A Lightweight RFID Protocol to protect against Traceability and Cloning attacks

Tassos Dimitriou
Athens Information Technology
19.5km Markopoulo Ave., 19002, Peania
Athens, Greece
tdim@ait.edu.gr

Abstract

RFID identification is a new technology that will become ubiquitous as RFID tags will be applied to every-day items in order to yield great productivity gains or "smart" applications for users. However, this pervasive use of RFID tags opens up the possibility for various attacks violating user privacy.

In this work we present an RFID authentication protocol that enforces user privacy and protects against tag cloning. We designed our protocol with both tag-to-reader and reader-to-tag authentication in mind; unless both types of authentication are applied, any protocol can be shown to be prone to either cloning or privacy attacks. Our scheme is based on the use of a secret shared between tag and database that is refreshed *to avoid tag tracing. However, this is done in such a way so that* efficiency *of identification is not sacrificed. Additionally, our protocol is very simple and it can be implemented easily with the use of standard cryptographic hash functions.*

In analyzing our protocol, we identify several attacks that can be applied to RFID protocols and we demonstrate the security of our scheme. Furthermore, we show how forward privacy *is guaranteed; messages seen today will still be valid in the future, even after the tag has been compromised.*

1 Introduction

Radio frequency identification (RFID) is a method of remotely storing and retrieving data using small devices called RFID tags. In a typical RFID system, each object is equipped with such a small, inexpensive tag. The purpose of the system is to enable data transmitted by the tag to be read by a reader and processed according to the needs of a particular application. Examples of data may include identification, location information or details about the tagged object such as price, color, date of purchase, and so on.

RFID tags are often envisioned as a replacement for bar-codes, having a number of important advantages over the older bar-code technology. Apart from their small size which allows them to be implanted within objects, identification by frequency allows objects to be read in large numbers without the need for a visual contact. Furthermore, RFID identifiers are long enough so that every object has a *unique* code. Such universal uniqueness means that a product may be tracked as it moves from location to location, finally ending up in the consumer's hands. This may help companies combat theft or improve management of stock and inventories in shops or warehouses [1, 2]. Additionally, the introduction of RFID tags in all objects could also directly benefit the consumer: waiting times at checkout lines may be drastically reduced by the use of reader technology that requires no bar-code scanning.

As RFID capabilities are refined, more pervasive (and invasive) uses for tags can be envisioned. It should be clear that in order for RFID tags to prevail and become a ubiquitous technology certain security problems must be overcome and most importantly violation of *user privacy*. Consumer products labelled with insecure tags may reveal sensitive information when queried by nearby readers. Examples include money, medicine (which may link to a particular disease), books (which may indicate a particular political preference), and so on. However, a closely related problem to that of violating consumer privacy is *location privacy* which may lead to tracking of individuals by the tags they carry. This is so because even when tag contents are secured, tag responses may help identify the location of the holder by correlating data from multiple readers at fixed locations. Something like that may eventually lead to consumer identification, especially when the consumer is buying the tagged product using some form of payment that reveals his or her identity (e.g. credit card).

In addition to the privacy risks imposed by the incorrect use of RFID tags, RFID deployment may suffer from tag cloning. As a motivating example [3], consider an attacker that spoofs a valid tag in an attempt to remove an item without being noticed. If the attacker can install a replacement tag which can continue authenticating itself to a reader, then the attacker can fool the system into believing the product is still on the shelf. Alternatively, an attacker can replace tags for expensive items with tags for cheaper ones.

Our contribution

Our contribution in this work is twofold: First, we propose a lightweight protocol that can be used for authenticating the tags while at the same time avoiding traceability. Our protocol allows for both tag-to-reader and reader-to-tag authentication. The first form of authentication is needed in order to prevent the tag cloning attack mentioned above while the second is needed in order to prevent queries by unauthorized readers which can be used in violating user privacy. Furthermore, we make sure that tag output is not fixed and cannot be in any way related to the ID of the tag, thus providing ID anonymity and helping prevent tag tracing. However, these ideas alone are not enough to guarantee privacy. RFID tags are inexpensive devices that offer no tamper resistance, hence they suffer from physical attacks that attempt to expose their memory contents. An attacker upon compromising a tag may be able to link this tag with past actions performed on the tag. So, we also designed our protocol with *forward privacy* in mind; messages transmitted today will still be secure in the future, even after compromising the tag.

Second, we analyze our protocol against a multitude of attacks that seem important in designing new RFID protocols and we identify *mutual authentication* as one of the key challenges in the area. Tags must reveal their identities only to authorized readers but this should happen only if the reader has been authenticated to the tag. However, there is a chicken-and-egg problem here: in order to eliminate spoofing attacks, the reader can only authenticate itself if it knows the secret of the tag; but this requires knowledge of the tag's ID. However, the tag cannot reveal its ID unless the reader has already been authenticated to it. In our work we show how to solve this paradox using standard cryptographic challenges. We also emphasize on *efficient* tag identification and point to several issues that need to be taken care when designing a secure RFID protocol.

The rest of the paper is organized as follows: In the next section we summarize previous work on the area and identify inefficiencies of existing protocols. In Section 3, we make a first attempt to address the problems mentioned in the introduction by designing a simple protocol that satisfies most of the requirements. However, this simplified version suffers from replay attacks that aim at destroying the proper operation of the tags. We show how to overcome these problems by introducing an enhanced protocol in Section 4. There, we discuss our assumptions about the threat model and we prove our protocol secure against many types of attacks. Finally, we conclude in Section 5.

2 Related Work

There have been many papers in the literature that attempt to address the security concerns raised by the use of RFID tags. The Hash Lock scheme [3] stores the hash of a random key K as the tag's $metaID$, i.e. $metaID = h(K)$. When queried by a reader, the tag transmits its $metaID$ which is forwarded to the back-end database. The database and eventually the reader responds with K. The tag hashes the key and compares it to the stored $metaID$. If the values match, it unlocks itself and offers its full functionality to the nearby readers. Although this scheme offers good reliability at low cost, an adversary can easily track the tag via its $metaID$. Furthermore, since the key K is sent in the clear, an adversary upon capturing the key can later spoof the tag to the reader.

A similar *randomized* version in [3] attempts to disguise the ID so that tag output is not fixed over time. Upon query, the tag responds with the pair $(r, h(ID, r))$, where r is a randomly generated number. Although this scheme deters tracking, it requires brute force search of known IDs in the database until it finds one that matches the signature $(r, h(ID, r))$. Even though this is an issue for large databases, it is also possible to apply the following very simple attack: An adversary can first query the tag to learn a valid pair $(r, h(ID, r))$. Then it can impersonate the tag by forwarding these values to a legitimate reader. This is a serious security flaw as the reader will identify the tag. In addition, the scheme allows the location history of the tag to be traced if the tag itself is compromised. Hence forward secrecy is not guaranteed.

In [4], the authors devise a scheme that uses a low cost *hash-chain* mechanism to defeat the tracing problem and ensure forward security in tag transactions. The basic idea is to modify the identifier each time the tag is queried so that the tag is recognized by authorized parties only. The scheme uses two hash functions, one to refresh the secret in the tag, the other to make responses of the tag untraceable by eavesdroppers. Hence this scheme can be seen as an extension to the randomized version in [3] that guarantees security of past transactions. However, just as in the previous scheme scalability is problematic as it requires exhaustive search in the back-end database to locate the ID of the tag. Although in [5] a time-space memory tradeoff is presented that reduces the scalability problem, perhaps more important from a security point of view is that an attacker can still query the tag then replay the tag's response to authenticate itself to a valid reader.

In [6], a security model is proposed that introduces a *challenge-response* mechanism which uses no cryptographic primitices (other than simple XORs). One of the key ideas in this work is the application of pseudonyms to help enforce privacy in RFID tags. Each time the tag is queried, it releases the next pseudonym from its list. In principle, then, only a valid verifier can tell when two different names belong to the same tag. Of course, an adversary could query a tag multiple times to harvest all names so as to defeat the scheme. So, the approach described involves some special enhancements to help prevent this attack. First, tags release their names only at a certain (suitably slow) prescribed rate. Second, pseudonyms can be refreshed by authorized readers. Although this scheme does not require the tags to perform any cryptographic functions (it uses only XOR operations) the protocol involves four messages and requires updating the keys and pads with new secrets, an operation which may be costly and difficult to realize.

The authors in [7] proposed a set of authentication protocols built from primitives that can be supported by low-cost RFID tags. Although these protocols can be used for authenticating the tags, they can be broken by a powerful adversary; hence, emphasis was placed on coming up with lower bounds on the abilities of an attacker and a trade-off between security and performance. Furthermore, these protocols relied on the existence of a shared secret between tag and reader, which makes it problematic for the reader to determine what secret corresponds to what tag. Most importantly, however, these protocols do not address the problem of reader-to-tag authentication and no attempt is being made to prevent tracking of the tags.

Another protocol that attempts to disguise the IDs of the tags using hash functions is given in [8]. When queried, the tag sends the hash of its ID, $h(ID)$, together with a commitment that is basically a function of its ID and the current transaction number. The reader forwards these values to the database which locates the ID and checks the validity of these values. Then it responds with a random number that can be used in refreshing the tag identifier. While this scheme allows for both tag-to-reader and reader-to-tag authentication it is subject to an attack that was demonstrated in [9]. Furthermore, due to a *side channel* created by the use of counter values, the tag can still be traced even if its ID changes with every valid transaction.

From the discussion above it seems necessary to have protocols that incorporate a form of *challenge-response* mechanism to avoid simple authentication attacks. In this paper we propose a scheme that provides for both tag-to-reader and reader-to-tag authentication and ensures forward privacy of transactions. Our scheme uses cryptographic hash functions to refresh tag IDs and to make responses indistinguishable from random values. The use of hash functions or pseudo-random ones has been applied to many past works [3, 4, 8, 10]. Although it is a general belief that cryptographic components may not be suitable for low-cost tags, recent work [11] has led to more compact implementations of symmetric key primitives and in particular the Advanced Encryption Standard (AES) for RFIDs. This result may lead to manifestation of real tags capable of performing challenge-response protocols. In addition to that, [6] points out the existence of moderate cost RFID tags that already perform challenge-response protocols.

3 A Simplified Protocol

In this section we proceed to discuss our authentication protocol. We start by presenting a simpler protocol that achieves many of the security requirements presented in the introduction but falls prey to a simple replay and *desynchronization* attack aiming at disrupting tag functionality. However, this simple protocol can be augmented in a straightforward way to achieve security against all possible attacks. Furthermore, despite its simplicity this protocol is already an improvement of the basic schemes proposed in [3, 4] offering properties such as efficiency of identification, privacy and forward security.

In the rest of the paper the following notation will be used:

Notation	**Meaning**
T	The tag participating in the protocol.
R	The reader querying the tag.
ID_i	The ID of the tag during the i-th query.
M_1, M_2	Concatenation of messages M_1 and M_2.
$h(M)$	Application of hash function h to M.
$h_K(M)$	*Keyed* hash function; The subscript denotes the key that was used to produce the hash value. Essentially this corresponds to a *Message Authentication Code* (MAC) used in authenticating the origin of M.
N	Nonce identifier.

3.1 On the use of secrets in tags

It should be noted that both the simple and the enhanced version rely on the existence of a *secret* shared between the tag and the reader (back-end database). In general, this secret can be common for all tags, however compromise of even a single tag leads to compromising the entire system. Another possibility is to have different secrets per tag. The disadvantage in this case is that a mechanism is required to allow the reader to determine which secret was used for which tag. Unfortunately, most obvious approaches either send an index to the system database (which opens the possibility of tracing) or require the database to *exhaustively search* over the collection of keys to identify the correct tag. However, the last option does not scale well if the system needs to accommodate a large number of tags. In our case, the secret information is the ID of the tag, but we use the *hash value of the ID* to search and identify the tag.

In general, our scheme relies on the following observation: If the ID of a tag changes in a secure manner after every read query then attempts like eavesdropping, spoofing, replaying messages, etc. cannot compromise the security of the scheme. Our protocol is reminiscent of the Hash lock scheme [3] (or the schemes in [4, 8]) where a *metaID* is used to identify the tag. However, our scheme does not suffer from the obvious security disadvantage of tag traceability as the tag ID *changes* with every application of the protocol. We achieve this through the use of secure one-way hash functions and random session identifiers so that tag responses remain untraceable. Here we define a secure one-way hash function h as one that is difficult to invert and does not leak any information about the message it is applied [12]. The simpler version of the protocol is shown in Figure 1.

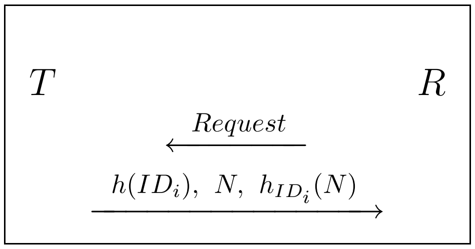

Figure 1. A simple protocol ensuring user privacy.

3.2 Initialization and Identification process

We assume that during system initialization the tag is loaded with an initial identifier ID_0 which is set to a random value. In a similar way, the back-end database contains the same data stored in the tag, together with a hash value of its ID, $h(ID_0)$, that serves as the *main key* to look for any information related to this particular tag. During normal operation, the tag is singularized by any standard method like binary tree-walking or any other anti-collision protocol [13]. In this stage of operation a tag exposes no information other than the *hash* of its ID that is used for identifying and addressing the tag. The identification process goes as follows (Figure 1):

1. The reader sends a request to the tag.

2. The tag generates a new nonce identifier N and sends back $h(ID_i)$, N and $h_{ID_i}(N)$ to the reader which then forwards these values to the back-end database.

   The database upon receiving $h(ID_i)$ uses this value to search and recover the identity ID_i of the tag. Once the database has the ID_i value, it can use the last part $h_{ID_i}(N)$ to connect all the pieces together and verify the authenticity of the message. This last step is vital as it acts as a message authentication code (recall the notation $h_K(M)$) preventing cloning attacks by an adversary[1].

At this point the database accepts the tag as authentic and *renews* the identity of the tag. Since the value ID_i is a secret shared by the tag and the database, refreshment of the ID can be done in any way that uses an irreversible computation, thus obtaining ID_{i+1}. The tag also performs the same computation, thus refreshing its identity, and *erases* any relevant information (N and ID_i) from its memory.

3.3 Security Analysis

It is obvious that our scheme satisfies many of the security requirements we mentioned in the introduction, in particular protection against tracing and forward security. This is so because every time the tag is queried, the adversary only sees $h(ID_i)$ which is going to be used only *once* to identify the tag. Furthermore, the one-wayness of h makes it impossible for any adversary to recover the original information and link ID_i to ID_{i+1}. In addition to that, even if the tag is compromised, an adversary cannot link the current ID to past actions as any relevant information has been deleted from the memory of the tag. Thus, forward security is guaranteed.

The security of the scheme relies on the secrecy of ID_i which acts as a shared secret between the tag and the system database. If the tag is compromised, this secret is revealed and hence an adversary may spoof the tag to the database. However, this should not be considered an attack as it can happen with any protocol where the ID of the tag is revealed. Furthermore, even in the case of an ID compromise, security of past transactions is guaranteed.

On the use of nonces

The identifier N can be implemented using a timestamp to prevent replay attacks. However, this imposes clock synchronization problems which in the case of a collection of readers it may be difficult to enforce. Furthermore, in the

[1]There are many ways to construct a Message Authentication Code from any secure one-way function h using a symmetric key K. However, one way that is provably secure is the HMAC construct[14].

case of simple tags it may be difficult to implement. An alternative way is the use of a counter that increases each time the tag is queried. However, this opens up the following avenue for an attack: An attacker observes the counter value in a successful identification of the tag and use this knowledge to trace the tag in subsequent transactions. Even if the tag ID is not known to the attacker, successive updates of the counter value create a *side channel* that can be used by the adversary to track the tag. We believe it is best for N to be a newly created random number. As the shared secret ID is updated each time the tag is queried, even if an attacker replays messages corresponding to old IDs in an attempt to spoof the tag to the reader, authentication will fail as each secret ID is used only once. Hence the system really authenticates the tag and ensures freshness of messages.

Replay attack

However, the attacker can still *replay* a message containing an ID that was never used before. Then it can appear as a valid tag to a reader or destroy tag functionality as we demonstrate below.

Database Desynchronization

This scheme falls prey to a *denial of service* attack aiming at desynchronizing the tag and the database. An attacker only has to spoof the reader to the tag by sending a request for read (first message in Figure 1). Although the response of the tag will be of no value to the attacker as it can recover no useful information, the tag following the protocol will refresh its ID (the secret information) updating the secret value from ID_i to ID_{i+k}, where k is the number of spoofed attempts made by the attacker. As a result, in the next identification the tag and the database will be desynchronized since the database tries to locate $h(ID_{i+k})$, an entry that does not exist, and discards the received message. From this point on the tag looses its functionality. It is exactly this reason why *any attempt for read must be authenticated by the tag* as well. As we will see next, the enhanced version of the protocol allows for both tag-to-reader and reader-to-tag authentication and eliminates these attacks.

4 The Enhanced protocol

The enhanced version is shown in Figure 2. Observe that this is basically the simpler protocol augmented with an extra message whose purpose is to authenticate the reader to the tag. This extra step essentially allows the tag to disregard any queries made by unauthorized readers. The identification goes as follows:

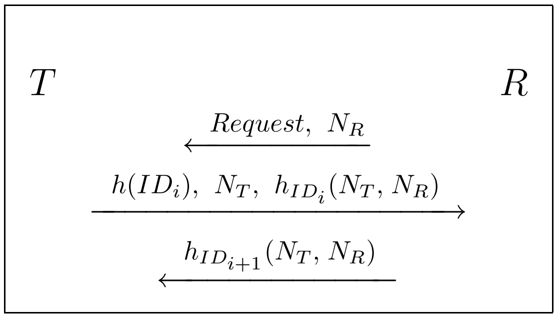

Figure 2. A challenge response protocol for tag-reader authentication.

1. Request by the reader. The reader also transmits a nonce N_R whose purpose will be explained in Section 4.1.

2. The tag generates a new nonce identifier N_T and sends back $h(ID_i)$, N_T and $h_{ID_i}(N_T,\ N_R)$ to the reader which then forwards these values to the back-end database.

   The database authenticates the tag and if everything checks out, it computes the new identity, ID_{i+1}. Notice, that the reader has to forward the nonce N_R to the database as otherwise it will not be able to compute and compare the MAC $h_{ID_i}(N_T,\ N_R)$. An implicit (and standard) assumption we make here is that the channel between the reader and the database is a secure one.

3. The back-end database constructs the message $h_{ID_{i+1}}(N_T,\ N_R)$ using the new key ID_{i+1}. It then sends this message to the reader who forwards it to the tag. Upon reception, the tag generates the new key on its own and computes the value $h_{ID_{i+1}}(N_T,\ N_R)$.

   If the value received is the same as the value computed, it accepts the response as authentic and only then it deletes the old key ID_i and the nonce N_T from its memory. Otherwise, it rejects the answer and keeps the old key ID_i.

4.1 Security Analysis

It is clear that the third message is used to protect the tag from desynchronizing from the database. A spoofed reader may try to query the tag, but it's only after the reception of the third message that the tag will update its secret. Hence any attempts to read the tag by unauthorized readers will have no effects on the tag functionality. We now proceed to examine one by one the attacks that can be applied to this protocol.

- *Attack on the tag*. This can happen only when the attacker masquerades as a valid reader. This type of attack is defeated by the shared secret which is not known to the attacker. As a result the attacker will not be able to produce the third message which requires knowledge of the secret ID and the tag will not authenticate the reader.

- *Attack on the reader*. In this case the goal of the attacker is to masquerade as a valid tag. Again this type of attack is not possible due to the shared secret. Even if the attacker replays old information, a reader will not accept the tag as authentic since the database will fail to recognize the secret ID.

  This is where the use of N_R is crucial. If the request from the reader (first message in Figure 2) didn't include the nonce identifier N_R an attacker could perform the following attack aiming at desynchronizing the tag and the database. The attacker could send a read request, thus receiving back (and storing) the message

  $$h(ID_i),\ \ N,\ \ h_{ID_i}(N)$$

  Although the tag would eventually reject the read attempt, the attacker could *replay* this value later on to the reader and hence to the database. The database then would validate the message and compute a new secret ID_{i+1}. The reason for this desynchronization is that the database has no way to know whether the received message is the result of a valid read attempt. Hence the need for inclusion of the nonce N_R in the read request and the transmission of this value in the secure channel between the reader and the database.

- *Attack on the communication between tag and reader*. Listening to messages exchanged in a particular session reveals no information because of the one-wayness of h. Furthermore, with every session a new nonce N is generated guaranteeing the freshness of messages.

  Modification attacks, where the attacker modifies parts of the second message in Figure 2 will have no effect as all the elements in the message are linked together and one part cannot be changed without making the database ignore the authentication attempt.

  One may argue however, that an attacker can block message three in Figure 2 from reaching the tag. If this is possible the tag and the database will agree on different keys as the database will accept the second message as authentic and renew the secret key from ID_i to ID_{i+1}, while the tag will keep the old value ID_i. Thus both will be in a desynchronized state, blocking any further tag functionality. Notice, however, the difference of this from the desynchronization scenario presented in Section 3.3. There, *any* read attempt made by the attacker will result in a desynchronized state. Here, the attacker must *interfere* in a valid read attempt and block the response from the database.

  We don't consider this issue of active denial-of-service attack as important since there are more simple and effective physical ways of disabling tags. For example, an adversary may use an electromagnetic weapon to disable a tag instead of resorting to highly sophisticated means of allowing the second message to be read by a valid reader and blocking the response. Furthermore, as the first request is by a valid reader, an attacker would run the risk of being discovered since the attack would have to take place in a monitored environment.

  For the same reason we assume a restriction on the ability of an attacker to perform *man-in-the-middle* attacks. Consider for example the scenario where an attacker would like to checkout an expensive item for the price of a cheap one. One way to do this is as follows: upon approaching the point of sale, the attacker first disables the tag corresponding to the expensive item (say by breaking it) and forwards the request $< Request, N_R >$ made by the reader to the tag corresponding to the cheap item. This tag would respond with the message $<\ h(ID_i), N_T, h_{ID_i}(N_T, N_R)\ >$ which the attacker could forward to the first reader. Since the database receives a valid response, it would authenticate the message thus treating the expensive item as the cheap one. We believe such attacks are inherent to RFID systems for the simple reason that the following principle in correct protocol design[15] is violated: "*If the name of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message*." In our case anybody can initiate a conversation and tag identifiers are explicitly left out so to protect user privacy.

  We must emphasize that our assumptions do not characterize the strongest possible type of an adversary. However, our goal is to achieve good, practical security by defending against a number of attacks that can applied to the constrained environment the tags operate. For a similar assumption on the capabilities of adversaries see also [6].

- *Attack on user privacy*. Our protocol makes sure that no code or identity is released by a tag participating in the protocol. Hence eavesdropping is not an issue as long as the hash function is selected in such a way that no usable information upon its pre-image is revealed. Therefore, user privacy is guaranteed.

- *Attack on location privacy*. Our scheme ensures that the secret identifier of the tag will change if the tag interacts with a valid reader. Hence location privacy is guaranteed as no static, traceable IDs exist any more. Notice, however, that between valid sessions the tag ID remains the same. This is required since our focus it to maintain *scalability*. Any attempt to hide $h(ID_i)$ will incur a cost in searching the database, thus making the scheme not scalable as in [3, 4, 7].

- *Attack against the key*. An attacker can record a valid transaction and perform an *offline* brute force attack on $h(ID_i)$ in an attempt to discover ID_i. If successful, she can use the recorded nonces to generate subsequent IDs and trace the tag. However, this type of attack does not work if hash function and the size of the key is selected carefully [16]. Furthermore, due to the forward privacy of our scheme, past transactions remain secure.

  Additionally, even if an attacker discovers the key, this information will be of little value to her due to the following simple fact: she can interact with targeted RFID tags only a limited number of *successive* times as it would be very dangerous to harvest information in a monitored environment. Therefore, if at some point misses the exchanged nonces she will be unable to compute ID_{i+1} even if she has at her possession ID_j, for $j < i$.

- *Attack against implementation*. One must be careful how to implement the various parts of the protocol. Consider for example the second message $< h(ID_i), N_T, h_{ID_i}(N_T, N_R) >$ forwarded to the database and its response $< h_{ID_{i+1}}(N_T, N_R) >$. If the next ID is produced in a careless way, say by the transformation $ID_{i+1} = h(ID_i \oplus N_R)$ then an attacker in an attempt to spoof the reader to the tag could choose $N_R = 0$ resulting in $ID_{i+1} = h(ID_i)$. But $h(ID_i)$ is already contained in the second message thus making the value of the next secret available to the attacker.

  Another bad case is when the next ID can be equal to the current one. Then, since $ID_{i+1} = ID_i$, an attacker could take the third part of the second message, $h_{ID_i}(N_T, N_R)$, and present it as the response from the database. The tag would accept the answer as authentic resulting in desynchronization from the database and destroying further tag functionality.

  Finally, the following attack can be applied when nonces are generated in a predictable manner. In a successful run, the database responds with the message $< h_{ID_{i+1}}(N_T, N_R) >$, signed with the new ID, ID_{i+1}. If in the next protocol initiation the reader creates the nonce N_T, an attacker can spoof the tag to the reader by presenting the nonce N_R. Observe then how the last part of the message $< h(ID_{i+1}), N_R, h_{ID_{i+1}}(N_R, N_T) >$ becomes identical to the previous response by the database.

- *Disassembling the tag*. RFID tags are not tamper-resistant devices so an adversary may disassemble the tag and retrieve cryptographic key material. Although such attack is inherent to all RFID systems, our protocol makes sure that past messages remain secure and the tag cannot be traced.

5 Conclusions

In this work we presented a protocol that provides for secure and authenticated tag-reader transactions. When designing a secure RFID protocol, one must always consider protection against standard attacks like impersonation, replay or cloning. Previous works offered protection against some of these attacks but not all. Through a simple challenge-response protocol we show how to provide authentication and eliminate tag cloning.

However, one of the major security problems, inherent to the use of RFID systems, is violation of user privacy. This can happen when the tag exposes its contents or transmits universal identifiers. Our scheme enforces privacy as it prevents information to be read by unauthorized third parties. This is due to the fact that no single, fixed ID is used throughout the tag's life as tag IDs get refreshed periodically. In addition to that our protocol offers a high degree of location privacy and is resistant to many forms of attacks that we described in the previous section. Furthermore, even in the case where the tag reveals its secrets through physical compromise, forward privacy is guaranteed: past transactions will remain secure as tag secrets change through the application of irreversible computations.

Finally, another characteristic of our scheme is its efficiency: tag identification by a valid reader does not require exhaustive search in the back-end database. Furthermore, only two single messages need to be exchanged, the communications channel need not be reliable and the reader (or any other third party) need not be trusted.

References

[1] Wal-Mart Details RFID Requirement, November 6, 2003. Article appears in `http://www.rfidjournal.com/article/articleview/642/1/1/`

[2] U.S. Military to Issue RFID Mandate, September 15, 2003. Article appears in

`http://www.rfidjournal.com/article/articleview/576/1/1/`

[3] S. Weis, S. Sarma, R. Rivest and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," in *1st Intern. Conference on Security in Pervasive Computing (SPC)*, 2003.

[4] M. Ohkubo, K. Suzuki and S. Kinoshita, "Cryptographic Approach to Privacy-friendly Tags," in *RFID Privacy Workshop*, MIT, 2003.

[5] G. Avoine and P. Oechslin, "A Scalable and Provably Secure Hash Based RFID Protocol," in *The 2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec)*, 2005.

[6] A. Juels, "Minimalist Cryptography for RFID Tags," In C. Blundo, ed., *Security of Communication Networks (SCN)*, 2004.

[7] István Vajda and Levente Buttyán, "Lightweight authentication protocols for low-cost RFID tags," *2nd Workshop on Security in Ubiquitous Computing*, 2003.

[8] Dirk Henrici and Paul Müller, "Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers," *Workshop on Pervasive Computing and Communications Security*, 2004.

[9] Gildas Avoine and Philippe Oechslin, "RFID Traceability: A Multilayer Problem," *Financial Cryptography*, 2005

[10] David Molnar and David Wagner, "Privacy and Security in Library RFID Issues, Practices, and Architectures," *Conference on Computer and Communication Security*, 2004.

[11] Martin Feldhofer, Sandra Dominikus and Johannes Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm," *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2004.

[12] Alfred Menezes, Paul van Oorshot and Scott Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[13] S. E. Sarma, S. A. Weis and D. W. Engels, "RFID systems, security and privacy implications," T*echnical report MIT-AUTOID-WH-014*, AutoID center, MIT, 2002

[14] M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology – Crypto 96 Proceedings, Lecture Notes in Computer Science* Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[15] Martin Abadi and Roger Needham, "Prudent Engineering Practice for Cryptographic Protocols," *IEEE Transactions on Software Engineering*, Vol. 22, No. 1, 1996.

[16] Arjen Lenstra and Eric Verheul, "Selecting Cryptographic Key Sizes," *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography*, 2000.