Universally Composable and Forward Secure RFID Authentication and Authenticated Key Exchange

Tri van Le, Mike Burmester, Breno de Medeiros
Department of Computer Science, Florida State University
Tallahassee, Florida, FL 32306-4530, USA
Email: {levan,burmester,breno}@cs.fsu.edu

December 5, 2006

Abstract

Protocols proven secure in universally composable models remain secure under concurrent and modular composition, and may be easily plugged into more complex protocols without having their security re-assessed with each new use. Recently, a universally composable framework has been proposed for Radio-Frequency Identification (RFID) authentication protocols, that simultaneously provides for availability, anonymity, and authenticity. In this paper we extend that framework to support key-compromise and forward-security issues.

We also introduce new, provably secure, and highly practical protocols for anonymous authentication and key-exchange by RFID devices. The new protocols are lightweight, requiring only a pseudo-random bit generator. The new protocols satisfy forward-secure anonymity, authenticity, and availability requirements in the Universal Composability model. The proof exploits pseudo-randomness in the standard model.

1 Introduction

While admittedly a new technology, RFIDs have great potential for application in the enterprise and/or as smart, mass-market, embedded devices. The security implications are considerable, and moreover important characteristics distinguish RFID authentication models from general-purpose authentication, as we elaborate on below.

- *Lightweight.* RFID authentication protocols must be lightweight. Many RFID platforms can only implement highly optimized symmetric-key cryptographic techniques.
- *Anonymity.* General-purpose authentication protocols may or not have support for anonymity. On the other hand, many proposed RFID applications typically require anonymity fundamentally, for instance for devices embedded in human bodies or their clothes, documents, etc. So anonymity should be considered a core requirement of RFID authentication protocols.
- *Availability.* When considering RFID authentication protocols, one should examine not only their vulnerability to attacks on authentication—impersonation, man-in-the-middle, etc—but also to attacks that force the RFID device to assume a state from which it can no longer successfully authenticate itself. This is of particular relevance for RFID devices, which are portable and can be manipulated at a distance by covert readers.

- *Forward Security.* As RFID devices can be lost or discarded, forward security is important to guarantee privacy of past transactions that happened before the long term key or the session key is exposed.

- *Concurrent Security.* As authentication and key exchange protocols are typically used as a component in larger secure protocols, the issue of maintaining security of the overall protocol in such concurrent environment with strong adversary that can adaptively modify communications is of importance [26, 28, 19].

Our goals are to design authentication protocols that will be used as subprotocols in ubiquitous applications, or as standalone applications in combination with other applications. Thus we desire to have our protocols be analyzed only once and then can be applied universally without having to reanalyze the protocols for each application. Security analysis in the universal composability framework allow us to achieve this goal. Our security is based on notions of interactive indistinguishability of real from ideal protocol executions. This approach requires the following components:

1. A mathematical model of real protocol executions, where honest parties are represented by probabilistic polynomial-time Turing machines that correctly execute the protocol as specified, and adversarial parties that can deviate from the protocol in an arbitrary fashion. The adversarial parties are controlled by a single (PPT) adversary that (1) has full knowledge of the state of adversarial parties, (2) can arbitrarily schedule the actions of all parties, both honest and adversarial, and (3) interacts with the environment in arbitrary ways, in particular can eavesdrop on all communications.

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

3. A proof that no environment can distinguish (with better than negligible accuracy) real- from ideal-world protocol runs by observing the system behavior, including exchanged messages and outputs computed by the parties (honest and adversarial). The proof works by translating real-world protocol runs into the ideal world.

An important separation between theory and practice is efficiency. We design our protocols to minimize security overheads when the system is not under attack. Achieving this goal together with availability and forward security in a lightweight manner suitable for RFIDs is a nontrivial task, as witnessed in the literature (see a review of the literature in Section §2).

Our contributions.

- A new UC authentication framework, including anonymity and forward security (Section §4.)

- New protocols that provide for optimistic forward-anonymous authentication and that guarantee availability and minimize security overhead in the honest case (Sections §4).

- Lightweight implementation of the protocols in a wide-variety of RFID architectures by using only PRGs (Section §6.)
- Featherweight PRG-based protocols that achieve identical security guarantees with a simpler architecture under the assumption that the adversary has only time-limited opportunities to interact with tags ("fly-by" attacks, in Section §7.)
- Security proofs for the protocol families (Sections §5, §6, §7.)

2 Previous work

The need for lightweight security mechanisms in RFID applications does not imply that one can afford to provide security under limited attack models, as there is no reason to expect that attackers will also have limited resources. For instance, Green et al. [8] have shown how realistic, simple attacks can compromise tags that use encryption with small keys—even though only brief interactions between attackers and the target tag ever take place—we shall call such limited-interaction attacks *fly-by attacks*. Proposed protocols, some very ingenious [26], and which moreover enjoy strong security properties under limited attack models [28] have been shown to be vulnerable to man-in-the-middle-attacks [19] that could be implemented as fly-by attacks. Other interesting protocols, such as YA-TRAP [35], use timestamps. While effective in reducing complexity, the use of timestamps leaves the tags vulnerable to denial-of-service attacks that can permanently invalidate the tags, as pointed out by G. Tsudik in [35].

The research literature in RFID security, including anonymous authentication protocols, is already quite extensive and growing—for reference, a fairly comprehensive repository is available online at [2]. Here, we shall refrain from a comprehensive review and focus consideration on those works most directly related to our construction. Ohkubo et al. [32] proposed a hash-based authentication protocol that bears close resemblance to our protocols. However, the scheme in [32] is vulnerable to certain re-play attacks. The proposed modifications in [3] address the replay-attack problem but does not consider the issue of *availability*, and their scheme is vulnerable to attacks where the attacker forces an honest tag to fall out of synchronization with the server so that it can no longer authenticate itself successfully. Dimitriou [18] also proposes an anonymous RFID protocol vulnerable to de-synchronization attacks against availability.

Another hash-based authentication protocol is introduced by Henrici et al [23]. Their solution does not provide full privacy guarantees, in particular, the tag is vulnerable to tracing when the attacker interrupts the authentication protocol mid-way. Molnar et al [30] propose a hash-tree based authentication scheme for RFIDs. As most tree-based scheme, the amount of computation required per tag is not constant, but logarithmic with the number of tags in the hash-tree. Also, if a tag is lost, anonymity for the rest of the hash-tree group may be compromised. Finally, the scheme does not provide for forward-anonymity. A scheme by Juels [25] only provides security against "fly-by" attacks where the attacker is allowed to interact with the tag for a fixed time budget but does not provide protection in the case of tag capture.

Our proposed solutions address all these issues within a comprehensive security framework. We note that relatively little work has been done on RFID protocols where security is provided in a unified model (for examples, see [1, 9]). Admittedly, in the RFID setting, one should be aggressive in making simplifications to security models that *are* justified, as in such a constrained environment some tradeoffs are needed in order to minimize the complexity and maximize the efficiency of the designed solution. One such restriction that we adopt is that we prohibit tags from parallel execution of authentication protocols (note that the prohibition does not extend to corrupt parties or non-tag entities). This restriction is readily relaxed when tags use multiple separate keys for concurrent executions.

In this paper we articulate security models for anonymous RFID authentication and key exchange protocols. These models extend the framework introduced in [9] in several ways. We support session-key compromise and replacement, extending the model to support analysis of key-exchange protocols ([9] considers only authentication). In [27], an alternative anonymity definition is proposed, following a traditional adversary-game approach (i.e., without consideration for composability issues).

The proposed model defines security in terms of indistinguishability between real and ideal protocol simulations, an approach first outlined by Beaver [7, 6, 5], and extended by Canetti as the universal composability framework [10, 11, 12]. A similar approach has also been pursued by Pfitzmann and Waidner [33, 34], under the name *reactive systems*. Several protocols have been proposed under the UC framework, including authentication and key-exchange [15, 24, 14], zero-knowledge proofs [13, 16], and other cryptographic primitives [29]. More recently, an RFID privacy-oriented protocol has been proven in the UC setting [1].

3 Ideal functionalities

In this section, we introduce three ideal functionalities, the anonymous (forward-secure) entity authentication functionality $\mathcal{F}_{\mathsf{aauth}}$, the anonymous (forward-secure) key-exchange functionality $\mathcal{F}_{\mathsf{aake}}$, and finally the anonymous communication functionality $\mathcal{F}_{\mathsf{com}}$. In the following, each of these functionalities is described in detail.

3.1 Anonymous entity authentication functionality

Entity authentication is a process in which one party is assured of the identity of another party by acquiring corroborative evidence. Anonymous authentication is a special type of entity authentication where the identities of the communication parties remain private to third parties that may eavesdrop on their communication or even invoke and interact with the parties. In the UC framework, it is captured by the parties having ideal access to a anonymous entity authentication functionality, which we denote by $\mathcal{F}_{\mathsf{aauth}}$. This functionality is presented in Figure 1.

Party. In our functionality, 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 $type(p)$ returns the type of party p in the current session. The adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ are considered system parties.

Session. A single session spans the complete life time of our authentication scheme. It consists of many concurrent subsessions, which are initiated by protocol parties upon receiving input INITIATE from the environment $\mathcal{Z}$. While the server and tags initiate subsessions, the adversary controls the concurrency and interaction between these subsessions. Two protocol parties are feasible partners in authentication if they are composed of a `tag` and a `server`. Upon successful completion of a subsession, each party accepts its corresponding partner as authenticated. The environment $\mathcal{Z}$

Functionality $\mathcal{F}_{\mathsf{aauth}}$

$\mathcal{F}_{\mathsf{aauth}}$ has session identity *sid* and only admits messages from the same session *sid*.

Upon receiving input Initiate **from protocol party** p : if party p is corrupted then ignore this message. Else generate a unique subsession identification s, record $\texttt{init}(s, p)$ and send $\texttt{init}(s, \mathit{type}(p), \mathit{active}(p))$ to the adversary.

Upon receiving message Accept(s, s') **from the adversary:** if there are two records $\texttt{init}(s, p)$ and $\texttt{init}(s', p')$ such that parties p and p' are feasible partners, then remove these records, record $\texttt{partner}(s', p', s, p)$ and write output $\texttt{ACCEPT}(p')$ to party p. Else if there is a record $\texttt{partner}(s, p, s', p')$ then remove this record and write output $\texttt{ACCEPT}(p')$ to party p.

Upon receiving message Impersonate(s, p') **from the adversary:** if there is a record $\texttt{init}(s, p)$ and party p' is corrupted then remove this record and write output $\texttt{ACCEPT}(p')$ to p.

Upon receiving message Corrupt(s) **from the adversary:** if there is a record $\texttt{init}(s, p)$ or $\texttt{partner}(s, p, s', p')$ such that p is corruptible then mark p as corrupted and remove $\mathit{state}(p)$.

Figure 1: Ideal anonymous authentication.

may read the output tapes of the tags and server any moment during the session, which terminates when the environment $\mathcal{Z}$ stops. The environment $\mathcal{Z}$ may contain many other sessions of arbitrary protocols, thus allowing our protocol to start and run concurrently with arbitrary others.

Authenticity. Successful authentication in the real world is a result of sharing *common secrets*—one party can corroborate the values produced by another 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 Accept, one for each partner. The true identity of the partner is given to the authenticating parties, regardless of the action of the adversary. This limits the adversary to invocation of the protocols and scheduling of the output of each party only.

Anonymity. The only information revealed to the adversary by the functionality is the type of the party, whether it is a `tag` or `server`. The observable difference is due to the fact that the real server always starts the protocol.

Forward security. The real adversary may corrupt parties—only tags, the server is incorruptible—obtaining keys and any persistent memory values. These may compromise the anonymity of the current subsession and earlier incomplete ones. In order to corrupt a tag not actively running, the environment $\mathcal{Z}$ may request the tag to start a new subsession and then inform the adversary to corrupt it.

The effect of corruption in the ideal world, via command Corrupt, is that the adversary can impersonate corrupted tags, via Impersonate command, and link all incomplete subsessions up to the last successfully completed one, via *active*(p)—the list of identifications of preceding incomplete subsession, returned from the functionality after a Initiate command. Once a subsession is successfully completed in the ideal world, this subsession and all earlier subsessions of the same party are protected against all future corruptions of any party.

In the functionality, $state(p)$ is the list of all subsession records maintained by the functionality concerning party p in the current session. This list is removed from the memory of ideal functionality up on corruption of the tag p, effectively leaves control of the corrupted tag to the adversary. The only information retained is the fact that p is corrupted.

Activation sequence. In our protocols and functionalities, the receiving party of any message or subroutine output is activated next. If no out going message or subroutine output is produced in the processing of an incoming message, then by convention the environment $\mathcal{Z}$ is activated next.

3.2 Anonymous authenticated key-exchange functionality

Functionality $\mathcal{F}_{\mathsf{aake}}$

$\mathcal{F}_{\mathsf{aake}}$ has session identity *sid* and only admits messages from the same session *sid*.

Upon receiving input INITIATE **from protocol party** p : if party p is corrupted then ignore this message. Else generate a unique subsession identification s, record $\texttt{init}(s, p)$ and send $\texttt{init}(s, type(p), active(p))$ to the adversary.

Upon receiving message ACCEPT(s, s') **from the adversary:** if there are two records $\texttt{init}(s, p)$ and $\texttt{init}(s', p')$ such that parties p and p' are feasible partners, then remove these records, generate a random key k, record $\texttt{partner}(s', p', s, p, k)$ and write output $\texttt{ACCEPT}(p', k)$ to party p. Else if there is a record $\texttt{partner}(s, p, s', p', k)$ then remove this record and write output $\texttt{ACCEPT}(p', k)$ to party p.

Upon receiving message IMPERSONATE(s, p', k') **from the adversary:** if there is a record $\texttt{init}(s, p)$ and party p' is corrupted then remove this record, and write output $\texttt{ACCEPT}(p', k')$ to p.

Upon receiving message CORRUPT(s) **from the adversary:** if there is a record $\texttt{init}(s, p)$ or $\texttt{partner}(s, p, s', p', k)$ such that p is corruptible then mark p as corrupted and remove $state(p)$.

Figure 2: Ideal anonymous authenticated key exchange.

The functionality for anonymous key-exchange $\mathcal{F}_{\mathsf{aake}}$ is presented in Figure 2. This functionality is a fairly straightforward extension of $\mathcal{F}_{\mathsf{aauth}}$. As in the previous case, parties include a server, tags, and an adversary. Authentic keys are computed as an additional, private output at the result of a successful subsession.

$\mathcal{F}_{\mathsf{aake}}$ is activated by an INITIATE input from a party belonging to the session. The list of existing subsessions since its last successfully completed subsession are released to the adversary via message $\texttt{init}(s, type(p), active(p))$, where s is a newly created subsession identification. $\mathcal{F}_{\mathsf{aake}}$ also stores locally the record $\texttt{init}(s, p)$.

Corruption is as in the entity authentication functionality. It is achieved by the adversary invoking the command CORRUPT. Again, successful authenticated key exchange in the real world is a result of sharing secrets. This is achieved in the ideal world by invocations of the command ACCEPT by the ideal adversary, one for each partner in the pair. This only succeeds if the two parties are both requesting authentication. Successful subsessions result in each party accepting the partner's true identity and generating a shared subsession key.

As in the previous case, the adversary can impersonate parties in the ideal world by invoking the command IMPERSONATE, which only succeeds if the impersonated party is corrupted.

3.3 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. This is a common assumption of many RFID security protocols.

Functionality $\mathcal{F}_{\mathsf{com}}$

$\mathcal{F}_{\mathsf{com}}$ has session identity *sid*. It only admits messages from the same session *sid*.

Upon receiving input CHANNEL **from party** p**:** generate a unique channel identification c, a record `Channel`(c, p) and write output c to and reactivate party p.

Upon receiving input LISTEN(c) **from party** p**:** if there is a record `Channel`(c, p) then record `Listen`(c, p) and send message `Listen`(c) to the adversary.

Upon receiving input BROADCAST(c, m) **from party** p**:** send message `Broadcast`(c, m) to the adversary.

Upon receiving message DELIVER(c, m) **from the adversary:** if there is a record `Listen`(c, p) then remove this record and write output m to and reactivate party p.

Figure 3: Ideal anonymous communication.

To model this requirement in the UC framework, we introduce an ideal functionality $\mathcal{F}_{\mathsf{com}}$. This anonymous communication functionality is described in Figure 3. As the communication anonymity requirement applies to both the real and idealized protocols, our description of the real protocol in Section 4 also makes use of $\mathcal{F}_{\mathsf{com}}$.

4 Forward-secure optimistic RFID authentication

In this section we define two novel optimistic RFID authentication protocols: O-FRAP and O-FRAKE. Both protocols offer forward-anonymity, while requiring only minimal overhead when the system is not under attack. Our protocols rely on a trusted setup and on the wireless communication functionality described earlier.

These protocols are lightweight enough for RFID deployments, yet provide strong UC security and therefore are suitable in other ubiquitous application contexts, such as sensor networks. The only restriction is that the each component playing the role of a single tag must use separate keys when performing authentication/key-exchange subsessions in parallel.

4.1 Trusted Setup

The following trusted setup is done in a physically secure environment. For each tag i, a fresh key pair (r_i, k_i^a, k_i^b) is randomly generated and shared between the tag and the server. The server stores all the keys in a database and the tag stores its key in its non-volatile memory. Additionally, the server also stores the previous value of each key (r_i, k_i^a, k_i^b) in the database before any updates, which happen only when the tag is authenticated. The subkey k_i^b is generated and used in key exchange protocol only.

4.2 RFID entity authentication

Our first protocol, O-FRAP, is an Optimistic two pass Forward-secure RFID Authentication Protocol. In this protocol, r_{sys} and r_{tag} are values generated pseudo-randomly by the server and the tag, respectively, so as to anonymize the session and to prevent replays. The value r_{tag} is generated pseudo-randomly for optimistic identification of the tag. Value k_{tag}^a is the tag's current key and is updated by the server after the tag is authenticated, and by the tag after the server is authenticated.

On activation by the server, the tag computes four values $\nu_1, \nu_2, \nu_3, \nu_4$ by applying the pseudo-random function F to $(k_{tag}^a, r_{tag}||r'_{sys})$. Note that in all our protocols we use the following convention: any sent value x is received as x' by the receiver. The value x' may be different from x if it is corrupted by the adversary.

In O-FRAP, ν_1 is used to update the pseudo-random value r_{tag}; ν_2 is used for authentication of the tag; ν_3 is used to authenticate the server; ν_4 is used to update k_{tag}^a. In our protocols we use the following convention: the four values computed by the server by applying the pseudo-random function F to $(k_j^a, r'_{tag}\|r_{sys})$ are denoted by $\nu_1^*, \nu_2^*, \nu_3^*, \nu_4^*$. When the adversary is passive, these values correspond to the unstared values. In particular $\nu_2^* = \nu_2'$ and $\nu_3^{*\prime} = \nu_3$, and the server and tag output `ACCEPT`.

Observe that the tag key k_{tag}^a is updated after each server authentication, giving strong separation properties between sessions. In particular, if a tag is compromised, it cannot be linked to transcripts of earlier sessions. This guarantees strong separation of sessions and hence forward-anonymity.

4.3 RFID authenticated key exchange

We next describe O-FRAKE, an Optimistic Forward-secure RFID Authenticated Key Exchange protocol –see Figure 4. The protocol is essentially the same as O-FRAP except that five random values $\nu_1, \nu_2, \nu_3, \nu_4, \nu_5$ are generated the pseudo-random function F. The output value k_{tag}^b is an agreed subsession key for securing the communication channel between the server and the tag, for example to protect transmission of private information collected by the tag. Corruption or replacement of k_{tag}^b (either during the authentication protocol or during later use) is an attack on the exchanged key and has no effect on the authentication key k_{tag}^a. Furthermore, even if the adversary corrupts the tag, prior session keys are protected and prior session transcripts are unlinkable. This enforces separation of sessions and provides forward-anonymity, authenticity and secrecy.

5 Proof of security

Theorem 1 *O-FRAP and O-FRAKE UC-securely implements the anonymous RFID authentication and anonymous RFID authenticated key exchange ideal functionalities, respectively.*

Proof. We shall prove the theorem for O-FRAKE. O-FRAP then follows similarly. Observe

Figure 4: O-FRAP and O-FRAKE: optimistic forward-secure RFID entity Authentication and Authenticated Key Exchange protocols, respectively. O-FRAKE differs from O-FRAP only in the generation of an additional value to be used as session key (here shown inside a $\boxed{\text{box}}$).

Server		Tag
$\textsc{Server}(D)$		$\textsc{Tag}(r_{tag}, k^a_{tag}, \boxed{k^b_{tag}})$
$c_{sys} \leftarrow \mathcal{F}_{\mathsf{com}}.\,\textsc{Channel}$		$c_{tag} \leftarrow \mathcal{F}_{\mathsf{com}}.\,\textsc{Channel}$
$\mathcal{F}_{\mathsf{com}}.\,\textsc{Broadcast}(c_{sys}, r_{sys})$		
	$\xrightarrow{\quad r_{sys} \quad}$	
		$r'_{sys} \leftarrow \mathcal{F}_{\mathsf{com}}.\,\textsc{Listen}(c_{tag})$
		$\nu \leftarrow F(k^a_{tag}, r_{tag} \Vert r'_{sys})$
		$(\nu_1, \nu_2, \nu_3, \nu_4, \boxed{\nu_5}) \overset{\mathsf{parse}}{\longleftarrow} \nu$
		$(\overline{r}_{tag}, r_{tag}) \leftarrow (r_{tag}, \nu_1)$
		$\mathcal{F}_{\mathsf{com}}.\,\textsc{Broadcast}(c_{tag}, \overline{r}_{tag} \Vert \nu_2)$
	$\xleftarrow{\quad \overline{r}_{tag} \Vert \nu_2 \quad}$	
$(\overline{r}'_{tag} \Vert \nu'_2) \leftarrow \mathcal{F}_{\mathsf{com}}.\,\textsc{Listen}(c_{sys})$		
if exists $(\overline{r}'_{tag}, k^a_i, \boxed{k^b_i}) \in D$ then		
$SearchRange \leftarrow [i, i]$		
else		
$SearchRange \leftarrow [1, n]$		
fi		
for all j in $SearchRange$ do		
$\nu^* \leftarrow F(k^a_j, \overline{r}'_{tag} \Vert r_{sys})$		
$(\nu^*_1, \nu^*_2, \nu^*_3, \nu^*_4, \boxed{\nu^*_5}) \overset{\mathsf{parse}}{\longleftarrow} \nu^*$		
if $\nu'_2 = \nu^*_2$ then		
output $\mathtt{ACCEPT}(\mathsf{tag}(j), \boxed{k^b_j})$		
$(r_j, k^a_j, \boxed{k^b_j}) \leftarrow (\nu^*_1, \nu^*_4, \boxed{\nu^*_5})$		
$\mathcal{F}_{\mathsf{com}}.\,\textsc{Broadcast}(c_{sys}, \nu^*_3)$		
	$\xrightarrow{\quad \nu^*_3 \quad}$	
fi		$\nu^{*\prime}_3 \leftarrow \mathcal{F}_{\mathsf{com}}.\,\textsc{Listen}(c_{tag})$
od		if $\nu_3 = \nu^{*\prime}_3$ then
		output $\mathtt{ACCEPT}(\mathsf{server}, \boxed{k^b_{tag}})$
		$(k^a_{tag}, \boxed{k^b_{tag}}) \leftarrow (\nu_4, \boxed{\nu_5})$
		fi

that if F in the protocol is a true random function then the keys used in all fully completed tag subsessions are uniformly random and mutually independent. This means that conversations in fully completed tag subsessions are independently and identically distributed. This property

Ideal adversary $\mathcal{S}$

$\mathcal{S}$ simulates interactions between $\{\widehat{\mathcal{A}}, \widehat{\mathsf{server}}, \widehat{tag}_s, \widehat{\mathcal{F}}_{\mathsf{com}}\}$ and between $\widehat{\mathcal{A}}$ and $\mathcal{Z}$ as specified in Figure 4. In addition, interactions between $\{\widehat{\mathsf{server}}, \widehat{tag}_s, \widehat{\mathcal{F}}_{\mathsf{com}}\}$ and $\mathcal{Z}$ are emulated as follows:

Upon receiving $\mathtt{init}(s, \mathtt{server}, list)$ from $\mathcal{F}_{\mathsf{aake}}$:
Create a new subsession s for $\widehat{\mathsf{server}}$ and send $\mathtt{init}(s, \mathtt{server}, list)$ to $\widehat{\mathcal{A}}$.

Upon receiving $\mathtt{init}(s, \mathtt{tag}, list)$ from $\mathcal{F}_{\mathsf{aake}}$:
If $list$ is empty then generate a random key (r_s, k_s^a, k_s^b), else copy the key from a subsession identified in $list$. Add the specified key (r_s, k_s^a, k_s^b) to database $\widehat{D}$ using simulated identity $\widehat{\mathsf{tag}}_s$. Create a new subsession s for $\widehat{\mathsf{tag}}_s$ and send $\mathtt{init}(s, \mathtt{server}, list)$ to $\widehat{\mathcal{A}}$.

Upon $\widehat{\mathsf{server}}$ outputting $\mathtt{ACCEPT}(\widehat{p}, k)$ during subsession s ($\widehat{p} \in \widehat{D}$):
If $\widehat{p}$ is corrupted then send $\textsc{Impersonate}(s, \widehat{p}, k)$ to ideal functionality $\mathcal{F}_{\mathsf{aake}}$. Else let $\widehat{p} = \widehat{\mathsf{tag}}_{s'}$, generate a record $\mathtt{partner}(s, s')$ and send $\textsc{Accept}(s, s')$ to ideal functionality $\mathcal{F}_{\mathsf{aake}}$.

Upon $\widehat{\mathsf{tag}}_{s'}$ outputting $\mathtt{ACCEPT}(\widehat{\mathsf{server}}, k)$:
Remove $\widehat{\mathsf{tag}}_{s'}$'s key from database $\widehat{D}$, lookup record $\mathtt{partner}(s, s')$ and send $\textsc{Accept}(s', s)$ to ideal functionality $\mathcal{F}_{\mathsf{aake}}$.

Upon $\widehat{\mathcal{A}}$ sending $\textsc{Corrupt}$ to $\widehat{\mathsf{tag}}_{s'}$:
Mark $\widehat{\mathsf{tag}}_{s'}$ as corrupted and store its key in $\widehat{D}$ permanently. In particular, instead of being regenerated, the key is updated in future executions as normally specified by the protocol. Send message $\textsc{Corrupt}(s')$ to ideal functionality $\mathcal{F}_{\mathsf{aake}}$.

Figure 5: The ideal adversary $\mathcal{S}$ for $\mathcal{F}_{\mathsf{aake}}$.

also holds for all subsessions separated by at least a fully completed subsession, where the key is refreshed. Our simulation is as follows:

- Simulate a copy $\widehat{\mathcal{A}}$ of the real adversary $\mathcal{A}$, a copy $\widehat{\mathsf{server}}_s$ of the real server, a copy $\widehat{\mathsf{tag}}_s$ of a real tag for each tag subsession s and a copy $\widehat{\mathcal{F}}_{\mathsf{com}}$ of ideal functionality $\mathcal{F}_{\mathsf{com}}$. Forward messages among simulated parties $\{\widehat{\mathsf{server}}, \widehat{\mathsf{tag}}_s, \widehat{\mathcal{A}}, \widehat{\mathcal{F}}_{\mathsf{com}}\}$ and also between $\widehat{\mathcal{A}}$ and $\mathcal{Z}$ faithfully.

- The database $\widehat{D}$ of $\widehat{\mathsf{server}}$ contains persistent keys of corrupted tags and transient keys of active tags. Keys are added to and removed from $\widehat{D}$ on demand.

- The secret key of $\widehat{\mathsf{tag}}_s$ is copied from the immediately preceding incomplete subsession, if there is one, or is randomly generated, if the immediately preceding incomplete subsession of the tag is fully completed. This key is temporarily added to $\widehat{D}$ during simulation of the subsession s, and is removed from $\widehat{D}$ after successful completion of the subsession s.

- If $\widehat{\mathsf{tag}}_s$ is corrupted during the execution of subsession s then its key will be marked as corrupted and will never be removed from $\widehat{D}$. This persistence allows corrupted tags to be impersonated by the adversary $\widehat{\mathcal{A}}$. In this case, the corrupted key is updated accordingly to the protocol after each successful impersonation of $\widehat{\mathsf{tag}}_s$ by $\widehat{\mathcal{A}}$.

- To emulate the externally visible part of the protocol: upon corruption, acceptance or impersonation of simulated tags, notify $\mathcal{F}_{\mathsf{aake}}$ respectively with one of three messages $\textsc{Corrupt}(s)$, $\textsc{Accept}(s, s')$ and $\textsc{Impersonate}(s, p')$.

We describe the simulations in Figure 5. It is straightforward to verify that if the following two conditions hold then keys used in real executions and ideal simulations are statistically identical:

1. F is a truly random function.

2. Each verification done by the server succeeds with at most one key in the database.

Consequently the real messages and the simulated messages are also statistically identical, i.e. real and ideal are equal. The first condition fails if F is distinguishable from true random function. The second condition fails while the first holds if there are two keys verifying the random challenge r_{sys} and reply $(\overline{r}_{tag}, \nu_2)$. For each given tag subsession, this happens with probability at most $n2^{1-\kappa}$, where κ is the security parameter, i.e. the minimum bit length of $r_{sys}, \overline{r}_{tag}$ and ν_2, and n is total number of tags managed by this server. Therefore the probability that the second fails while the first holds is at most $nL2^{1-\kappa}$, where L is the total number of tag subsessions. Since both conditions fail with negligible probabilities, the real and simulated messages are computationally indistinguishable by the environment $\mathcal{Z}$. $\square$

The server and tags in our protocols are always synchronized because as the initiator of the protocol, the server is always at most one step ahead of the tag in updating the key. Therefore storing the previous value of the key on the server before any updates will allow the server to accommodate tags that are behind key update due to interferences of the adversary. It appears that our protocol described in Figure 4 does not makes use of the session identification *sid* provided in the universal composability framework. However, this is not quite true. The *sid* is used implicitly in the trusted setup (which is implemented externally to our protocol) to guarantee that the server and tags in the same session share the same secret keys. Without this trusted setup assumption, the security and functionality of our protocols is not guaranteed.

Security reduction and concrete complexity. A security reduction from distinguishing real-vs-ideal to distinguishing pseudo-vs-true random is simple: run a faithful simulation of the real world and use $\mathcal{Z}$ as the distinguisher. When a truly random function F is used in the real simulation, we obtain exactly the ideal simulation, modulo a negligible event that the second condition fails given that F is truly random. Therefore the advantage of distinguishing real from ideal is at most:

$$Adv_F(T + nL, nL) + nL2^{1-\kappa},$$

where $Adv_F(t, q)$ is the advantage of distinguishing F from true random by running in at most t time and making at most q queries to F, L is the number of tag subsessions, n is the number of tags and T is the combined time complexity of the environment Z and the adversary $\mathcal{A}$.

6 Lightweight constructions for O-FRAP and O-FRAKE

In this section we show how to achieve a very efficient, practical construction of O-FRAP and O-FRAKE by using only a pseudo-random generator (PRG). Estimation of the hardware requirements of a prototypical specification are of the order of 2000 gates.

6.1 Lite pseudo-random function families

We describe how to achieve a very efficient, practical construction of large-length output 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 [21] 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 1 *If PRG is a pseudo-random generator and PRF is a pseudo-random function then* $F = PRG \circ PRF$ *is a pseudo-random function.*

Proof. Let X, Y, W, and Z be efficiently sampleable domains and let $PRF : X \times Y \to W$ be a pseudo-random function and $PRG : W \to Z$ be a pseudo-random generator. We show that $F = PRG \circ PRF : X \times Y \to Z$ 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. Notice that $F(x, y_i) = PRG(w_i)$ where $w_i = PRF(x, y_i)$. 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 $\vec{z} = (PRG(w_1), \ldots, PRG(w_n))$ is indistinguishable from $\vec{z}^* = (PRG(w_1^*), \ldots, PRG(w_n^*))$, where $w_1^*, \ldots, w_n^*$ are randomly and independently selected from W. By pseudo-randomness of the distribution of $PRG(w_i^*)$ and the multi-sample indistinguishability theorem of Goldreich [20] and Yao [36], $\vec{z}^*$ is indistinguishable from a random vector in Z^n. □

For practical RFID implementations a very efficient hardware implementation of a PRG should be used. In general a PRG can be implemented much more efficiently than a standard cryptographic pseudo-random function can. For instance, the shrinking generator [1] of Coppersmith, Krawczyk, and Mansour [17] can be implemented with fewer than 2000 gates with approximately 80-bit security [4], 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 [4]. Alternatively, Grain [22] or any other secure hardware stream cipher can be used. See [31] for examples of such ciphers.

Standard cryptographic constructions, such as HMAC (requiring a cryptographic hash function with pseudo-random property) or CBC-MAC with a block cipher (for instance, AES) would require around 10-15K gates. These constructions are suitable only for a narrow range of higher cost RFID tags. However, using our constructions, one obtains a full-fledged implementation of the O-FRAP and O-FRAKE protocols using approximately 2000–3000 gates, which covers a much wider range of RFID architectures.

7 Featherweight RFID authentication

In this section we consider a family of RFID authentication and key exchange protocols secure against fly-by attacks, named A-TRAP after Optimistic "Absolutely" Trivial RFID Authentication Protocols, to emphasize their minimalistic structure and overhead. These protocols only require a pseudo-random generator and a *Time-Delay Scheduler* (TDS).

The TDS is a very simple hardware device that controls the time-delay between authentication sessions. The time-delay is minimal, say t_0, between complete authentication sessions—i.e., sessions that terminate with the tag's key update. After each incomplete session, the time delay is doubled. So, after m successive incomplete sessions there will be a time-delay of $2^m t_0$. The TDS is used to thwart attacks in which the adversary triggers incomplete sessions to desynchronize the key

[1]Using the shrinking generator requires care (buffering) to avoid the introduction of vulnerabilities to timing and side-channel attacks.

updates of the tag and the server. A limited number of time-delay doublings can be easily achieved using capacitors, acquiring enough energy before running the protocol, and/or counters. During this delay, the whole tag is power downed except for a counter and the clock rate is reduced to minimal, only enough to run the counter. These have the potentials to extend the delay by few orders of magnitude.

A-TRAP protocols offer limited protection against desynchronization attacks: a tag that can be "interrogated" longer than an upper bound of m successive times will become permanently invalidated. However, for attacks that interact with a tag for a time period shorter than $2^m t_0$ time units (a fly-by attack), these protocols offer provably secure authentication, forward-anonymity, availability, and key-indistinguishability.

$d_{1,1}$	$\dots$	$d_{1,m-j}$	$d_{1,m-j+1}$	$\dots$	$d_{1,m}$
$\vdots$		$\vdots$	$\vdots$		$\vdots$
$d_{i,j+1}$	$\dots$	$d_{i,m}$	$g_i^{(1)}$	$\dots$	$g_i^{(j)}$
$\vdots$		$\vdots$	$\vdots$		$\vdots$
$d_{n,1}$	$\dots$	$d_{n,m-j}$	$d_{n,m-j+1}$	$\dots$	$d_{n,m}$

Figure 6: $\mathsf{Update}(D, i, j)$: discard the first j entries of the i^{th}-row, shift the remaining entries to the left, and finally fill in the empty cells with the next j values extracted from g_i.

7.1 A-TRAP: featherweight mutual authentication

A-TRAP is a mutual authentication protocol in which, the tag and the server exchange values ν_1, ν_2, ν_3, respectively, generated by the pseudo-random generator g_{tag} – see Figure 7. The server checks that the received value g'_{tag} is in its database $D = \{d_{i,j}\}$: if $d_{i,j} = {g_{tag}}'$ then it accepts the tag as authentic. In this case it updates the i-th row of it directory D by: (a) discarding its first j entries, (b) shifting the remaining entries to the front, and finally (c), filling the empty cells with the next j values $g_i^{(1)}, \dots, g_i^{(j)}$ extracted from the pseudo-random generator g_i (see Figure 6). If the value g'_{tag} is not in D then the tag is rejected. A variant of A-TRAP achieves key exchange by generating a third value ν_3 using the pseudo-random generator g_{tag}. The security of O-FRAP, O-FRAKE, and the A-TRAP protocol families is discussed.

8 Conclusion

We present highly practical RFID authentication protocols that are provably secure, providing forward-anonymity, authenticity, availability, and session key indistinguishability within a universal composability framework. The suggested implementation of the protocols requires only the use of pseudo-random generators and is feasible for a wide range of RFID architectures.

References

[1] Ateniese, G., Camenisch, J., and de Medeiros, B. Untraceable RFID tags via insubvertible encryption. In *Proc. ACM Conf. on Computer and Communication Security (ACM CCS 2005)* (2005), ACM Press, pp. 92–101.

Figure 7: A-TRAP: The tag and the server are mutually authenticated using a pseudo-random generator g_{tag}. In the authenticated key-exchange (AKE) variant, the tag and the server also compute $\nu_3^* = \nu_3$ as their shared key. The third value is within in a $\boxed{\text{box}}$ to indicate that it only appears in the AKE variant.

SERVER(D)		TAG(g_{tag})
		$\nu_{tag} \leftarrow g_{tag}$
		$\nu_1 \| \nu_2 \| \nu_3 \| \boxed{\nu_4} \xleftarrow{parse} \nu_{tag}$
	$\xleftarrow{\nu_1}$	
if exists $d_{i,j} = \nu_1' \| \nu_2^* \| \nu_3^* \| \boxed{\nu_4^*}$ in D		
	$\xrightarrow{\nu_2^*}$	
		if $\nu_2^{*\prime} = \nu_2$
	$\xleftarrow{\nu_3}$	
if $\nu_3' = \nu_3^*$		output(ACCEPT, $\boxed{\nu_4}$)
update(D, i, j)		else
output(ACCEPT, $\boxed{\nu_4^*}$)		output(REJECT, $\perp$)
endif		endif
endif		
output(REJECT, $\perp$)		

[2] AVOINE, G. Security and privacy in RFID systems. `http://lasecwww.epfl.ch/~gavoine/rfid/`.

[3] AVOINE, G., AND OECHSLIN, P. A scalable and provably secure hash-based RFID protocol. In *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2005)* (2005), IEEE Press, pp. 110–114.

[4] BATINA, L., LANO, J., MENTENS, N., ÖRS, S. B., PRENEEL, B., AND VERBAUWHEDE, I. Energy, performance, area versus security trade-offs for stream ciphers. In *The State of the Art of Stream Ciphers, Workshop Record* (2004), ECRYPT.

[5] BEAVER, D. Foundations of secure interactive computing. In *Proc. Advances in Cryptology (CRYPTO 1991)* (1991), vol. 576 of *LNCS*, Springer, pp. 377–391.

[6] BEAVER, D. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology 4:2* (1991), 75–122.

[7] BEAVER, D., AND GOLDWASSER, S. Multiparty computation with faulty majority. In *Proc. Advances in Cryptology (CRYPTO 1989)* (1989), vol. 435 of *LNCS*, Springer, pp. 589–590.

[8] BONO, S. C., GREEN, M., STUBBLEFIELD, A., RUBIN, A. J. A. D., AND SZYDLO, M. Security analysis of a cryptographically-enabled RFID device. In *Proc. USENIX Security Symposium (USENIX Security 2005)* (2005), USENIX, pp. 1–16.

[9] Burmester, M., Le, T. V., and de Medeiros, B. In *Proc. Second International Conference on Security and Privacy in Communication Networks (SECURECOMM 2006)*.

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

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

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

[13] Canetti, R., and Fischlin, M. Universally composable commitments (extended abstract). In *Proc. Advances in Cryptology (CRYPTO 2001)* (2001), vol. 2139 of *LNCS*, Springer, p. 19.

[14] Canetti, R., and Herzog., J. Universally composable symbolic analysis of cryptographic protocols (the case of encryption-based mutual authentication and key exchange). Tech. Rep. E-print Report # 2004/334, International Association for Cryptological Research, 2004.

[15] Canetti, R., and Krawczyk, H. Universally composable notions of key exchange and secure channels (extended abstract). In *Proc. Advances in Crypotology (EUROCRYPT 2002)* (2001), vol. 2332 of *LNCS*, Springer, p. 337.

[16] Canetti, R., Lindell, Y., Ostrovsky, R., and Sahai, A. Universally composable two-party and multi-party secure computation. In *Proc. ACM Symp. on Theory of Computing (STOC 2002)* (2002), vol. 34, ACM Press, pp. 494–503.

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

[18] Dimitriou, T. 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)* (2005), IEEE Press.

[19] Gilbert, H., Rodshaw, M., and Sibert, H. An active attack against HB+ – a provably secure lightweight authentication protocol. Tech. rep., International Association for Cryptological Research, 2005.

[20] Goldreich, O. *The foundations of cryptography.* Cambridge University Press, 2001.

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

[22] Hell, M., Johansson, T., and Meier, W. Grain - a stream cipher for constrained environments.

[23] Henrici, D., and Müller, P. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *Proc. IEEE Intern. Conf. on Pervasive Computing and Communications (PerCom 2004)* (2004), IEEE Computer Society Press, pp. 149–153.

[24] Hofheinz, D., Müller-Quade, J., and Steinwandt, R. Initiator-resilient universally composable key exchange. In *Proc. European Symp. on Research in Computer Security (ESORICS 2003)* (2003), vol. 2808 of *LNCS*, Springer, pp. 61–84.

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

[26] Juels, A., and Weis, S. A. Authenticating pervasive devices with human protocols. In *Proc. Advances in Cryptology (CRYPTO 2005)* (2005), vol. 3621 of *LNCS*, Springer, p. 293.

[27] Juels, A., and Weis, S. A. Defining strong privacy for RFID. E-print report 2006/137, International Association for Cryptological Research, 2006.

[28] Katz, J., and S.Shin, J. Parallel and concurrent security of the HB and HB+ protocols. In *Proc. Advances in Crypotology (EUROCRYPT 2006)* (2006), LNCS, Springer.

[29] Laud, P. Formal analysis of crypto protocols: Secrecy types for a simulatable cryptographic library. In *Proc. ACM Conf. on Computer and Communication Security (ACM CCS 2005)* (2005), ACM Press, pp. 26–35.

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

[31] Network of Excellence within the Information Societies Technology (IST) Programme of the European Commission. Estream: The stream cipher project. `http://www.ecrypt.eu.org/stream`.

[32] Ohkubo, M., Suzuki, K., and Kinoshita, S. Cryptographic approach to "privacy-friendly" tags. RFID Privacy Workshop, November 2003.

[33] Pfitzmann, B., and Waidner, M. Composition and integrity preservation of secure reactive systems. In *Proc. ACM Conf. on Computer and Communication Security (ACM CCS 2000)* (2000), ACM Press, pp. 245–254.

[34] Pfitzmann, B., and Waidner, M. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. IEEE Symp. on Security and Privacy (S & P 2001)* (2001), IEEE Press, pp. 184–200.

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

[36] Yao, A. C. Theory and application of trapdoor functions. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS 1982)* (1982), pp. 80–91.