Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer

Chae Hoon Lim and Taekyoung Kwon*

Dept. of Computer Engineering, Sejong University, Seoul 143-747, Korea
{tkwon, chlim}@sejong.ac.kr

Abstract. RFID technology arouses great interests from both its advocates and opponents because of the promising but privacy-threatening nature of low-cost RFID tags. A main privacy concern in RFID systems results from clandestine scanning through which an adversary could conduct silent tracking and inventorying of persons carrying tagged objects. Thus, the most important security requirement in designing RFID protocols is to ensure untraceability of RFID tags by unauthorized parties (even with knowledge of a tag secret due to no physical security of low-cost RFID tags). Previous work in this direction mainly focuses on backward untraceability, requiring that compromise of a tag secret should not help identify the tag from past communication transcripts. However, in this paper, we argue that forward untraceability, i.e., untraceability of future events even with knowledge of a current tag secret, should be considered as an equally or even more important security property in RFID protocol designs. Furthermore, RFID tags may often change hands during their lifetime and thus the problem of tag ownership transfer should be dealt with as another key issue in RFID privacy problems; once ownership of a tag is transferred to another party, the old owner should not be able to read the tag any more. It is rather obvious that complete transfer of tag ownership is possible only if some degree of forward untraceability is provided. We propose a strong and robust RFID authentication protocol satisfying both forward and backward untraceability and enabling complete transfer of tag ownership.

1 Introduction

Radio Frequency Identification (RFID) is an automated identification technology in which a small transponder, attached to a real world object, receives and responds to radio-frequency queries from a transceiver. The transponder is usually called an RFID *tag* while the transceiver is an RFID *reader*. The RFID tag incorporates silicon chips with radio antennas for electronic operations and wireless data transmissions. It tends to have extremely limited capabilities in every aspect of computation, communication, and storage for economic viability. Passive tags are not equipped with an internal power source, contrary to

* Research by the 2nd author was supported by grant No. R01-2005-000-11261-0 from Korea Science and Engineering Foundation in Ministry of Science & Technology.

P. Ning, S. Qing, and N. Li (Eds.): ICICS 2006, LNCS 4307, pp. 1–20, 2006.

semi-passive or active tags with built-in batteries. They store authentic data and respond for identification and authentication, with neither physical nor visual contact. The RFID reader communicates with tags and cooperates with a backend database which contains information on the tagged objects.

In fact, this technology is not fundamentally new; rather it has been around since the late 1960s and is being used in the public domain [11]. Recently, RFID has aroused a great interest from various communities due to the promising nature of small low-cost RFID tags in future smart applications. Rapid RFID progress has already been made in retail sectors, such as Wal-Mart and Procter & Gamble, as well as in government sectors, such as U.S. DoD and Postal Service [14]. The U.S. government also has mandated adoption by Oct 26, 2006 of *e-passports* (biometrically-enabled RFID tags) by the 27 countries in the Visa-Waiver Program [16]. It is widely believed that RFID tags will more rapidly spread over and its cost will go down fast in the near future.

RFID systems however raise a lot of privacy concerns, mainly due to the possibility of clandestine tracking and inventorying of tags [27,30,24,5,15,16,14]. For example, adversarial parties equipped with commodity RFID readers may trace a person carrying a tagged item by recognizing the same tag in different places at different times. This traceability problem is considered as the biggest security challenge to general acceptability and wide-scale deployment of RFID technology. Actually the boycott movement from those fearing privacy infringement made companies like Benetton and Gillette drop or reconsider their RFID-tagging plans [7,29]. Fortunately, a number of studies have also been done for handling such security and privacy issues in RFID systems [17,24,10,13,19,4,8,23]. The approaches taken in these studies vary, from schemes based on weak but realistic models to strong cryptographic techniques, and each approach may have its own merit and demerit.

In this paper, we are more interested in a stronger security model, assuming that tag secrets may be read by an adversary, since most low-cost RFID tags have no protection capability of the tag memory. Since reading the tag memory content endows the adversary with full capability of the tag from the moment, it is very important to see how the past and the future transactions of the tag are related with the current internal state of the tag at the time of memory break-in. This observation brings us the security notions of backward (resp. forward) untraceability, meaning that knowledge of a tag's current internal state must not help identify the tag's past (resp. future) interactions.[1] Most previous studies focus on backward untraceability and, as far as we know, no attention has been paid explicitly to forward untraceability yet. In this paper, we would like to call our attention to the importance of forward untraceability and related issues.

We argue that *forward untraceability* is even more important than backward untraceability in RFID systems. Suppose that compromise of tag secrets results in complete loss of control over the tags. Then, it may be catastrophic if tag secrets are compromised in some point of tag deployment or during their cir-

[1] Note that we used the terms 'forward' and 'backward' opposite to usual definitions. See Section 2 for our justification.

culation within supply chains; then it would be much easier to trace the tags and reproduce cloned tags. Another important related issue is the problem of *ownership transfer*. Since tags may change hands frequently during their lifetime, it is certainly necessary to provide some means of ownership transfer of a tag from one party to another. Ownership of a tag means the ability to read the tag and thus ownership transfer should guarantee that once ownership of a tag is transferred, the tag should be able to be read only by the new owner but never by the old owner. Such a complete transfer of tag ownership would be impossible unless some degree of forward untraceability is provided, since the old owner would have already owned all the information necessary to control the tag. Note that we are talking about perfect ownership transfer between users, contrary to Molnar et al.'s temporary ownership transfer or time-limited access delegation [23] (See Section 4 for more details).

Our Contribution. As discussed above, there is of no doubt on the importance of forward untraceability, in addition to traditional backward untraceability, in designing RFID authentication protocols. Backward untraceability is easy to achieve by updating tag secrets based on a one-way key chain and has been widely studied in the literature. However, it is never easy to achieve forward untraceability using cryptographic techniques in low-cost RFID tags, due to the very limited resources available in such tags. The mobility of tagged items is our primary finding as a means of achieving forward untraceability with little increase of complexity. That is, even if an adversary learns the tag secret of a particular person's belonging, he will not be able to physically track the target item all the way from the moment of tag break-in. Thus, assuming that it is not possible for the adversary to eavesdrop all the interactions of the target tag afterwards, we will be able to completely refresh the tag secret in synchronization with the backend database by injecting into the tag secret the shared randomness involved in every successful authentication. In this paper, we first bring the notion of forward untraceability explicitly and rigorously in the design of RFID authentication protocols and propose such a protocol achieving both requirements of forward and backward untraceability. Furthermore, we show that our protocol enables perfect transfer of tag ownership between users. This feature will be essential in trading tagged objects in the real world. We also show that this feature can be used to delegate access to tags to potentially untrustworthy readers for distributed processing of a central database and may help thwart tag cloning by refreshing the tag secret whenever necessary.

2 RFID Systems and Security

2.1 The Communication Model

An RFID system consists of three main entities such as RFID tags, RFID readers, and a backend database server, along with communication channels between them. Figure 1 depicts the high-level view of the communication and security model for conducting RFID authentication in general. The channels between the

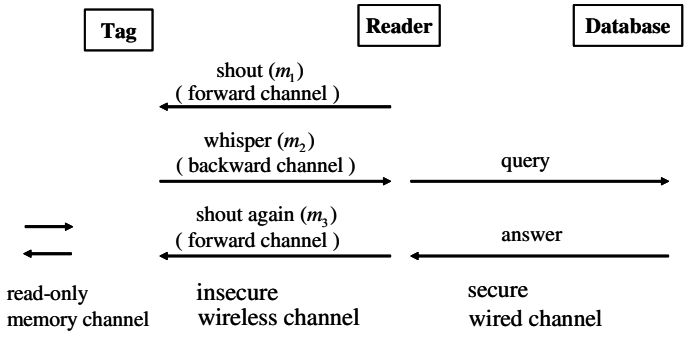

Fig. 1. The top-level communication and security model for RFID systems

tag and the reader are wireless radio channels which can be read or engaged by an adversary, while those between the reader and the database might be wired channels which are usually assumed secure. The channel from a reader to a tag is called the forward channel and its opposite is the backward channel. Due to the difference of signal strengths flowing in different directions, the adversary may have in general more chances to read the forward channel. Since we can't imagine physically secure tag chips in most low-cost RFID tags, it is reasonable to assume that the tag memory could be read by an adversary. In this respect, we may view read access to the tag memory as another hypothetical channel, called the memory channel [2], wiretapable by the adversary.

It is still a challenging task to design and analyze RFID authentication protocols since the capabilities of passive tags are extremely limited. The problem becomes rather paradoxical in the context of tag authentication without public key cryptography. Actually a legitimate reader cannot authenticate itself to a tag until it knows which key to use, requiring the tag's identity ahead, while the tag does not want to reveal its identity to an unauthenticated reader for privacy reasons. In Figure 1, the reader may shout without knowing the tag's identity in the first flow, and the tag should not reveal its identity to an illegitimate reader while whispering in the second flow. The third flow may be necessary for authenticating the reader and possibly updating the tag's internal state.

2.2 Forward Versus Backward Untraceability

In the above communication model, an adversary is able to collect a set of readings on both the forward and backward communication channels and also to tamper with a target tag's memory channel to learn its internal state at a certain moment. Thus we need to define untraceability of a tag in either direction of time travels from the moment of tag memory break-in. This brings us the notions of backward and forward untraceability. Unfortunately, the terms 'forward' and 'backward' have been used in security definitions somewhat ambiguously and are still controversial [28,1]. The cryptographic community has long time used

the term 'forward security' loosely to mean the protection capability of past traffic even with disclose of a current secret in many public key designs, in order to mitigate the damage from the long-term secret key exposure (e.g., key exchange protocols, public key encryption and digital signatures). However, the opposite concept, 'backward security', is rarely used explicitly (though some key-evolving signatures introduce an external trusted authority to achieve such a security property; e.g., see [22]). This might be one reason for the lack of efforts in establishing agreed-upon concrete definitions for both security notions. Note however that these definitions are exactly opposite to the verbatim meaning of the words and also to our intuition. They are awkward in particular when used in conjunction with untraceability. We will thus use the terms 'forward' and 'backward' untraceability literally as defined below (more general and formal definitions on various untraceability notions are provided in Appendix A).

Backward and forward untraceability are concerning the indistinguishability of past and future interactions of a tag with knowledge of the current internal state of the tag at the time of memory break-in. Backward untraceability states that even if given all the internal states of a target tag at time t, the adversary should not be able to identify the target tag's interactions that occurred at time $t' < t$. That is, it requires that knowledge of a tag's current internal state should not help identify the tag's past interactions. Backward untraceability has been considered as the most important security requirement in strong RFID authentication, since otherwise the past transcripts of a tag (e.g., from readers' logs) may allow tracking of the tag owner's past behaviors.

On the other hand, to our best knowledge, the opposite concept, forward untraceability, has not yet brought explicitly in the research community of RFID security. Forward untraceability can be similarly defined as requiring that knowledge of a tag's internal state at time t should not help identify the tag's interactions that occurred at time $t' > t$. In fact, not much attention has been paid to this security notion, since it is obvious that there exists no way (without the help of some external trusted authority) to maintain the future security once the current secret is exposed. The only thing we can do is to detect key compromise a.s.a.p. and to replace the exposed key with a fresh one to protect future transactions. However, this may not be easy in RFID systems; it is almost impossible to detect compromise of a tag secret and the tag secret may not be manageable by the tag owner.

Obviously, perfect forward untraceability makes no sense, since the adversary ia able to trace the target tag at least during the authentication immediately following a compromise of the tag secret. Thus the minimum restriction that may be imposed to achieve forward untraceability would be such that there should exist some non-empty gap between the time of memory break-in and the attack time in which the adversary could not hear the interactions (see Appendix for more details). Forward untraceability is thus harder to achieve than backward untraceability in general, in particular under the very constrained environment such as RFID tags. Nevertheless, we note that forward untraceability is never

less important than backward untraceability in RFID systems. There may exist some situations in which forward untraceability is even more important.

Both security requirements will be equally important in fighting against the universal surveillance threat by some powerful organizations (such as intelligence agencies) capable of collecting a huge amount of interaction logs (legally or illegally) almost without limitation in time and coverage. On the other hand, in the case of target tracing, we may not need such a power. It suffices to somehow steal the tag secret attached to a particular target's always-carry-on item and collect interaction logs from the target's frequently visiting places to trace the future behaviors of the particular target. Such a target tracing is trivial without forward untraceability. An even catastrophic scenario without forward untraceability would be such a case that tag secrets are leaked at some point of tag deployment or during the stay in supply chains. Then, all such tags could be traced afterwards. We thus raise a strong motivation to the need of forward untraceability in RFID protocol designs (even if not perfect), in addition to the well-recognized backward untraceability. This property is also closely related to the problem of ownership transfer of tags as we will see in Section 4.

2.3 Previous Work

There have been proposed a number of RFID security protocols in the literature [30,25,17,24,12,13,6,8,21,19,23]. They can be classified into two broad classes; a class of protocols trying to enhance privacy and security in RFID systems without using standard cryptographic primitives, e.g., [17,24,13,19], and a class of protocols relying on symmetric-key primitives such as block ciphers and hash functions, e.g., [30,25,24,12,6,8,21,23]. The former proposals aim at finding some security enhancements best achievable and easy to implement under the current hardware and functionality of RFID tags (e.g., EPC UHF Gen2 tags) but they still have a number of practical issues to be addressed for actual implementations and not so easy to implement in the current standard tags either. On the other hand, the latter protocols assume enhanced tags with built-in hardware circuits for a symmetric primitive and pursue stronger security under still resource-constrained environments. There also exist some work on optimized design and implementation of block ciphers for low-cost RFID tags, e.g., [10,20].

We do not survey previous work in detail, but refer the reader to, e.g., [5,2,15,14]. We briefly examine the Ohkubo-Suzuki-Kinoshita (OSK, for short) protocol [25,6] however, as it is most relevant to and the starting point of our proposed protocol. The basic idea of the scheme is to use a one-way key chain to evolve a tag secret in response to every query request. Then, only the backend database can identify the tag since it is the only other party with knowledge of the initial tag secret for the one-way key chain. More specifically, a tag T_i is initialized with a random secret s_i and, whenever queried, emits $r_i = h(0, s_i)$ and evolves the tag secret s_i as $s_i \leftarrow h(1, s_i)$, where h is a one-way function. If the backend server keeps a key chain of length m for each tag T_i, i.e., $\{r_i^k\}_{k=0}^{m-1}$, where $r_i^j = h(0, s_i^j)$, $s_i^j = h(1, s_i^{j-1})$ and $s_i^0 = s_i$, then a tag can be identified just by searching the database for each query response r_i. Once the tag T_i is

identified, the backend server updates the precomputed key chain so that it now contains m key chain values starting from the verified tag secret s_i. Thus the parameter m specifies the maximum number of authentication failures allowable between two valid sessions. Actually, this protocol should be modified into a challenge-response type to get resistance against replay attacks and there may exist more efficient time-memory tradeoffs to enhance the efficiency of the backend server (see [6,4]). It is easy to see that this protocol is backward untraceable due to the tag secret evolution through a one-way key chain.

3 The Proposed Authentication Protocol

3.1 Design Rationale

Our proposed protocol starts with the simple OSK protocol and augments it with mutual authentication and further protection capability in view of forward untraceability, thus making the resulting protocol immune against both the forward and the backward tracking attacks. First note that the OSK protocol achieves backward untraceability by updating a tag secret deterministically in response to every authentication request. The backend database then maintains a key chain of length m evolved from the tag secret of the last successful authentication, so that desynchronization up to m times can be resolved within this key chain.

The basic idea to enhance the protocol with forward untraceability is to refresh the tag secret simultaneously within both the tag and the central database, whenever the authentication is completed successfully, using the authentic random numbers exchanged during the protocol execution. Note that we use the term *update* to mean deterministic evolution of tag secrets while *refresh* to mean probabilistic evolution. That is, in every authentication session, the tag secret is evolved using a one-way key chain in two different ways; If the authentication succeeds, then both the tag and the database refresh the tag secret probabilistically using the exchanged random numbers, while, if the protocol fails anyway, the tag updates its secret deterministically as in the OSK protocol. Then, the resulting protocol would be made forward untraceable from the moment that an adversary is missing even one successful authentication session after compromising the tag secret.

One problem still remains in the above approach. If the adversary executes the protocol with a tag immediately after compromising the tag secret (she can do it successfully since she knows the tag secret), then the tag secret will be permanently desynchromized in the tag and the backend database, and the tag can be read only by the adversary.[2] This is because the tag refreshes its secret probabilistically using the randomness only shared with the adversary. To repair this problem, we introduce another one-way key chain maintained by the database and verified by the tag. The tag then refreshes its secret only if a received key chain value is verified. Note that this key chain is used in reverse

[2] This very property can be used to transfer tag ownership from the database to a consumer, as we can see later.

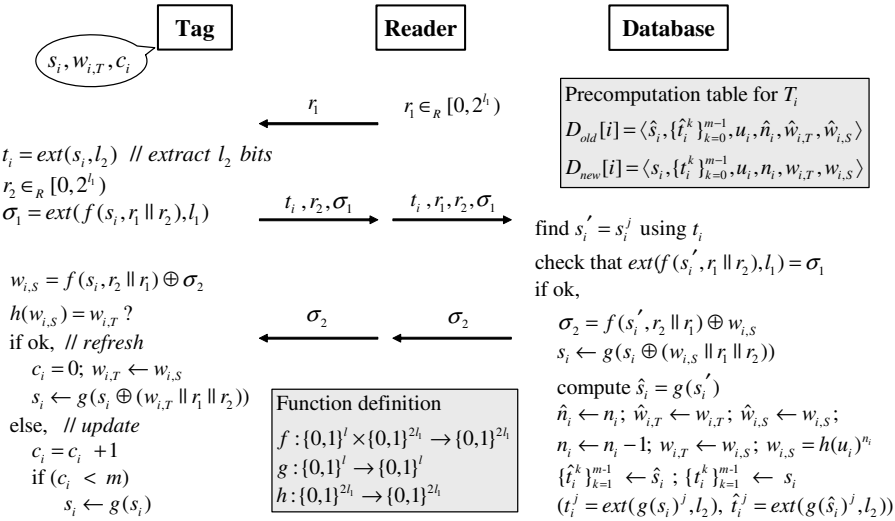

Fig. 2. The proposed protocol

order, contrary to the key chain for tag secret update, so we call it as a *backward key chain* while the latter as a *forward key chain*. We thus use the forward key chain for tag secret evolution and the backward key chain for server validation which triggers a refresh of the tag secret.

Finally, we note that there may still exist some subtle desynchronization problem in the case that the protocol message in the third flow is lost during transmission due to either unreliable medium or denial-of-service attacks by an adversary. The loss of the last protocol message in transit again results in permanent desynchronization of the tag secret, since then the tag will update the tag secret while the server will refresh it. We solve this problem by making the database keep two key chains of length m of relevant secrets, one based on the old secret and the other based the new secret, and examine both key chains in the next authentication. The problem can then be resolved, since the tag secret will belong to one of the two key chains, depending on whether or not the last protocol message arrives correctly.

3.2 Protocol Description

Parameters. The following parameters are used in the proposed protocol.

– m : The maximum number of allowable authentication failures between two valid sessions. If the protocol fails more than this threshold after the last successful interaction with the server (via an honest reader), then the tag stops evolving its secret and keeps using the last updated secret until the next successful authentication.

- n : The length of the backward key chain used for server authentication. This value can be set around the maximum number of successful authentications (not counting failed sessions) expected during the tag lifetime.
- ℓ : The bit-length of a tag secret.
- ℓ_1 : The bit-length of random challenges and responses.
- ℓ_2 : The bit-length of the tag secret transmitted in clear to help the backend server to identify the tag secret in its database. This value may depend on the tag population managed by the server (e.g., $\ell_2 \simeq \log_2 2mN$, where N is the maximum expected number of tags). The tag secret of length $\ell = \ell_2 + \ell'$ thus has the effective key length of ℓ' bits.

Pseudorandom Functions. Our protocol makes use of three pseudorandom functions f, g and h, all of which may be constructed from a single lightweight block cipher as can be seen later. We denote by $g(x)^n$ n-times applications of the function g on x :

- $f : \{0,1\}^\ell \times \{0,1\}^{2\ell_1} \rightarrow \{0,1\}^{2\ell_1}$: A pseudorandom function to generate authenticators.
- $g : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$: A pseudorandom function to build the forward key chain used to evolve tag secrets.
- $h : \{0,1\}^{2\ell_1} \rightarrow \{0,1\}^{2\ell_1}$: A pseudorandom function to build the backward key chain used to authenticate the server.

Tag & Database Initialization. Each tag T_i is initialized by the backend database server as follows:

- The server chooses a random secret s_i (say, of 128 bits) for the tag T_i, evaluates $(m-1)$ evolutions of s_i, $s_i^0 = s_i$ and $s_i^j = g(s_i^{j-1})$ for $1 \leq j \leq m-1$, and extracts the key identifiers t_i^j for s_i^j as $t_i^j = ext(s_i^j, \ell_2)$ for $0 \leq j \leq m-1$, where $ext(x, \ell)$ denotes a simple extract function returning ℓ bits out of x (e.g., $x \bmod 2^\ell$).
- The server also chooses a random $u_i \in [0, 2^{2\ell_1})$ for each tag T_i and computes a key chain of length n, $\{w_i^j\}_{j=0}^{n-1}$, such that $w_i^n = u_i$ and $w_i^j = h(w_i^{j+1})$ for $0 \leq j < n$. This key chain is used in reverse order to authenticate the server and to trigger a refresh of a tag secret.
- The tag then stores the pair of (tag secret, server validator) $\langle s_i, w_{i,T} \rangle$ and initializes the failure counter c_i as $c_i = 0$, where $w_{i,T} = w_i^0$.
- The server makes two entries for T_i, $D_{old}[i]$ (initially empty) and $D_{new}[i]$, in its database and stores the T_i's identification data $\langle s_i, \{t_i^j\}_{j=0}^{m-1}, u_i, n_i, w_{i,T}, w_{i,S} \rangle$ in the entry $D_{new}[i]$, where $w_{i,S} = w_i^1$ and $n_i = n$. Here, the variable n_i maintains the depth of the current $w_{i,S}$ in the key chain (i.e., $w_{i,S} = h(u_i)^{n_i}$). Note that $w_{i,T} = h(w_{i,S})$.

Authentication Procedures

1. The reader picks $r_1 \in_R [0, 2^{\ell_1})$ and sends it to the tag.
2. The tag chooses $r_2 \in_R [0, 2^{\ell_1})$, computes $t_i = ext(s_i, \ell_2)$ and $\sigma_1 = ext(f(s_i, r_1 \,\|\, r_2), \ell_1)$ and sends $\langle t_i, r_2, \sigma_1 \rangle$ to the reader.

3. The reader then queries $\langle t_i, r_1, r_2, \sigma_1 \rangle$ to the database server.
4. The server searches its database to find an entry containing the received key identifier t_i. If no match is found, the server responds with $\sigma_2 = \perp$ (denoting 'failure') and stops. Suppose that a match of $t_i = t_i^j$ for some j is found in one of T_i's entries, $D_{old}[i]$ or $D_{new}[i]$, containing $\langle s_i, \{t_i^k\}_{k=0}^{m-1}, u_i, n_i, w_{i,T}, w_{i,S} \rangle$. Then, the server computes the tag secret corresponding to t_i^j by $s_i' = g(s_i)^j$ and checks that $ext(f(s_i', r_1 \parallel r_2), \ell_1) = \sigma_1$. If the check fails, the server stops with output $\sigma_2 = \perp$. If the check succeeds, the server sends the response σ_2 to the reader, where $\sigma_2 = f(s_i', (r_2 \parallel r_1)) \oplus w_{i,S}$. The server then updates the two entries, $D_{old}[i]$ and $D_{new}[i]$, of the identified tag T_i as follows:
   1) The server moves the set of data found in the identified entry to $D_{old}[i]$ after updating the key identifiers t_i^j's according to the verified tag secret s_i'; $\hat{t}_i^k = t_i^{j+k+1}$ for $0 \le k \le m-j-1$, $\hat{s}_i = g(s_i')$ and $\hat{t}_i^k = ext(g(\hat{s}_i)^{k-m+j}, \ell_2)$ for $m-j \le k \le m-1$. Thus, we have $D_{old}[i] = \langle \hat{s}_i, \{\hat{t}_i^k\}_{k=0}^{m-1}, u_i, n_i, w_{i,T}, w_{i,S} \rangle$.
   2) The server then generates new data for $D_{new}[i]$ as follows: $s_i \leftarrow g(s_i \oplus (w_{i,S} \parallel r_1 \parallel r_2))$, $t_i^j = ext(g(s_i)^j, \ell_2)$ for $0 \le j \le m-1$, $n_i \leftarrow n_i - 1$, $w_{i,T} \leftarrow w_{i,S}$ and $w_{i,S} = h(u_i)^{n_i}$ and stores the set of data $\langle s_i, \{t_i^k\}_{k=0}^{m-1}, u_i, n_i, w_{i,T}, w_{i,S} \rangle$ in the entry $D_{new}[i]$.
5. If $\sigma_2 = \perp$, the reader stops. Otherwise, it forwards the received σ_2 to the tag.
6. The tag computes $w_{i,S}' = \sigma_2 \oplus f(s_i, r_2 \parallel r_1)$ and checks that $h(w_{i,S}') = w_{i,T}$. If the check succeeds, then the tag sets $c_i = 0$ and updates its secret and validator pair $\langle s_i, w_{i,T} \rangle$ as $w_{i,T} \leftarrow w_{i,S}'$ and $s_i \leftarrow g(s_i \oplus (w_{i,T} \parallel r_1 \parallel r_2))$. If the check fails, the tag increases the failure counter $c_i \leftarrow c_i + 1$ and, if $c_i < m$, updates its secret by $s_i \leftarrow g(s_i)$, and, if $c_i \ge m$, does nothing, keeping its current state unchanged.

Construction of Functions f, g and h. We may use a block cipher as a building block for construction of the functions f, g and h. Note that block ciphers usually allow more compact hardware implementations than hash functions and that we only need to implement the encryption function for our purpose. In particular, it was shown that a lightweight block cipher deigned for resource-constrained environments could be implemented using just a few thousand (NAND-equivalent) gates [20] (see also [10]).

Suppose that we have chosen parameters of $l = 128, l_1 = l_2 = 32$ and that we have a 64-bit block cipher with 128-bit key length, denoted by $E : \{0,1\}^{128} \times \{0,1\}^{64} \to \{0,1\}^{64}$. The function f can then be simply the encryption function E, so $f(k, x) = E_k(x)$. The one-way function g may be constructed by $g(k) = f(k, c_0) \parallel f(k, c_1)$ for some 64-bit constants c_0 and c_1. Similarly, we may construct the half-sized key chain function h by $h(k) = f(k \parallel k, c_2)$ for another 64-bit constant c_2. If the encryption function is secure, then inversion of functions g and h would require an exhaustive search for the keys involved. Also note that even if 32 bits of k are disclosed, there remain large enough secret bits so that an exhaustive search is still infeasible.

3.3 Security and Privacy

It is easy to see that the proposed protocol works correctly if both the tag and the backend server behave honestly. We show in this section that our protocol is secure, and backward and forward untraceable under some reasonable assumption. For the security analysis, we assume that the functions f, g and h are one-way and behave like random functions.

Attacks on tag authentication. We first consider possible attacks by the adversary impersonating a tag. The aim of the adversary in these attacks in the context of authentication is to make the backend database accept a fake tag as valid. First note that the tag secret constantly evolves in every query request from readers and thus past tag responses may be assumed uniformly distributed irrespective of the queries requested (thus of no use for future attacks). Thus a fake tag $\tilde{T}_i$ without knowledge of a valid secret s_i has no better strategy than to reply with random $\tilde{t}_i \in [0, 2^{\ell_2})$ and $\tilde{\sigma}_1 \in [0, 2^{\ell_1})$. Let N be the total number of tags managed by the backend server. Then the probability of this reply being accepted by the server is at most $2mN/2^{\ell_1+\ell_2}$ for each query response, since for a random tag identifier $\tilde{t}_i$ there exist at most $2mN/2^{\ell_2}$ matching tag identifiers in the database and for each matching tag identifier the probability of a random $\tilde{\sigma}_1$ being verified is $1/2^{\ell_1}$. Thus, our protocol achieves tag authentication with a cheating probability of at most $2mN/2^{\ell_1+\ell_2}$.

Attacks on reader authentication. We next consider possible attacks by the adversary impersonating a legitimate reader. The aim of the adversary in these attacks in the context of authentication is to make an honest tag accept the adversary as a legitimate reader. Possible results of a successful attack may include tracking a tag or illegal modification of tag's internal states. A fake reader without knowledge of valid secrets $(s_i, w_{i,S})$ associated with an honest tag T_i again has no better strategy than to send a random $\tilde{\sigma}_2$ in the final protocol flow. The probability of such a response being accepted by the tag is negligible (around $1/2^{2\ell_1}$ on average).

We now consider an adversary tampering with a tag T_i at some point. Suppose that the adversary somehow obtains T_i's secrets $(s_i, w_{i,T})$ at time t. Obviously, this information becomes useless if T_i has completed even one valid session (with a legitimate reader) at time $t' > t$ which the adversary could not eavesdrop, since then the tag secret s_i would have been refreshed with a random number of length $2\ell_1$ unknown to the adversary. We thus only consider the case of the adversary attacking T_i immediately after compromising the tag secrets. Even in this case, the adversary cannot successfully cheat T_i without knowledge of the corresponding server validator $w_{i,S}$. The only way to get a live value of $w_{i,S}$ would be to intercept a valid σ_2 (sent by a legitimate reader) and then to invalidate it immediately so that it cannot be accepted by T_i. Then the adversary can recover a live value of $w_{i,S}$ from σ_2 which can be used later to trigger a refresh of the tag secret by T_i. This is the only potential threat identified but inevitable in our protocol. However, the feasibility of such an attack requiring

instant intercept-then-invalidate operation over the air is highly questionable for proximate wireless transmissions in typical RFID tag environments.

Other attacks. Our protocol has strong resistance against Denial-of-Service (DoS) attacks on the last protocol message. Any alteration of this message may cause desynchronization of tag secrets in the tag and the backend server, but such a desynchronization problem can be resolved by dual copies of tag state information maintained in the server. Authentication failures more than the threshold m just cause the tag secret to remain static, which may break untraceability, but the tag secret is restored to a fresh after a secure reading of the tag.

Tag cloning is also ineffective in our protocol. Cloned tags are made useless as soon as tag secrets of the original tags are refreshed by a legitimate reading.

Forward and backward untraceability. From the above discussions, it is obvious that our protocol is backward untraceable and also forward untraceable under a reasonable assumption. Tag secrets at time t does not help identify tag interactions executed at time $t' < t$, as far as the one-wayness of the function g remains intractable, thus resulting in backwrad untraceability.

Forward untraceability is also provided to some extent under the natural assumption that the adversary compromising a tag cannot eavesdrop all the future interactions of the tag. The tag secret is refreshed upon every successful interaction with the server and thus a compromised tag becomes untraceable from the moment that the adversary misses even a single valid session of the tag. That is, forward untraceability of a tag, even if broken at any point during its lifetime, can be restored just by a single secure reading of the tag. However, we should be careful for the potential threat to this nice property as explained above (i.e., a possibility of intercept-then-invalidate of wireless signals over the air), since its realization would result in complete loss of control over the tag.

3.4 Efficiency Considerations

Suppose that the function f is implemented as the encryption function of a 64-bit block cipher supporting 128-bit keys and the functions g and h are derived from f as explained at the end of Section 3.2. Also suppose that we choose the parameters $\ell = 128$ and $\ell_1 = \ell_2 = 32$. Each tag T_i then needs to store 196 bits of rewritable data (a 128-bit s_i and a 64-bit $w_{i,T}$) and exchange 192 bits of data per session (send 96 bits and receive 96 bits). A tag requires 5 blocks of encryption for each session.

Consider an RFID system with $N = 2^{20}$ tags. The length m of precomputed key chains stored in the backend server controls untraceability between two refreshes and determines the storage requirement of the server; it need not be chosen too large, since the tag owner can refresh the tag secrets whenever necessary (e.g., $m = 64$). The server requires at least $(m + 3)$ invocations of the function g for each successful query from the reader to update the precomputation table entries. The length n for the backward key chain determines the number of valid authentications during the whole tag lifetime and thus may be

chosen quite large for long-lived tags, say $n = 2^{20}$. The server requires n_i invocations of the function h to update the current server validator $w_{i,S}$ in each successful authentication. This workload however can be reduced to a fixed c invocations if the server precomputes (say, at every midnight) and stores the c-th upward value in addition, assuming that at most c legitimate readings of the tag in a day are sufficient (e.g., $c = 100$). The server then requires storage of just around 620 Mbytes for tag identification data (i.e., $D_{old}[i]$'s and $D_{new}[i]$'s) and less than 250 evaluations of the encryption function for each successful query on average. Note that the storage requirement of the server never exceeds tens of gigabytes (still easily available even in potable devices such as PDAs) even if we take a larger value of m and store full values in both forward and backward key chains for computational efficiency and robustness against DoS attacks.

4 Ownership Transfer, Delegation and Anti-cloning

4.1 Ownership Transfer

Ownership of RFID tags may be changed frequently during their lifetime. For example, tags are initially created and attached to objects by manufacturers and tagged objects are then handed over to retailers, and finally consumers buy tagged objects in shopping malls. The owner of a tagged object may also transfer its ownership to another party (e.g., by buying an object and giving it to his friend as a birthday present, or by selling or swapping used objects via a garage sale or in a swap meet). Ownership transfer of a tag means transfer of authorization to read the tag. Thus, such an ownership transfer must guarantee that once ownership is transferred to another party, the old owner should not be able to read the tag any more.

The problem of ownership transfer seems not extensively studied in the RFID security community yet, but we argue that this problem is in fact in the core of the RFID privacy problem. Suppose that Alice bought an tagged item from a shopping mall. The ideal consequence of this transaction would be such that Alice should take over the complete control of the tag via her portable RFID reader[3] and then even the backend server of the shopping mall should not be able to trace the sold tag any longer. Once receiving all the necessary information to control the tag, the new owner's reader can now take the place of the server. Obviously, this problem is closely related to forward untraceability, since the backend server still maintains all the secret information on the tag, which should be made useless (at least in tracing the tag) after the sales transaction. This requirement is more obvious in the case of ownership transfer between users.

It is rather simple to transfer tag ownership in our protocol. Suppose that Alice wants to take over a tag T_i from the database server since she bought

[3] Commodity mobile RFID readers will be available soon in various mobile devices, such as mobile handsets and PDAs. In particular, mobile phones with built-in RFID readers will constitute the majority of mobile readers. Mobile phones could also be the best place to host RFID proxy for various personal belongings (e.g., see [26,18]).

the tagged item. Alice then uses her mobile reader to securely communicate with the server via the checkout reader and receives all the relevant information from the server via the secured channel. The information received will certainly include the T_i's table entries, $D_{old}[i]$ and $D_{new}[i]$, and the tag ID. Alice can then take over the ownership of the tag simply by reading the tag via her mobile reader. This will make the tag refresh its secret based on the randomness shared only with Alice's mobile reader and thus no one else can read the tag from the moment. Note that no eavesdropper, except the backend server, can refresh the tag secret, since the backward key chain needed for this operation is only known to Alice's reader (and the backend server; thus tag reading may be done outside the communication range of the checkout reader for safety).

We note that no previous work explicitly deals with this kind of perfect ownership transfer between users. Molnar et al.'s pseudonym protocol [23] is the only one we found that deals with the problem of ownership transfer explicitly. However, their method for ownership transfer is not complete in the sense that the backend server still maintains all the control power of the tag. Only partial information is delivered to a reader, so that the reader can read the tag by some predetermined number of times without on-line connectivity to the server. Thus, strictly speaking, their scheme corresponds to time-limited access delegation rather than ownership transfer. Contrary to this, ownership transfer in our protocol is perfect. Ownership transfer can be carried out every time tagged objects change hands during their lifetime.

4.2 Access Delegation and Anti-cloning

In most RFID authentication scenarios, an RFID reader tends to act as a dumb relay, just passing protocol messages back and forth between a tag and a central database server. This may overburden the server in both computing and communication complexities. Availability may also be a big issue; on-line access to the database server should be always available in a reliable way but readers may have intermittent connectivity for various reasons. It is however not easy to distribute the functionality of the database into a large number of readers scattered over the RFID infrastructure. Readers may not be always trustworthy and consistency of the database is hard to manage in such a large scale distribution.

Time-limited access delegation may be useful in such a case. We may delegate access to a set of tags to a particular reader, so that the reader can read the tags in a limited time span without on-line connectivity to the server. After the specified time span, however, the reader's access to the tags should expire and thus the reader should not be able to read the tags any more without interaction with the server. Molnar et al. proposed such a time-limited access delegation in their pseudonym protocol [23].

In our protocol, we propose a different approach to time-limited access delegation. Due to the probabilistic nature of tag secret evolution, it is not possible in our protocol to delegate access authorization that automatically expires after a specified time limit. However, our protocol has such a nice feature that only the server can refresh tag secrets using the backward key chain for server validators.

We may thus deliver only two forward key chains for the tag secret and the tag ID to the reader but not the backward key chain for server validation. Then, the reader may identify the tag from the first two protocol flows but cannot respond with a valid answer in the third flow. We note that our protocol works correctly even if the third protocol message is suppressed (the reader may send an arbitrary dummy answer as σ_2 for the tag to proceed without waiting until the timeout). This makes the tag secret evolved deterministically and eventually remain static after m readings. Thus the server never loses its control over the tag. Though untraceability may be broken by more than m readings, this may not be a problem, since untraceability is in fact not much necessary before tagged objects are handed over to consumers.

A remaining issue in the above scenario is how to cancel access authorization given to the reader. The solution is fairly simple. We can just read the tag as before using a reader on-line connected to the server to refresh the tag secret, so that the access delegated reader cannot read the tag any more. This method of distributed processing can be effectively used to manage a large volume of tags as well. For example, suppose the case of warehouse inventory of tagged objects. The database server makes a copy of its database only containing two forward key chains for the tag secret and the tag ID for each tag and delivers it to a local database or a set of readers used for inventorying the warehouse (even hand-held readers (say, PDAs) may have storage of several or tens of gigabytes, so they can easily hold a copy of the database). The distributed local databases can be made useless at any time by scanning the tags using a reader on-line connected to the central server as before.

The authorized refreshability of tag secrets in our protocol may also be used to thwart tag cloning. Suppose that a warehouse employee in the above example steals a set of tag secrets from the local database and reproduces cloned tags. These cloned tags however can be made easily obsolete by refreshing tag secrets in both the original tags and the central database, say, before tagged objects leave the warehouse. This time, we may need two readings in order to wipe out old tag secrets in the central database as well. Note that tag refresh cannot be done by the malicious employee, instead, to make the original tags obsolete, since he is not given any value of the backward key chain.

5 Conclusion

We have introduced the concept of forward untraceability and its importance in designing RFID security protocols. It has also been shown that forward untraceability is the key ingredient for perfect ownership transfer of RFID tags. Based on these observations and requirements, we presented a strong and robust RFID security protocol providing both forward and backward untraceability. As far as we know, our protocol achieves the strongest possible security in RFID authentication. The proposed protocol also has several nice features such as complete ownership transfer between users and distributed processing capability of the central database maintaining tag identification information. Though our protocol

may not be easy to implement in low-end RFID tags under the current standard and technology, we expect that it could be used right away in high-end tags for stronger security and probably low-end tags as well in the near future as hardware technology advances.

References

1. R. Anderson, "Two remarks on public key cryptology," Technical Reports, UCAM-CL-TR-549. Univ. of Cambridge, http://www.cl.cam.ac.uk/TechReports/, 2002.
2. G. Avoine, "Adversarial model for radio frequency identification," *Cryptology ePrint Archive*, Report 2005/049, 2005.
3. G. Avoine, Security and privacy in RFID systems (A complete list of related papers), http://lasecwww.epfl.ch/ gavoine/rfid/, Last Access: May 2006.
4. G. Avoine, E. Dysli, and P. Oechslin, "Reducing time complexity in RFID systems," *Selected Areas in Cryptography - SAC 2005*, LNCS 3897, Springer-Verlag, pp.291-306, 2005.
5. G. Avoine and P. Oechslin, "RFID traceability: A multilayer problem," *Financial Cryptography - FC 2005*, LNCS 3570, Springer-Verlag, pp.125-140, 2005.
6. G. Avoine and P. Oechslin, "A scalable and provably secure hash based RFID protocol," *The 2nd IEEE International Workshop on Pervasive Computing and Communication Security - PerSec 2005*, IEEE Computer Society Press, pp.110-114, 2005.
7. Boycott Benetton Home Page, http://www.boycottbenetton.com/, 2003.
8. T. Dimitriou, "A lightweight RFID protocol to protect against traceability and cloning attacks," In *IEEE SecureComm*, pp.59-66, 2005.
9. EPCglobal Web site, http://www.EPCglobalinc.org, 2005.
10. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong authentication for RFID systems using the AES algorithm," *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2004*, LNCS 3156, Springer-Verlag, pp.357-370, 2004.
11. K. Finkenzeller, *RFID Handbook*, John Wiley & Sons, 1999.
12. D. Henrici and P. Müller, "Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers," *Workshop on Pervasive Computing and Communications Security - PerSec 2004*, IEEE Computer Society, pp.149-153, 2004.
13. A. Juels, "Minimalist cryptography for low-cost RFID tags," *The Fourth International Conference on Security in Communication Networks - SCN 2004*, LNCS 3352, Springer-Verlag, pp.149-164, 2004.
14. A. Juels, "RFID security and privacy: a research survey," IEEE Journal on Selected Areas in Communications, 2006.
15. A. Juels, S. Garfinkel, and R. Pappu, "RFID privacy: An overview of problems and proposed solutions," *IEEE Security and Privacy*, 3(3):34- 43, 2005.
16. A. Juels, D. Molnar, and D. Wagner, "Security and privacy issues in e-passports," *IEEE SecureComm'05*, IEEE, 2005, Referenced 2005 at http://www.cs.berkeley.edu/ dmolnar/papers/papers.html.
17. A. Juels, R.L. Rivest, and M. Szydlo, "The blocker tag: Selective blocking of RFID tags for consumer privacy," *8th ACM Conference on Computer and Communications Security*, ACM Press, pp.103-111, 2003.
18. A.Juels, P.Syverson and D.Bailey, High-power proxies for enhancing RFID privacy and utility, In *Privacy Enhancing Technology 2005*, LNCS 3856, Spinger-Verlag, 2005, pp.210-226.

19. A.Juels and S.Weis, Authenticating pervasive devices with human protocols, In *Advances in Cryptology-Crypto 2005*, LNCS 3621, Spinger-Verlag, 2005, pp.293-308.
20. C.H.Lim and T.Korkishko, mCrypton-A lightweight block cipher for security of low-cost RFID tags and sensors, In *WISA 2005*, LNCS 3786, Spinger-Verlag, 2006, pp.243-258.
21. S. Lee, Y. Hwang, D. Lee, and J. Lim, "Efficient authentication for low-cost RFID systems," *Computing System Applications*, LNCS 3480, Springer-Verlag, pp.619-627, 2005.
22. T.Malkin, S.Obana and M.Yung, The hierarchy of key evolving Signatures and a characterization of proxy signatures, In *Advances in Cryptology-EUROCRYPT 2004*, LNCS 3027, pp.306-322. Springer-Verlag, 2004.
23. D. Molnar, A. Soppera, and D. Wagner, "A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags," *Selected Areas in Cryptography - SAC 2005*, LNCS 3897, Springer-Verlag, pp.276-290, 2005.
24. D. Molnar and D. Wagner, "Privacy and security in library RFID : Issues, practices, and architectures," *ACM Conference on Communications and Computer Security*, ACM Press, pp.210 - 219, 2004.
25. M. Ohkubo, K. Suzuko, and S. Kinoshita, "Cryptographic approach to "privacy-friendly" tags," In *RFID Privacy Workshop*, 2003.
26. M.Rieback, B.Crispo and A.Tanenbaum, RFID guardian: A battery-powered mobile device for RFID privacy management, In *ACISP 2005*, LNCS 3574, Springer-Verlag, pp.184-194, 2005.
27. S. Sarma, S. Weis, and D. Engels, "RFID systems and security and privacy implications," *Workshop on Cryptographic Hardware and Embedded System - CHES 2002*, LNCS 2523, Springer-Verlag, pp.454-469, 2002.
28. R. Shirey, "Internet Security Glossary," IETF RFC 2828, at http://www.ietf.org/rfc/rfc2828.txt, May 2000.
29. Stop RFID, http://www.stoprfid.org.
30. S. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," *International Conf. on Security in Pervasive Computing - SPC 2003*, LNCS 2802, Springer-Verlag, pp.454-469, 2003.

A The Security Model for Untraceability

Not much work has been done yet on formal security models for RFID security protocols, though a number of proposals based on varying assumptions have been proposed and analyzed in the literature. Juels have proposed a somewhat weak but realistic security model specific to his pseudonym protocol [13]. On the other hand, Avoine introduced a rather strong cryptographic model to define the strong privacy notion of untraceability in RFID protocols [2]. Here, we would like to take a step toward a generic security model that can cover weak to strongest possible security in RFID protocols. Our model closely follows Avoine's model but makes it more general and flexible by incorporating various possible restrictions existing in RFID systems. We primarily focus on the strongest privacy notion of untraceability, in particular, forward and backward untraceability, but it would be rather easy to extend it to include the general notion of authentication as well.

Oracles. An RFID protocol, $\mathcal{P}$, is formally a probabilistic algorithm that determines how instances of the principals, a tag T and a reader R, behave in response to inputs sent from their environment. The inputs may be given by an adversary, $\mathcal{A}$, that may have complete control over the environment. The adversary is a probabilistic algorithm with a distinguished query tape. Queries written on this tape are answered by principals according to $\mathcal{P}$. Each of the principals can run several instances of $\mathcal{P}$. We denote a tag instance at time i by π_T^i and a reader instance at time j by π_R^j. The adversary $\mathcal{A}$ is allowed to have access to the following oracles:

- $\mathsf{Query}(\pi_T^i, m_1, m_3)$: this query models $\mathcal{A}$ actively querying T. It sends a request m_1 to T through the forward channel and subsequently responds with the message m_3 after having received an answer from T.
- $\mathsf{Send}(\pi_R^j, m_2)$: this query models $\mathcal{A}$ actively querying R. It sends the message m_2 to R through the backward channel and receives an answer from R.
- $\mathsf{Execute}(\pi_T^i, \pi_R^j)$: this query models $\mathcal{A}$ passively eavesdropping on the communication channels between T and R. It executes an instance of $\mathcal{P}$ between T and R and obtains the messages exchanged on both the forward and the backward channels.
- $\mathsf{Execute}^*(\pi_T^i, \pi_R^j)$: this query models $\mathcal{A}$ passively eavesdropping only on the forward channel. It executes an instance of $\mathcal{P}$ between T and R, but only obtains the messages exchanged on the forward channel. This oracle may be used to model the secure backward channel assumption in some protocols.
- $\mathsf{Reveal}(\pi_T^i)$: this query models $\mathcal{A}$ obtaining the content of T's memory channel. This query is only allowed during the time interval of $\mathcal{A}$'s training phase. Other queries can still be used even after the reveal query (possibly under some restriction).

For simplicity, let Q,S,E,E^*, and R represent, respectively, the oracles Query, Send, $\mathsf{Execute}$, $\mathsf{Execute}^*$ and Reveal. Note that the adversary is passive when using the oracles E and E^*, while it is active when using the oracles Q and S. In fact, the oracle E may be simulated using the oracles Q and S by the man-in-the-middle attack, but the reverse is not true.

Attack Models. The adversary attacking untraceability may use the above oracles in arbitrary manner according to its strategy, except the reveal oracle. The aim of the adversary is to distinguish a particular tag or a set of tags from others in different instances of the protocol (a more formal definition will be given below). The adversary's ability to achieve this aim can be characterized by the oracles to which the adversary is given access. We can thus classify the attack models according to the set of oracles available to the adversary. For example, the QSE model and the QSER model may be most intersting among others.

In typical RFID system environments, however, it may be too strong to allow unrestricted access to the oracles provided. Typically, tags and readers operate only at short communication range and for a relatively short period of time. Furthermore, RFID tags may be assumed in many cases highly mobile and thus

hard to trace physically (otherwise, we do not need to worry about privacy infringement due to the traceability of tagged items). Therefore, it may not be unrealistic in practice to put some restrictions to the adversary's oracle access in terms of access time and frequency. We thus consider two access models, namely the universal or unrestricted access (UA) model and the restricted access (RA) model. Access restriction in the RA model is hard to define generically and thus a specific RA model should include a description on the imposed restrictions. For example, natural restrictions that could be imposed on the adversary may include some limitation in the number of successive queries to target tags and some limitation in the number of successive valid sessions that can be read by the adversary (e.g., as in the minimalist security model [13]).

We can thus talk of untraceability under the combined model of available oracles and access restriction. For example, we can say that a given protocol is untraceable under the UA-QSE model, meaning that the protocol is untraceable against any adversary who can interact with target tags and readers and eavesdrop interactions between target tags and legitimate readers at any time it wishes.

Attack Games. Let $\omega_i(T)$ be the result of the application of an oracle Q,E,E^*, or R on a tag T, where $\omega_i(T) \in \{\mathsf{Query}(\pi_T^i, *), \mathsf{Execute}(\pi_T^i, *), \mathsf{Execute}^*(\pi_T^i, *), \mathsf{Reveal}(\pi_T^i)\}$ (note that tags are involved in all oracles, except the Send oracle). A tag *interaction* is defined as a set of oracle execution results on the same tag (identified by physical tracing or through an active query). More precisely, an interaction is defined by $\Omega_I(T) = \{\omega_i(T) | i \in I\} \cup \{\mathsf{Send}(\pi_*^i, *) | i \in J\}$, where $I, J \subset \mathbb{N}$. The length of an interaction $\Omega_I(T)$ is equal to $|I|$ by definition. We use the notation $I < J$ for $I, J \subset \mathbb{N}$ to denote that numbers in I precede those in J; i.e., $I = [a, b], J = [c, d]$ such that $a \leq b < c \leq d$.

Untraceability can be formally defined by a game being played between a *Challenger* and an adversary $\mathcal{A}$, where the adversary is allowed to interact with given oracles. The game begins with the *Challenger* randomly choosing a target tag T and providing it to the adversary $\mathcal{A}$. After having experimented with the target T using a set of oracles $\mathcal{O}$ provided (possibly including the reveal oracle R) under some access restriction $\mathcal{R}$ and thus obtaining an interaction $\Omega_I(T)$ over its chosen interval I, the adversary $\mathcal{A}$ requests challenge tags from the *Challenger*, which then provides two tags T_0 and T_1, one of which is T (i.e., $T_b = T$ for some hidden bit $b \in \{0, 1\}$). The adversary continues experimenting with the two tags as before, except that the reveal oracle, if provided, is not allowed to query in this stage, and thus obtains two interactions $\Omega_{I_0}(T_0)$ and $\Omega_{I_1}(T_1)$, where the intervals I_0 and I_1 should not overlap with the interval I. Finally, $\mathcal{A}$ outputs her best guess b' based on the experiments. If the probability of $b = b'$ is negligible for every I_0 and I_1 and for every $\mathcal{A}$, then we say that the protocol is *untraceable under the $\mathcal{R}$-$\mathcal{O}$ model*.

More formally: Let $\mathcal{O}$ be the set of oracles available to the adversary, where $\mathcal{O} \in \{\mathsf{E}^*, \mathsf{E}, \mathsf{QS}, \mathsf{QSE}^*, \mathsf{QSE}, \mathsf{QSER}\}$, and let $\mathcal{R}$ be the (description of) access restriction imposed on the oracles, where $\mathcal{R}$ may be empty. Let $\mathcal{O}'$ be the same as $\mathcal{O}$ except that $\mathcal{O}' = \mathsf{QSE}$ in the case of $\mathcal{O} = \mathsf{QSER}$. Let $\mathcal{O}_{\mathcal{R}}$ be the set of oracles

$\mathcal{O}$ under the access restriction $\mathcal{R}$. That is, when a query is received, $\mathcal{O}_{\mathcal{R}}$ first checks the given access restriction $\mathcal{R}$ and returns an answer only if it satisfies the restriction. We use *Oracle* to simulate the set of oracles to which the adversary has access. More precisely, *Oracle* takes as inputs a tag T and a time interval I, makes calls to the oracles of $\mathcal{O}_{\mathcal{R}}$ or $\mathcal{O}_{\mathcal{R}}{}'$ and sends back an interaction $\Omega_I(T)$. Let ℓ_{ref} and ℓ_{ch} be security parameters controling the length of interactions that the adversary can execute during the training and the cracking phases of the experiment. We first define a general experiment for untraceability, the strongest privacy notion, and then define forward and backward untraceability based on this experiment.

1. The *Challenger* randomly picks a target tag T (among all possible random tags) and gives it to the adversary $\mathcal{A}$, together with the permission of oracle accesses to $\mathcal{O}_{\mathcal{R}}$.
2. $\mathcal{A}$ chooses I and calls $Oracle(T, I, \mathcal{O}_{\mathcal{R}}\})$ where $|I| \leq \ell_{ref}$, and gets $\Omega_I(T)$.
3. $\mathcal{A}$ requests the *Challenger* to provide challenge tags and receives T_0 and T_1 such that $T_b = T$ for a hidden bit $b \in \{0, 1\}$.
4. $\mathcal{A}$ chooses I_0 and I_1 such that $|I_0|, |I_1| \leq \ell_{ch}$ and $(I_0 \cup I_1) \cap I = \emptyset$, calls $Oracle(T_0, I_0, \mathcal{O}_{\mathcal{R}}{}')$ and $Oracle(T_1, I_1, \mathcal{O}_{\mathcal{R}}{}')$, and gets $\Omega_{I_0}(T_0)$ and $\Omega_{I_1}(T_1)$.
5. $\mathcal{A}$ finally outputs her best guess b'.

Note that the reveal oracle R, if provided, can only be queried during the training phase (step 2) but never in the cracking phase (step 4). The advantage of $\mathcal{A}$ for a given protocol $\mathcal{P}$ under the $\mathcal{R}$-$\mathcal{O}$ model is defined by $\mathsf{Adv}_{\mathcal{P}}^{\mathcal{UNT}}(\mathcal{A}^{\mathcal{O}_{\mathcal{R}}}) = \Pr(b' = b) - \frac{1}{2}$, where the probability is taken over the coin tosses of $\mathcal{A}$ and *Challenger* and over the choice of random intervals and random tags. In general, we say that $\mathcal{P}$ is $\mathcal{O}$-untraceable under the restriction $\mathcal{R}$ if this advantage is negligible w.r.t the security parameters ℓ_{ref} and ℓ_{ch}. We simply say that $\mathcal{P}$ is untraceable if $\mathcal{R} = \varnothing$ (the UA model) and $\mathcal{O} = \mathsf{QSE}$, since this is the best achievable untraceability notion without the reveal oracle.

When the reveal query is allowed (i.e., if $\mathcal{O} = \mathsf{QSER}$), we make a further distinction according to the additional restriction on the choice of experiment time intervals. If I, I_0 and I_1 are chosen such that $I > I_0$ and $I > I_1$, then the protocol is said to be backward untraceable under the restriction $\mathcal{R}$ (simply backward untraceable if $\mathcal{R} = \varnothing$). If the restriction is such that $I < I_0$ and $I < I_1$, then we say that the protocol is forward untraceable under the restriction $\mathcal{R}$. Note that forward untraceability under the UA model makes no sense, since once obtaining the tag secret by the reveal query, the adversary takes all the power of the tag itself and thus can trace the target tag at least during the authentication immediately following the attack. Thus, the minimum restriction for forward untraceability is such that there should exists some non-empty gap not accessable by the adversary between the time of a reveal query and the attack time. That is, there should exist some non-empty intervals J_0 and J_1 such that $I < J_0 < I_0$ and $I < J_1 < I_1$. Forward untraceability under this restriction would be the best one we can achieve in any RFID authentication protocol.