server looks up the ephemeral userid in the current table. (Since hash tables are used, the lookup cost is constant.) Assuming that the timestamp used by the (authentic) traveling user to compute the ephemeral userid is reasonably recent (accurate), the hash table lookup is guaranteed to succeed. This allows a traveling user to be authenticated while avoiding any tracing by foreign agents or domains.

One of the main advantages of this approach is that the home domain server does not need to compute anything on demand, as part of each request processing. Instead, it pre-computes the current hash table and waits for requests to come in. The cost of processing a request amounts to a table lookup (constant cost) which is significantly cheaper than a similar approach using nonces or random challenges. In the latter case, the server would need to compute an entire table on-the-fly in order to identify the traveling user. As time goes by, an ephemeral userid table naturally ‘expires’ and gets replaced with a new one. This is the main feature we would like to borrow for tag authentication purposes.

Although the technique from [Herzberg et al. 1994] works well for traveling/mobile users, it is not directly applicable to the envisaged RFID environment. First, a mobile user can be equipped with a trusted personal device that keeps accurate time. It can be as simple as a wristwatch or as sophisticated as a PDA. (Moreover, even without any trusted device, a human user can always recognize grossly incorrect
time, e.g., that which is too far into the future.) Such a device can be relied upon to produce reasonably accurate current time. An RFID tag, on the other hand, cannot be expected to have a clock. Thus, it is fundamentally unable to distinguish among a legitimate and a grossly inaccurate (future) time-stamp.

However, if the tag keeps state of the last time-stamp it “saw” (assuming it was legitimate), then it can distinguish between future (valid) and past (invalid) time-stamps. We capitalize on this observation and rely on readers to offer a putatively valid timestamp to the tag at the start of the identification protocol. A tag compares the time-stamp to the stored time-stamp value. If the former is strictly greater than the latter, the tag concludes that the new time-stamp is probably valid and computes a response derived from its permanent key and the new timestamp. A tag thus never accepts a time-stamp earlier than – or equal to the one stored. However, to protect against narrowing attacks\(^6\), even if the timestamp supplied by the reader pre-dates the one stored, the tag needs to reply with a value indistinguishable from a normal reply (i.e., a keyed hash over a valid timestamp). In such cases, the tag replies with a random value which is meaningless and cannot be traced to the tag even by the actual server.

6.3 RIP Protocol description

**Notation.** If the sender writes the value \(x\) to a channel, it is observed as \(x'\) by the receiver. The value \(x'\) may differ from \(x\) if corrupted by the adversary while in transit. Also, a value computed as \(x\) by the tag is computed as \(x'\) by the server (these values should coincide if the server is using the correct key and has untampered input). The protocol is illustrated in Figure 6.

The important part of the protocol encompasses the interaction between reader and tag. It consists of only two passes and two messages, with the size of the first message determined by the timestamp \(t_{sys}\) and the second – by the pseudo-random value \(h\). (In each case, the size can be quite short, see Section 8.) The latter steps (between the reader and the server) assume the existence of a secure (private and authentic) channel. Note that we do not use \(F_{comm}\) to model this transmission. Moreover, the server is assumed to talk to non-compromised (non-malicious) readers. The contents of the message created by the server as a result to validating a tag’s response depends on the application requirements. If the application allows genuine readers to identify/track valid tags, the server returns a meta-id of the tag: \(F(k_i, 0)\), where \(F(k_i, \cdot)\) denotes a pseudo-random function (PRF) with tag key \(k_i\). Otherwise, it suffices to inform the reader that the tag in question is valid.

\(G_{tag}()\) denotes the (uniquely seeded) tag’s PRG and \(D(t_{sys})\) is the server database table for the time period \(t_{sys}\). Its entries are of the form \(\langle h_i, i, \text{mem}(i) \rangle\), with \(\text{mem}(i) = \langle k_i, t_i, t_{i,max} \rangle\) as for the tags, and it is indexed by \(h_i = F(k_i, t_{sys})\).

In batch mode, the reader interrogates a multitude of tags, collects their responses and, at a later time, off-loads the collected responses, along with the corresponding

---

\(^6\)Informally, a narrowing attack occurs when the adversary queries a tag with a particular timestamp and then later tries to identify the same tag by querying a candidate tag with a timestamp slightly above the previous one.

Fig. 6. RIP (Batch mode).

Reader()

\[ t_{sys} \xleftarrow{\text{interval}} \text{Rdr\_Clock} \]

update \( D(t_{sys}) \)

\( c_{rdr} \leftarrow \mathcal{F}_{\text{com}} \cdot \text{CHANNEL} \)

\( \mathcal{F}_{\text{com}} \cdot \text{Broadcast}(c_{rdr}, t_{sys}) \)

\[ t_{sys} \leftarrow \mathcal{F}_{\text{com}} \cdot \text{LISTEN}(c_{tag}) ; \]

if \( t_{tag} < t_{sys} < t_{max} \)

\( t_{sys} \leftarrow t'_{sys} ; \ h \leftarrow F(k_{\text{tag}}, t_{tag}) \)

else \( h \leftarrow G_{\text{tag}}() \)

\( \mathcal{F}_{\text{com}} \cdot \text{Broadcast}(c_{tag}, h) \)

\[ h' \leftarrow \mathcal{F}_{\text{com}} \cdot \text{LISTEN}(c_{rdr}) \]

Reader()

\( \text{SERVER}(D) \)

\[ \text{BatchSend}((t'_{sys}, \{ h'_1, \ldots, h'_n \})) \]

foreach \( i \)

for \( j, 0 < j \leq n_i \)

if \( D(t'_{sys}).\text{retrieve}(h'_j) \) returns \( s < 0 \)

\( \text{MSG} \leftarrow \text{TAG\_ERROR} \)

else \( \text{MSG} \leftarrow \text{TAG\_VALID}(t'_{sys}, t_{tag}) \)

or \( \text{MSG} \leftarrow F(k_{\text{tag}}, 0) \) (non-anonymous)

\[ \text{MSG} \]

\( t_{sys} \) value(s)\(^7\) to the server. The server then identifies the tags. In this usage mode, RIP is highly advantageous. Even currently most efficient techniques such as the MSW protocol [Molnar et al. 2006], require the server to perform \( O(\log n) \) pseudo-random function (PRF) operations to identify a single tag, where \( n \) is the number of tags. This translates into \( O(n \log n) \) operations for \( n \) tags. Whereas, RIP would only need \( O(n) \) operations for the same task, since the same \( t_{sys} \)-specific hash-table is used for all lookups and each lookup takes constant time.

6.4 RIP+: Adding Tag Authentication

It is easy to see that RIP does not provide tag authentication: it merely identifies that a response was originally computed by a specific (identified) tag sometime

\(^7\)If tag responses are collected over multiple time intervals, the reader needs to group responses according to the \( t_{sys} \) value used.
earlier, and does not provide for freshness of response (so, could be a replay), an essential feature of strong authentication. In order to authenticate itself, a tag needs to reply to a random challenge by the reader. Obviously, \( t_{sys} \) is not random, thus, the reply in step 3 only identifies the tag.

Adding tag authentication to RIP, requires a few protocol changes. First, we can amend the initial reader-tag message to include a random challenge \( r_{sys} \). Then, we add a PRF computed as \( h_{auth} = F(k_{tag}, r_{sys}, r_{tag}) \) to the tag’s reply, where \( r_{tag} \) is the tag’s own randomizer computed via \( G_{tag}() \). Later, once the tag is identified by the server, it can be authenticated by verifying the PRF. The identification step is the same as in RIP. The resulting protocol, RIP+, is not shown since it is substantially similar to RIP. For the server, the extra cost of authentication is negligible, amounting to one PRF operation. The additional cost for the tag consists of one PRG invocation and one PRF to compute \( F(k_{tag}, r_{sys}, r_{tag}) \).

Introducing tag authentication serves another useful purpose. In the event that the tag has been previously de-synchronized (incapacitated) by a rogue reader and its \( t_{tag} \) value has been set far into the future, \( h_{auth} \) alone can be used as a fallback in order to identify and authenticate the tag. This would require the server to perform \( O(n) \) operations—for each \( tag_j \), \( 0 \leq j < n \), compute \( PRF_{k_j}(r_{sys}, r_{tag}) \) and compare with \( h_{auth} \). This side-benefit of \( h_{auth} \) is useful in mitigating DoS attacks.

This heavier load on the server is (arguably) more tolerable in batch mode. However, in online mode, it is useful to consider the possibility of allowing the server to execute a third pass to re-synchronize the wayward tags and re-enable authentication in constant time. This essentially requires reader authentication, and involves some careful considerations. To authenticate a reader, the tag must maintain state between passes 2 and 3 to store the challenge it sends to the reader. This brings up the possibility of the reader’s reply never arriving, due to reader failure or malicious interference. In our solutions we avoid this, by assuming that the tag stores this transient state in volatile memory. The protocols are safe if interrupted and such state erased: The tag may engage in new protocol instances after a half-completed protocol without this representing a threat to any of the security properties of the scheme. Even with such considerations, a 3-pass protocol increases complexity of code footprint and associated implementation costs. Therefore, we only consider this a useful feature if greater functionality is achieved.

Note that RIP and even RIP+ have serious drawbacks. RIP is vulnerable to a trivial denial-of-service (DoS) attack: the adversary can send a wildly inaccurate timestamp \( (t_{sys}) \) via a rogue reader and incapacitate a tag either fully (if the timestamp is the maximal allowed) or temporarily. On the other hand, RIP+ makes an important assumption that a given tag is never authenticated (interrogated) more than once within the same interval. This influences the choice of the interval. A relatively short interval (e.g., a second) makes the assumption realistic for many settings. However, it translates into heavy computational burden for the server, i.e., frequent computation of ephemeral tables. A longer interval (e.g., an hour) results in much lower server burden, albeit, it may over-stretch our assumption, since a tag may need to be interrogated more than once per interval. One solution is to sacrifice some untraceability in favor of increased functionality, i.e., allow a tag to iterate over the same time value \( (accept t_{sys} = t_{tag}) \) a fixed number of times, say.
This would entail storing an additional counter on the tag; once the counter for the same $t_{tag}$ reaches $m$, the tag refuses to accept $t_{sys} = t_{tag}$ and starts responding with random values. The resulting protocol would be $m$-traceable— an adversary would be able to track a tag over at most $m$ sessions, with the same $t_{sys}$ value. (Note that the adversary can track actively, by interrogating the tag, or passively, by eavesdropping on interactions between the tag and valid readers.) However, this is only an issue if a tag is under control of the adversary multiple times within the same interval which we consider an unlikely scenario. Thus, for some settings, $m$-traceability might be a reasonable compromise.

In the following section, we discuss a modification of RIP+, RAP, that achieves DoS resilience [Burmester et al. 2006].

6.5 RAP, an RFID Authentication Protocol

RAP is a 2-pass protocol suitable for batch mode utilization. However, it supports an extra (optional pass) to deal with large-scale DoS attacks, discussed above in on-line mode. It realizes the UC functionality $\mathcal{F}_{aake}$, which implies availability. The 3-pass version also supports a kill-key (secure disabling) functionality, adhering to our philosophy that adding a pass should be used only when extra functionality is desired or needed. In the first two passes the tag is authenticated as in RAP, whereas in the third pass the server authenticates the timestamp. The protocol is given in Figure 7.

In the protocol in Figure 7, the reader sends the pair $(r_{sys}, t_{sys})$, where $r_{sys}$ is a random challenge generated by the server and $t_{sys}$ is a timestamp. In batch mode, the reader can download several $r_{sys}$ values (or a seed to expand these values) in advance. The trust model in this case extends to the reader (for the duration of that batch). More generally, if individual readers are trusted to generate good quality pseudo-random challenges, $r_{sys}$ can be unilaterally generated by the reader. The server uses a database $\mathcal{D}(t_{sys})$ pre-computed at the beginning of a new subsession with timestamp $t_{sys}$. See Figure 8. This database needs to be recomputed at the beginning of each new server subsession, characterized by a new value for $t_{sys}$ and $r_{sys}$. The server iterates over $j$, and computes the pseudo-random value: $r_j || h_j || u_j \leftarrow F(k_j, r_{sys} || t_{sys})$.

When the adversary is passive, the reader gets the correct values from the tag, and the server can authenticate the tag by using the lookup table (optimistic authentication). If, however, the tag has been previously interrogated by a corrupt reader and its $t_{tag}$ has advanced beyond the current timestamp, the server can still authenticate the tag by exhaustive search as in RIP+. If desired (in real time mode), the server can choose to re-set a tag’s time-stamp ($t_{tag}$) to reflect the current time by using an optional third pass, shown within a box. In this third pass, the reader authenticates the timestamp $t_{sys}$. This step is appropriate for any period when the number of DoS attacks is beyond a certain threshold and the server would like to re-synchronize the correct timestamp for all the tags. This makes the scheme resistant to DoS while being almost as efficient as RAP. However, DoS-resistance comes at a certain price: the third pass prompts more protocol logic on the tag and requires for the tag to keep temporary state (between passes). If this is undesirable (and kill-key functionality is not needed) it is possible to use O-RAP (Section 6.6), where re-synchronization can be performed unilaterally by the server without having to communicate authenticated values to tags.

Note that in RAP, if the server authenticates a time value beyond the maximum time $t_{max}$ for the tag, the tag will update its time $t_{tag}$ accordingly, and will no longer be able to communicate. The first step in the tag operation is to validate that the current value for $t_{tag}$ is not greater than $t_{max}$. This allows for the additional kill functionality to disable tags.
Fig. 7. RAP: A DoS-resistant variant of RIP+. The optional step, to deal with large-scale DoS attacks, is shown within a box.

\[
\text{READER}(r_{sys}) \quad \text{TAG}(k_{tag}, r_{tag}, t_{tag}, t_{max}),
\]

\[
r_{sys} \xleftarrow{\text{random}} \{0, 1\}^n, \quad t_{sys} \xleftarrow{\text{interval}} \text{Rdr\_Clock}
\]

update \(D(t_{sys})\)

\[
c_{rdr} \leftarrow \mathcal{F}_{\text{com}} \cdot \text{CHANNEL}
\]

\[
\mathcal{F}_{\text{com}} \cdot \text{BROADCAST}(c_{rdr}, r_{sys} || t_{sys})
\]

\[
t'_{sys}, r'_{sys} \leftarrow \mathcal{F}_{\text{com}} \cdot \text{LISTEN}(c_{tag});
\]

\[
\text{if } t_{tag} < t'_{sys} < t_{max}
\]

\[
t_{tag} \leftarrow t'_{sys}; \quad r || h \| u \leftarrow F(k_{tag}, r'_{sys} || t'_{sys})
\]

\[
\text{else } r || h \| u \leftarrow F(k_{tag}, r'_{sys} || t'_{sys} || t_{tag})
\]

\[
\mathcal{F}_{\text{com}} \cdot \text{BROADCAST}(c_{tag}, r_{sys} || t_{sys})
\]

\[
(r'_{tag} || h') \leftarrow \mathcal{F}_{\text{com}} \cdot \text{LISTEN}(c_{rdr})
\]

\[
r_{tag} \leftarrow r
\]

if \(D(t_{sys}) \cdot \text{retrieve}(h') \) returns \(i > 0\)

\[
\text{SearchRange} \leftarrow [i, i]; \quad x \leftarrow t_{sys}
\]

else \(\text{SearchRange} \leftarrow [1, n]; \quad x \leftarrow t_{sys} || r'_{tag}
\]

for \(j \) in \(\text{SearchRange}\) do

\[
r^* || h^* || u^* \leftarrow F(k_j, r_{sys} || x);
\]

\[
\text{if } h' = h^* \text{ then output } \text{ACCEPT}(t_{tag})
\]

\[
\mathcal{F}_{\text{com}} \cdot \text{BROADCAST}(c_{rdr}, u^*)
\]

\[
u^* \leftarrow \mathcal{F}_{\text{com}} \cdot \text{LISTEN}(c_{tag})
\]

\[
\text{if } u = u^* \text{ then output } \text{ACCEPT}(\text{server})
\]

\[
t_{tag} \leftarrow t'_{sys}
\]

Fig. 8. The database table \(D(t_{sys})\) for RAP

<table>
<thead>
<tr>
<th>pseudonym</th>
<th>(h_1)</th>
<th>(h_2)</th>
<th>\ldots</th>
<th>(h_n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>(t_1)</td>
<td>(t_2)</td>
<td>\ldots</td>
<td>(t_n)</td>
</tr>
</tbody>
</table>

We shall show the following:

**Theorem 6.1.** RAP guarantees availability, anonymity, and mutual authentication in the security framework defined in Section \(\S5\). More precisely, RAP+ realizes the UC-functionality \(\mathcal{F}_{\text{auth}}\), including the client disabling functionality.

**Proof.** The proof of this theorem is provided in Section 7. \(\Box\)

6.6 O-RAP: Optimistic RFID Authentication Protocol

We now construct another variant called O-RAP—Optimistic RFID Authentication Protocol. In this protocol, shown in Figure 9, \( r_{sys} \) and \( r_{tag} \) are pseudo-random \( \kappa \)-bit values produced by the server and the tag, respectively, in order to anonymize the session and prevent replay attacks. The value \( r_{tag} \) is a pseudonym used for the optimistic identification of the tag. The value \( k_{tag} \) is the tag’s authentication key.

Upon activation by the reader, the tag computes two values \( r \) and \( h \), by applying the pseudo-random function \( F \) to \( (k_{tag}, r_{tag} || r'_{sys}) \). In O-RAP, \( r \) is used to update the pseudo-random value \( r_{tag} \); \( h \) is used to authenticate the tag. When the adversary is passive, these values correspond to the non-starred values. In particular \( h^* = h' \) and the server outputs ACCEPT. Figure 10 describes the server database. As in the previous case, the database \( D = \{ \langle r_i, i, k_i \rangle \} \), needs to be recomputed at the beginning of each new server subsession, characterized by a new value for \( r_{sys} \). The server iterates over \( i \), and computes the pseudo-random value \( r_i || h_i || u_i \leftarrow F(k_i, r_i || r_{sys}) \).

O-RAP is simpler than RAP, at the cost of not supporting kill-keys.\(^8\) The security for

\(^8\) It is common practice to define a kill operation by revealing the secret key \( k_{tag} \). This is not acceptable from a privacy standpoint, as it will compromise the anonymity of that tag’s past
Fig. 11. O-RAKE: Optimistic RFID Authenticated Key Exchange.

**SERVER** ($D$)

- $r_{sys} \leftarrow \{0, 1\}^n$; update $D(r_{sys})$
- $c_{sys} \leftarrow \mathcal{F}_{com}.\text{CHANNEL}$
- $\mathcal{F}_{com}.\text{BROADCAST}(c_{sys}, r_{sys})$

**TAG** ($r_{tag}, k^a_{tag}, k^b_{tag}$)

- $c_{tag} \leftarrow \mathcal{F}_{com}.\text{CHANNEL}$

$r_{sys} \leftarrow \mathcal{F}_{com}.\text{LISTEN}(c_{tag})$
- $r\|h\|u\|k \leftarrow F(k^a_{tag}, r_{tag}\|r'_{sys})$
- $\mathcal{F}_{com}.\text{BROADCAST}(c_{tag}, r_{tag}\|h)$

$r_{tag} \leftarrow r$

$(r'_{tag}\|h') \leftarrow \mathcal{F}_{com}.\text{LISTEN}(c_{sys})$
- if $D.\text{retrieve}(r'_{tag})$ returns $i > 0$
  - $\text{SearchRange} \leftarrow [i, i]$
- else $\text{SearchRange} \leftarrow [1, n]$
- for $j$ in $\text{SearchRange}$ do
  - $r^*\|h^*\|u^*\|k^* \leftarrow F(k^a_j, r'_{tag}\|r_{sys})$
  - if $h' = h^*$ then output $\text{ACCEPT}(\text{tag}(j), k^b_j = k^*)$
  - $\mathcal{F}_{com}.\text{BROADCAST}(c_{sys}, u^*)$

$u^* \leftarrow \mathcal{F}_{com}.\text{LISTEN}(c_{tag})$
- if $u = u^*$ then $k^b_{tag} \leftarrow k$;
  - output $\text{ACCEPT}(\text{server}, k^b_{tag})$

---

O-RAP is similar to that of RAP+:

**THEOREM 6.2.** O-RAP guarantees availability, anonymity, and client authentication in the security framework defined in Section §5. More precisely, O-RAP realizes the UC functionality $\mathcal{F}_{acauth}$.

**PROOF.** The proof of this theorem is provided in Section 7. □

### 6.7 O-RAKE: Optimistic RFID Authenticated Key Exchange

We now describe O-RAKE—Optimistic RFID Authenticated Key Exchange protocol. As can be seen from Figure 11, O-RAKE is similar to O-RAP, except that four random values $r, h, u$, and $k$ are generated by the pseudo-random function $F$ and there is a new third pass for mutual authentication and key exchange. The value $u$ authenticates the server to the tag, allowing the tag to agree on the shared key. The value $k$ defines a shared sub-session key (output value $k^b_{tag}$), that can be used to secure the communication channel between the server and the tag, e.g., to protect transmission of private information collected by the tag.

sessions. Clearly, O-RAP (or any RFID protocol) can use a second secret-key for this purpose.

Theorem 6.3. O-RAKE guarantees availability, anonymity, mutual authentication, and key exchange with session-key indistinguishability within the security framework defined in Section §5. More precisely, O-RAKE realizes the UC-functionality $F_{\text{ake}}$.

Proof. The proof of this theorem is provided in the following section. □

7. SECURITY ANALYSIS

We first prove Theorem 6.3. We then show how the proof can be adjusted for the other protocols (Theorems 6.1 and 6.2).

7.1 Observations

As stated earlier, a UC proof of security for a protocol $\pi$ requires that:

1. A trusted functionality $F_\pi$, be defined, such that the security properties of $F_\pi$ are obvious from its definition. In particular its security should not rely on the use of cryptography.

2. For every adversary $A$, there exists a simulator $S$ that translates runs of $\pi$ in the real-world in the presence of $A$ into runs of the ideal functionality $F_\pi$ in the presence of an ideal-world adversary $A^{*}$ such that: There does not exist a PPT environment $Z$ that can distinguish whether it is interacting with an instance of $\pi$ and $A$ or an instance of $F_\pi$ and $A^{*}$, with non-negligible probability.

7.2 Proofs

Proof of Theorem 6.3. We shall show that O-RAKE realizes $F_{\text{ake}}$. Recall that our protocols assume the existence of anonymous communication channels, as captured by the functionality $F_{\text{com}}$ (Figure 2). In UC terminology, we work in the $F_{\text{com}}$-hybrid model, i.e., both real and ideal simulations can count on the services of this functionality. The proof works as follows. First, the environment $Z$ is activated, instantiating the protocol O-RAKE with parties server and several tag$_i$, and in turn activating the adversary $A$. Then the simulator $S$ performs the following actions to emulate the real-world.

- $S$ simulates copies of the parties $\hat{\text{server}}$ and $\hat{\text{tag}}_i$, initialized by $Z$, and activates an adversary $\hat{A}$.
- $S$ adds or removes keys to a database $\hat{D}$ of $\hat{\text{server}}$ that contains persistent keys of adversarially controlled tags as well as transient keys of honest tags.
- $S$ faithfully translates real-world messages into their ideal-world counterparts for all protocol parties, including the adversary.
- $S$ simulates the interactions with $Z$, i.e., the externally visible part of the protocol. More specifically, it invokes $F_{\text{ake}}$ with message ACCEPT$(s, s')$ when the real-world adversary forwards unmodified inputs between honest tags and the server, and invokes IMPERSONATE$(s, p')$ when the real-world adversary succeeds in authenticating adversarially controlled tags, respectively.
- $S$ prevents honest parties from outputting ACCEPT$(\cdot, \cdot)$ in the ideal-world when $A$ tampers with messages created by honest tags.

The interactions between $S$ and $F_{\text{ake}}$ are described in detail in Figure 12. Observe that, if the function $F$ in O-RAKE is a true random function, then the transcripts exchanged in tag subsessions are uniformly random and mutually independent. Under this assumption (of a true random $F$), the real and ideal simulations might differ only when $S$ intervenes to prevent ACCEPT$(\cdot, \cdot)$ in the ideal-world, i.e., when $A$ tampers with messages from honest parties AND this results in acceptance in the real-world. Now, this can only happen
where $Adv_F(n, L, T + nL) + 2^{1-\kappa} nL$, 

where $Adv_F(q, t)$ is the advantage of distinguishing $F$ from a true random function by making at most $q$ queries to $F$ and using at most $t$ computational steps (execution time); $n$ is the number of tags; $L$ is the number of tag subsessions; and $T$ is the combined time complexity of the environment $Z$ and the adversary $A$.

Proof of Theorem 6.2. O-RAP is a restriction of O-RAKE, correspondingly exactly to the restriction of functionality $F_{\text{ake}}$ to $F_{\text{scache}}$. The adaptations to derive a proof for O-RAP are therefore straightforward.

Fig. 12. Simulator $\hat{S}$

Upon init$(s, server)$ from $F_{\text{ake}}$: Create a new subsession $s$ for server and send init$(s, server)$ to $\hat{A}$.

Upon init$(s, tag)$ from $F_{\text{ake}}$: Create a new tag subsession $s$ for a new tag named $\hat{t}_s$. Generate a random key $(r_s, k_s^a, k_s^b)$, and add it to the database $\hat{D}$ using identity $\hat{t}_s$. Send init$(s, tag)$ to $\hat{A}$.

Upon server outputting ACCEPT($\hat{p}, k$) during subsession $s$ ($\hat{p} \in \hat{D}$): If $\hat{p}$ is adversarially controlled then send IMP personate$(s, \hat{p}, k)$ to $F_{\text{ake}}$. Else let $s'$ be the tag subsession identifier such that $\hat{p} = \hat{t}_{s'}$. Generate a record partner$(s, s')$ and send ACCEPT$(s, s')$ to $F_{\text{ake}}$.

Upon tag$_{s'}$ outputting ACCEPT(server, $k$): Remove the key of $\hat{t}_{s'}$ from database $\hat{D}$, lookup record partner$(s, s')$ and send ACCEPT$(s', s)$ to $F_{\text{ake}}$. 

if there is a collision between the outputs of $F$ computed by the same party in that subsession. Since $F$ is truly random by assumption, the adversary cannot count on that happening with more than negligible probability. More concretely, for each given server subsession, this happens with probability at most $2^{1-\kappa} nL$, where $\kappa$ is the minimum bit length of $r_{\text{sys}}, r_{\text{tag}}$, and $h, n$ is total number of tags managed by this server, and $t$ is the number of tag subsessions during this server subsession. Therefore, the total probability of simulation distinguishability is bound by $2^{1-\kappa} nL$, where $L$ is the total number of tag subsessions through the entire simulation.

It follows that, if $Z$ can distinguish real from ideal simulations, it can also distinguish real simulations with the pseudo-random function $F$ from real simulations with a truly random function. This will lead to a contradiction, if $F$ is indistinguishable from random by any PPT adversaries.

Session identifiers. In our proof, we have not explicitly stated the nature of the session identifier $sid$. We now rectify this. In our protocols the $sid$ provided by the UC framework includes the tag names and their corresponding keys. This guarantees that the server and the tag share the same secret key in the same session. Naturally, without this assumption neither the security nor the functionality of our protocols is guaranteed.

Security reduction and concrete complexity. A concrete security reduction must relate distinguishing real-vs-ideal worlds to distinguishing pseudo-vs-true randomness. To accomplish this, faithfully simulate the real-world and use $Z$ as the distinguisher. The probability of distinguishing real from ideal simulations is at most the sum of: (1) Distinguishing real from ideal when a truly random function $F$ is used in the real simulation, the probability of which we have bound in the proof of Theorem 6.3 and, (2) the advantage of distinguishing $F$ from a truly random function. Therefore, the advantage of distinguishing real from ideal is at most:

$$Adv_F(n, L, T + nL) + 2^{1-\kappa} nL,$$

where $Adv_F(q, t)$ is the advantage of distinguishing $F$ from a true random function by making at most $q$ queries to $F$ and using at most $t$ computational steps (execution time); $n$ is the number of tags; $L$ is the number of tag subsessions; and $T$ is the combined time complexity of the environment $Z$ and the adversary $A$.

Proof of Theorem 6.2. O-RAP is a restriction of O-RAKE, correspondingly exactly to the restriction of functionality $F_{\text{ake}}$ to $F_{\text{scache}}$. The adaptations to derive a proof for O-RAP are therefore straightforward.

PROOF OF THEOREM 6.1. The security of RAP+ follows very similar arguments to ORAKE. The main differences are: (1) One needs to define “loose clock synchronization” between readers and the server to rely on readers generating the same value $t_{sys}$ that the server expects;\(^9\) (2) The trust model is slightly different in batch mode, as the readers transiently store trusted server values; (3) The UC formalization of $\mathcal{F}_{auth}$ extends $\mathcal{F}_{sake}$ slightly by introducing a Kill command to capture the corresponding feature of RAP+. However, it should be obvious that this additional feature can be described as a mutual authentication step (included in $\mathcal{F}_{sake}$ as well) that validates the transmitted value $t_{sys}$. The correct disabling of the tag is simply the event that condition $t_{sys} > t_{max}$ is satisfied. \(\square\)

8. IMPLEMENTATION AND EFFICIENCY CONSIDERATIONS

In this section we show how to achieve a very efficient, practical construction of each of the primitives required by our protocols (namely, PRFs/PRGs) by using only a pseudo-random generator (PRG). Estimation of the hardware requirements of a prototypical specification are of the order of 2000 gates.

8.1 Lite pseudo-random function families

First, we design a large-length output pseudo-random function (PRF) from a fixed-length output PRF and a PRG. Using ideas from [Goldreich et al. 1986] one can then implement the protocols by using a PRG only. For the sake of completeness we include a proof of security of the lemma below.

**Lemma 8.1.** If PRG is a pseudo-random generator and PRF is a pseudo-random function then $F = \text{PRG} \circ \text{PRF}$ is a pseudo-random function.

**Proof.** Let $X, Y, W,$ and $Z$ be efficiently sampleable domains and let $\text{PRF} : X \times Y \rightarrow W$ be a pseudo-random function and $\text{PRG} : W \rightarrow Z$ be a pseudo-random generator. We show that $F = \text{PRG} \circ \text{PRF} : X \times Y \rightarrow Z$ is a pseudo-random function. Indeed, let $y_1, y_2, \ldots, y_n \in Y$ be distinct values and let $x \in_R X$. We show that $\vec{z} = (F(x, y_1), \ldots, F(x, y_n))$ is indistinguishable from a random vector in $Z^n$. Observe that $F(x, y_i) = \text{PRG}(w_i)$ where $w_i = \text{PRF}(x, y_i)$. Since PRF is a pseudo-random function, the vector $\vec{w} = (w_1, \ldots, w_n)$ is pseudo-random in $W^n$. This implies that $\vec{z} = (\text{PRG}(w_1), \ldots, \text{PRG}(w_n))$ is indistinguishable from $\vec{z}^* = (\text{PRG}(w_1^*), \ldots, \text{PRG}(w_n^*))$, where $w_1^*, \ldots, w_n^*$ are randomly and independently selected from $W$. By the pseudorandomness of the distribution of $\text{PRG}(w_i^*)$ and the multi-sample indistinguishability theorem of Goldreich [Goldreich 2001] and Yao [Yao 1982], $\vec{z}^*$ is indistinguishable from a random vector in $Z^n$. \(\square\)

8.2 Practical implementation

For practical RFID implementations a very efficient hardware design for a PRG should be used. In general a PRG can be implemented much more efficiently than a standard cryptographic pseudo-random function. For instance, the shrinking generator\(^{10}\) of Coppersmith, Krawczyk, and Mansour [Coppersmith et al. 1994] can be implemented with fewer than 2000 gates with approximately 80-bit security [Batina et al. 2004], which is feasible for a wide range of RFID architectures. The best known attacks on the shrinking generator are not practical in this range of the security parameter [Batina et al. 2004].

\(^9\)Note that this is an availability concern only when the server is configured to NEVER compute new tables on demand (for unexpected values of $t_{sys}$). Otherwise, this is purely a performance issue that is irrelevant for the security proof.

\(^{10}\)Using the shrinking generator requires care (buffering) to avoid the introduction of vulnerabilities to timing and side-channel attacks.
The architecture of a variable output length PRF construction based on a PRG is very flexible. We can independently tune the key length and output bit length of the PRF to any (reasonable) values desired. This allows to fully optimize for security-efficiency trade-offs.

Using this PRG-based strategy, it is possible to obtain a full-fledged implementation of the RAP+, O-RAP, and O-RAKE by using approximately 2000–3000 gates, feasible for a wide range of RFID architectures. For comparison, most proposed RFID authentication protocols rely on standard cryptographic constructions, such as those based on HMAC (with the extra property that the cryptographic hash function in the construction should pseudo-random), or CBC-MAC with a block cipher (for instance, AES) would require around 8-15K gates. These constructions are suitable only for a narrow range of higher cost RFID tags.

Alternatively, secure stream ciphers suitable for constrained hardware architectures could be used—some candidates have been submitted to the European eStream project [Network of Excellence within the Information Societies Technology (IST) Programme of the European Commission]. However, designing such highly efficient stream ciphers remains challenging. For example, the proposed Grain family of stream ciphers [Hell et al. 2005] has recently been shown not to achieve full security [Maximov 2006].

9. EXTENSIONS AND OPEN QUESTIONS

9.1 Corruption of Tags

In case a tag is compromised, it is natural to consider the privacy of past occurrences (sessions) involving the same tag. Ideally, the compromise of a tag yields no information about its prior life. We refer to this property as forward privacy. (This is a notion different from PFS – Perfect Forward Secrecy – encountered in key exchange protocols. PFS aims to preserve the secrecy of past session keys. In contrast, forward privacy aims to preserve the anonymity/untraceability of past sessions.) To obtain forward privacy, tags and the trusted server must periodically evolve their respective secret keys. However, extending the UC security model to express issues of forward privacy introduces several complexities (see [van Le et al. 2007]). Although it is an important feature, we consider forward privacy to be outside the scope of this paper.

9.2 Timing Attacks

In our security model the tags and the trusted server take exactly one computation step between sending and receiving authentication data. A secure implementation should reflect this semantic. In particular the time taken for each pass must be constant. This is a bigger concern when the protocols are executed in on-line mode, since in batch mode the server extra computational time can be amortized, and moreover delays cannot be assigned to any particular tag in the batch. One solution is to insert an artificial delay on the trusted server. This does not effect the throughput and workload of the server, which is the objective of our scalable optimistic protocols.

9.3 Reducing Server Search

Our optimistic anonymous RFID protocols are very efficient in the absence of active attackers. However, in the presence of such adversaries, the server is required to perform a linear search on all registered keys. This can be mitigated by assigning multiple, replicated

\[ 1 ]^{11} \text{The attack succeeds in } O(2^{54}) \text{ steps, while Grain promises 80-bit security. However, the attack requires considerable amount } (O(2^{51}) \text{ bits}) \text{ of keystream (alternatively, plaintext/ciphertext pairs), an unrealistic amount of data in the context of RFID applications.}

keys to tags, with the effect of a (at most) linear cost increase for the tags while the server search space decreases exponentially.

A different approach to reduce the server search and mitigate DoS vulnerabilities in the RIP protocol family has been recently proposed [Anonymized under request]. It entirely avoids the use of server challenges, and instead the time $t_{sys}$ released by the server is authenticated directly through the use of a shared hash chain. More precisely, time stamps are aggregated in larger intervals, which are made to correspond to values in a hash chain that is pre-computed by the server. The (secret) root of the chain is stored in the tags. At the beginning of a new interval the authenticator (chain value) for that period is released and, with this value, readers are able to make tags accept and respond to time stamps within the interval. The significant advantage of this approach is that the server can now authenticate each tag in constant time always. The disadvantage is that an attacker can still temporarily disable a tag by releasing the maximum valid time stamp within the current interval. It also restricts the tag to a single authentication per period, or alternatively $n$ authentications if $n$-traceability is tolerable.

Both of these approaches increase the vulnerability of the scheme to key compromises. Accommodating optimal efficiency at the server—i.e., abandoning the optimistic framework—while avoiding extra complexity in the tag (in particular, if 2-pass protocols are required) is an open question of considerable practical interest.

A third approach to reduce the server search—the key-lookup problem, is proposed in [?]. This uses a public key obfuscator to anonymize tags, whose secret key (trapdoor) is known to the server. Although the search complexity is reduced to constant, there is an extra cost for the tags: the logic for obfuscation is at least 1,000 GEs, and the tag responses are longer—resulting in extended time responses.

9.4 Relay attacks

Our formal security model does not exclude simple relay attacks in which an adversarial tag, $t_{adv}$, in the range of the server can impersonate an uncorrupted tag, $t_{i}$, that is out of the range of the server, by relaying the server’s challenge to $t_{i}$, and the response of $t_{i}$ to the server (and the server’s confirmation to $t_{i}$, in the three pass protocol). Since the server’s interrogation in our protocols is temporal, relay attacks have to be on-line. If these attacks are of concern, and cannot be addressed by other means (e.g., physical), they can be mitigated by shortening the lifespan of the server’s challenge. For a discussion on relay attacks against RF security devices see [Reid et al. 2007; Kfir and Wool 2005].

9.5 Side-channel attacks

Side-channel attacks, in particular power-consumption cryptanalysis, have been shown to be extremely effective, completely recovering cryptographic keys [Oren and Shamir 2006]. In order to achieve strong security in practice, research is needed into either making RFID hardware more tamper-proof (e.g., via Faraday cages), or into the development of obfuscating techniques for cryptographic computations in RFIDs.

From a theoretical standpoint, an interesting open question is whether physical security can be modeled within a UC framework, e.g., by introducing information leakage channels and proving that such cannot give an advantage to adversaries, even in arbitrary composition and concurrency settings.

10. CONCLUSION

We presented a new, universally composable framework that facilitates the design of secure and efficient of RFID identification and authentication protocols. We instantiate the framework by describing several provably secure, optimistic RFID authentication proto-
cols: RIP+, RAP, O-RAP, and O-RAKE.

REFERENCES

ANONYMIZED UNDER REQUEST. A family of dunces: Trivial RFID identification and authentication protocols. Under review.


