RFIDDOT: RFID Delegation and Ownership Transfer made simple

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

ABSTRACT

In this work we introduce RFIDDOT, a protocol for secure access, delegation and ownership transfer of tags along with a model for formally defining privacy in such an environment. As current RFID tags emit constant identifiers that may help in identifying user habits and tracking of people, RFIDDOT allows a user to securely own tagged products. Once a person becomes the owner of such an item, no one can have access to the tag nor find any information about it. Thus user privacy is guaranteed. Additionally, the protocol is secure against such attacks as tag cloning, tag/reader spoofing, eavesdropping, desynchronization and so on. Furthermore, since we don't expect a tagged item to stay with same owner forever, we provide the means to achieve ownership transfer and release without compromising the privacy of future or past owners. And in the unlikely case where user privacy is compromised, it can be restored in a simple and intuitive manner. Thus RFIDDOT achieves a very strong notion of security that is necessary in RFID ownership transfer: forward and backward privacy.

Categories and Subject Descriptors

K.6 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Algorithms, Security

Keywords

RFID Security and Privacy, Ownership Transfer, Delegation, Forward and Backward Security

1. INTRODUCTION

About fifteen years ago Mark Weiser developed a vision in which the increasing availability of computing power would be accompanied by decreasing visibility [1]. Radio Frequency Identification (RFID) is a technology that allows the interaction with physical objects and everyday items, enabling new forms of ubiquitous communication between people and things as well as between things themselves.

Despite this increase in convenience, one must wonder, however, whether this pervasive use of RFID technology will open up the possibility for jeopardizing user privacy. Currently, RFID tags respond to any reader request within range. Consequently, anyone carrying a tagged item effectively broadcasts a fixed identifier that may help in violating his or her own privacy.

Our contribution. In this work we focus on secure delegation and ownership transfer of RFID tags. While there exist many works in the literature that focus on securing tag transactions, only a few consider the concept of ownership transfer. In this work we present RFIDDOT, a protocol suite that allows a user to acquire, access, transfer and release tags without undermining the privacy of current or future owners. We also propose a model and privacy definitions strong enough to capture the requirements of secure ownership transfer. We show that our protocol is secure against attacks such as violation of privacy, tag cloning, tag/reader spoofing, desynchronization and so on. Furthermore, RFIDDOT ensures forward and backward privacy which is a prerequisite in secure RFID ownership transfer.

2. DESIGN GOALS

RFIDDOT was designed with the following security requirements in mind:

Confidentiality and Privacy: No information should leak from the tag (tag's secret key or identity) that can help in identifying tag contents or owner. Current RFID tags aim to be cheap, so they emit constant identifiers that may reveal personal and sensitive information. Even if tag contents are encrypted, no fixed identifiers should be emitted that could help identify a person bearing a tag (*location privacy*). In general, no information should be deduced from tag-to-reader communications.

Authentication: Authentication between tags and readers should be enforced in order to avoid such attacks as tracking or tag cloning. In a tag-cloning attack, an attacker may install a replacement tag that emits an identifier similar to the original one. This would fool the system into believing the product is still on the shelf, or alternatively, an expensive item could be purchased for the price of a cheap one. These types of attacks can have serious consequences especially

SecureComm 2008 , September 22 - 25, 2008, Istanbul, Turkey

when RFID tags are used for access control. Examples of such attacks have been demonstrated in various works ([2, 3]).

Delegating authority: To guarantee the previous two requirements, tags must be protected and reveal information only to legitimate readers. At times, however, there is need to delegate the authority to query a tag. This can be the case, for example, when a user buys an item that is going to be used in a smart home environment. The sensors in a such an environment should be able to query the tag and find state information about the tagged item. However, this cannot be done unless these readers are equipped with the secrets carried by the tag in order to decrypt tag responses.

Ownership transfer and tag release: Very similar to the above, at times the item will be passed to a new owner or the tag must be brought to its original state such as when the item must be returned for service. In the first case, the privacy of the new (as well as the past) owner should be guaranteed, while in the second, the possibility of tag cloning or spoofing should be eliminated.

Forward and backward security: RFID tags are inexpensive devices that offer no tamper resistance, hence they suffer from physical attacks that attempt to expose their secret keys. *Forward security* is the security notion which guarantees that even if a tag is compromised now, all *past* transactions remain secure. Thus, an attacker upon compromising a tag will not be able to link this tag with past actions performed on the tag. *Backward security* means that knowledge of the tag's secret now, does not help in identifying the tag's transactions in the *future*.

Since in this work we will be dealing with ownership transfer of tags, we must guarantee that current owners (or adversaries) will not be able to identify transactions and compromise the privacy of past or future ones.

3. THREAT MODEL AND ASSUMPTIONS

A typical RFID architecture consists of a tag, a reader and a back-end database. Upon a scan request by a reader, the tag responds with a unique ID that is transferred to a back-end infrastructure for further processing. The tag is identified and particular info about the tagged product can be retrieved by means of accessing a system database containing all possible tag identifiers.

Communication between tags and readers is wireless and is therefore subject to eavesdropping. Thus most protocols aim to secure this part either by securing tag-to-reader communications or by making tag responses indistinguishable from random data [5]. On the other hand reader-to-database communications can be assumed to be secure since both reader and back-end systems are more powerful devices and can handle the overhead introduced by encryption.

In general, we assume that the tag shares some secret K with the back-end database. This key will be used to secure the identity transmitted by the tag and enhance user privacy against clandestine reading and scanning. One point that has to be clarified, however, and which distinguishes this work from others, is that all past solutions protect users against unauthorized scanning by *third* parties who want to find information about a user's tags. This protection is basically achieved by refreshing tag identifiers and making tag responses look like random data. But there is still potential for violation of privacy. Recall, that in all these solutions the back-end database has knowledge of *all* the secrets associated with the company's tagged products. While this company may not be able to track the tags issued by other companies, it can still track the movements of tags issued by it. Furthermore, a *coalition* of such companies may create a federation network in order to exchange information about a user's movements, habits and profile.

So the concept of *delegation* and *ownership transfer* seems like a necessary solution to the problem of illicit tracking and profiling. Once an item is purchased, the user should be able to hide information about the tag, even by the original owners of this item. This can be achieved by first acquiring all secrets associated with a particular tag and then by refreshing these secrets so that past actions cannot longer be connected to current activities on the tag.

However, our threat model extends beyond the back-end database which possesses the secret of every available tag. It also includes (possibly malicious) owners who have acquired tags and now they transfer them to new users. These malicious users can lie about the identities of the tags, can try to clone them or even track the activities of the new owners. Thus ownership transfer must be done in a way to guarantee protection against these types of attacks.

In the protocol we describe in the next section, we are assuming that tags are capable of performing simple cryptographic operations, like evaluating a pseudo-random or hash function on a given input as in many past works ([4, 5]). This is further justified by recent developments ([6, 7, 8]) which have demonstrated the capability of AES (Advanced Encryption Standard) computations on low-cost tags. The following notation will be used throughout the paper.

Notation	Meaning
Owner_i	i-th owner of the tag.
K	A secret key possessed by tag (and shared with the reader/database of current owner).
M_1, M_2	Concatenation of messages M_1 and M_2.
$F_K(M)$	Application of pseudo-random function F on message M and key K.
N_R, N_T	Nonce identifiers (random numbers used once) sent by reader and tag, respectively.

4. RFIDDOT: A PROTOCOL SUITE FOR RFID DELEGATION AND OWNERSHIP TRANSFER

The basic protocol[1] for tag-reader interaction is shown in Figure 1. In this protocol we are assuming that the back-end system has a table entry associated with each tag containing such information as secret key, tag identifier, product info, etc.

The database, upon receiving $F_K(N_T, N_R)$, uses this value to search and recover the identity of the tag. Notice that this process requires exhaustive search on behalf of the back-end system since the database has no way of disambiguating the tag other than trying all possible keys K and testing which key matches the "signature" $F_K(N_T, N_R)$. This is something that can be fixed by enhancing the protocol using the techniques described in [10] and [11] (basically arranging

[1]This is not a novel protocol. Similar tag-reader protocols have been suggested in a number of places in the literature [4, 5]. However, we use this protocol as the basis for providing strong RFID privacy. Furthermore, we show how to extend this protocol to achieve ownership transfer.

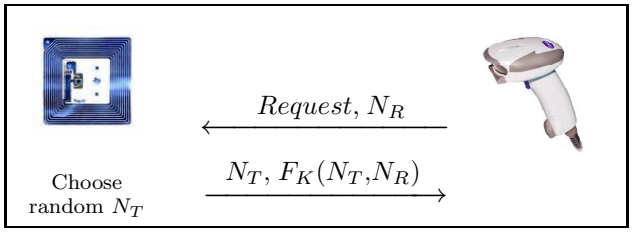

Figure 1: Basic protocol for tag-reader interaction.

the tags in a tree hierarchy). Notice, however, that the increase in efficiency may result in some loss in privacy as was demonstrated in [11] and [12]. So, at this point we will consider only the basic version since the scalability issue is not really a concern here.

This protocol guarantees privacy (see also a formal proof for a similar scheme in [18]) since no fixed identifiers are emitted (recall that the tag always includes a fresh identifier in the evaluation of F) and no information leaks because of the one-wayness of the function F. [2]

4.1 Delegating authority

This section refers to situation where a user has acquired a set of tagged products and wants to use them in a private environment like a person's home.

Here we emphasize in the notion of a smart home or "ubiquitous computing" environment in which tiny computers embedded in everyday objects would respond to the presence and needs of humans without being actually manipulated ([1, 13]). In an environment equipped with this kind of "calm technology", readers placed in strategic points would detect the data embedded in everyday items (clothes, shoes, drugs, books, food containers, etc.) and relay this data to a central computer which could act based on that information (eg. turn on light and climate control, play our favorite music, and so on). However, given that tags responses are encrypted (recall the protocol of Figure 1), these sensing devices/readers should be made capable of interpreting tag responses. In order for this to happen, knowledge of the secret key stored in the tags is required.

Information about tag secrets can be retrieved, for example, at a point of sale. The only requirement is that a person buying a tagged item must carry with her a mobile reader. Such a reader could be embedded in a mobile phone or a PDA [14]. This assumption is also justified by the establishment of the Near Field Communications (NFC) forum [15] that aims in integrating mobile phones with existing passive RFID products. The NFC standard is compatible with both ISO 14443 and ISO 15693 and allows devices to operate as either a reader or a tag, thus allowing both transmission and reception of data.

Using the mobile phone, the user can connect to a checkout server through a secure channel and receive all relevant information about the tag. Alternatively, the secret key K can be printed in a receipt in a 2D barcode and have the mobile reader scan the receipt. When this procedure is complete, the user's device will be able to communicate and interpret tag responses without the need of relaying this information to a back-end database.

[2]Notice that the protocol of Figure 1 does not entirely defeat *person-in-the-middle* attacks. For example an attacker can forward the reader request and then replay the response of a distant tag [3]. Unfortunately, these types of attacks are inherent to RFID systems since tag identities are explicitly left out and tag responses cannot be authenticated without further involvement of the reader. One can try distance bounding techniques like those in [9], at the risk, however, of loosing compliance with the RFID paradigm. Here, we will opt for simplicity since this type of attack does not really constitute a problem in our case, at the understanding, however, that more advanced protocols can be plugged in and used with the RFIDDOT framework.

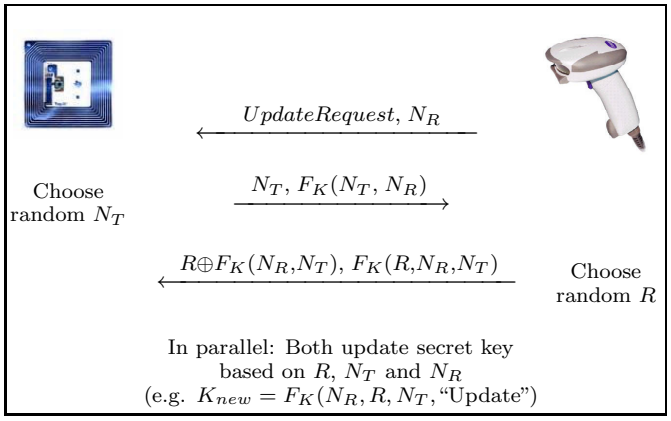

Figure 2: Ownership transfer protocol.

4.2 Ownership Transfer

The ideal consequence of the previous operation should be that the user takes complete control of the tag and that no one should be able to trace the tag any more. Recall, however, that the back-end database still possesses the secret of the tag. Thus, a network of readers connected to this central location can still identify the tag and track the user. Hence what is needed is the ability by the current owner to *update* the secret information stored in the tag. This is achieved using the protocol shown in Figure 2.

The first two steps are similar to the one shown in Figure 1. The only difference is that the mobile or home reader of the user signifies this message exchange to be an *update-request* so that the tag in the end of the protocol updates its secret key. Towards that purpose, the reader first computes another random number R and transmits the third message shown in the figure. Upon receiving this message, the tag first computes $F_K(N_R, N_T)$ and then extracts the random number R. The second part, $F_K(R, N_R, N_T)$, of the message essentially acts as a Message Authentication Code (MAC). This is used to verify that R was retrieved successfully by the tag, since if an adversary modifies the first part, the tag would end up with a wrong R and eventually with a wrong key. The MAC also offers protection against malicious users issuing rewrite commands since in both cases the MAC will not verify and the tag will abort the update operation.

If all tests pass successfully, both the reader and the tag update the current key to a new key K_{new}. This new key will be used in subsequent transactions between the tag and the reader. And since K_{new} is generated by an *unpredictable* R and *fresh* random data like N_T and N_R, the back-end database cannot any more track or identify the tag.

We open a parenthesis here to discuss the generation of K_{new} a bit more. It is important that the new key is generated by the application of an *one-way* function on R, N_T

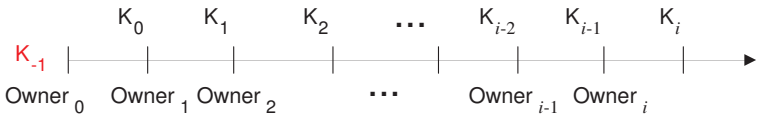

Figure 3: Sequence of owners and secrets.

and N_R. An alternative would be that the new key is *planted* directly by the reader. However, this last option opens the possibility of tag spoofing attacks by malicious owners that may attempt to make this tag simulate the responses of another. As the one-wayness of F guarantees that, given N_T and N_R, no R can be found, which maps to a specific key, tag cloning is eliminated. We will return to this point in Section 6.

We should also stress that the key update operation must take place in an environment where *tag-to-reader responses* (usually called the *back channel*) *are protected from eavesdropping*. This is a very weak assumption since this channel is much harder to eavesdrop than the forward channel (from reader to tag) [16, 17]. Thus, after buying a tagged product, the user can refresh the secret in the privacy of her home. Furthermore, this update operation can be run as *many times* as necessary. If, during these updates, an adversary who knows all past secrets, ever misses one, then all future interactions are secured as the last shared secret will be known only between the user's reader and the tag. In the Appendix, we show how to model forward and backward privacy which is a prerequisite in secure ownership transfer.

5. SECURITY ANALYSIS

In this section we analyze the security properties of the protocol. Generalizing the discussion so that it can be applied to more than one owners, we consider a timeline, where Owner_i denotes the i-th owner of the tag and K_i the key that is handed over from Owner_i to Owner_{i+1} (Figure 3). It should be clear that Owner_0 denotes the back-end database and K_0 the key that is handed over to the first owner. K_{-1} is the *original* key of the tag, one that is *never* revealed and only used by the back-end system itself (the need for K_{-1} will become clear in Section 6 when we talk about releasing tags).

Subsequent owners correspond to users who came in possession of the tag after this has been transfered by the previous owner. It should also be clear that when Owner_{i-1} wishes to transfer the tagged item to Owner_i (say by selling it), Owner_{i-1} should also hand over a valid tag-access key K_{i-1} using the methods described in Section 4.1. Notice, that this key does not have to (and *should not*) be the key that Owner_{i-1} used to access the tag but only one that is given to Owner_i during ownership transfer (we will say more on this when analyzing forward/backward security). Finally, we emphasize again that when Owner_i wishes to update the key K_{i-1} that was handed over by Owner_{i-1} to a new key, this should happen in an environment protected from eavesdropping.

5.1 Privacy enforcement

The protocol enforces privacy since no fixed identifiers are emitted (recall the continuous use of unpredictable numbers) and no information ever leaks because of the one-wayness of F. Furthermore, no one can recover R from the message $R \oplus F_{K_{i-1}}(N_R, N_T)$ without knowing K_{i-1}. This is because the generic construct $\langle r, F_K(r) \oplus m \rangle$, for a random r, is known to be semantically secure and leak no information. Combining this with the last part of the message (the MAC), we obtain security against modification attempts. Notice that Owner_{i-1} also cannot recover R since she has no access to the update messages that take place in the privacy of Owner_i's home. Thus, while Owner_{i-1} knows K_{i-1}, she does not have access to the message containing R.

5.2 Attacks on tag and reader authentication

The purpose of an adversary impersonating the tag would be to make the back-end database (or any owner for that matter) accept a fake tag as valid. Notice that tag responses (second message in Figure 2) also depend on the nonce N_R supplied by the reader. Hence a fake tag, without knowledge of the current secret key has no way of generating a valid response other than sending something at random. But then the probability of fooling the reader is at most $N/2^k$, where N is the number of items held by the current owner and k is the security parameter of the system (i.e. key lengths). Thus the protocol achieves tag authentication with a cheating probability at most $N/2^k$, which is negligible.

Consider now an adversary who tries to impersonate a legitimate reader (current owner). This could lead to tracking or making the tag accept the adversary as its new owner. As before, an adversary without knowledge of the secret key has no better strategy than sending random data in the third message of Figure 2. However, without knowledge of the key, the MAC supplied by the adversary would not verify and the tag would not update its state. Hence the probability of such response being accepted by the tag is again negligible.

5.3 Tag cloning

Tag cloning would be possible if a malicious owner, say Owner_i, could plant a secret key of her choice. This would allow her to make the key equal to the key of another tag T. Thus the current tag would be indistinguishable from T and a back-end server or a new owner could be fooled to accept a fake tag as authentic, say for the purpose of upgraded service or better price during tagged product transfer.

The approach we follow here is to let the tag generate the new key using the one-way function F. The one-wayness of F guarantees that given a target key K and a nonce N_T supplied by the tag, a malicious owner *cannot* come up with N_R and R that generate this given key. Thus tag cloning is practically impossible.

5.4 Desynchronization and Denial of Service (DoS)

Our protocol has strong resistance against DoS attacks on the last protocol message sent by the reader in Figure 2.

Consider what happens if this message never reaches the tag, say because of transmission problems or even worse because of an adversary blocking this message. Could this result in tag-reader desynchronization (i.e. the reader updates the key to K_{new} while the tag aborts the update operation)?

In the unlikely case of this event, there is a simple solution to the problem. If the update process is unsuccessful, the reader can test this by querying the tag using the basic protocol (Figure 1) and the newly established key K_{new}. If the tag cannot be recognized, this is an indication that key update failed and that the tag still has the old key. So, the reader only has to issue another update using the old key until the update is successful. Thus the only requirement is that the reader remembers the old key until it verifies that the new key was installed in the tag. Then it deletes the old key from its memory.

5.5 Forward and backward security

Forward security suggests that even if a tag secret has been compromised (or given away) at time t, all tag transactions that took place at time $t' < t$ still remain secure. Without this requirement a future owner could still track the movements of past ones.

So, consider the existing owner Owner_i of the tag. As we said in the beginning of this section this owner has acquired key K_{i-1} from the previous owner in order to update the tag's key. Hence this owner can still identify the past transactions of Owner_{i-1}, provided that the last one used K_{i-1} for tag access. The solution to this problem is again simple. Before Owner_{i-1} handovers her secret to Owner_i, she updates the key one more time (this can be done since the update protocol can be run as *many* times as desired). Hence the key that is given to Owner_i is not really connected to the key that Owner_{i-1} used to access the tag and therefore traceability is not possible. Similarly, in order to protect the back-end database (Owner_0), the key K_0 that is handed over to the first owner does not have to be the original key (say K_{-1}) that is used by the retail store but it can be the result of an update operation on K_{-1}.

Backward security suggests that knowledge of a tag's state at time t cannot help in identifying the tag's transactions at time $t' > t$. This is true under the reasonable assumption that an adversary compromising the current tag key of Owner_i cannot eavesdrop on *all* future interactions of the tag. Thus, if the same owner ever updates the tag key again or hands over the tag to a new owner who securely updates the key, then future interactions are secured.

In summary, if location privacy is broken at some point during the lifetime of the tag, *it can be restored by just one secure key update.*

6. RELEASE OF A TAG

We conclude the description of the RFIDDOT protocol suite by discussing the case where the current owner wants to bring a tag to its *original* state. This may be required when the tagged item must be returned for service or when another compatible product is sought for. The manufacturer or the retail store (Owner_0) should be able to access the tag freely.

One way to solve this would be to let the current owner hand over to Owner_0 the key that (s)he is using to access the tag. However, this does not prevent the current owner from cheating; the current owner may submit the wrong key or the key of another tag to Owner_0. Then the last one would not be able to tell if the tag is authentic.

As we have already excluded the possibility of owners implanting/handing over keys directly to eliminate tag cloning attacks, the solution is to let the tag *itself* restore its original secret when instructed by the current owner. The protocol for doing this is shown in Figure 4.

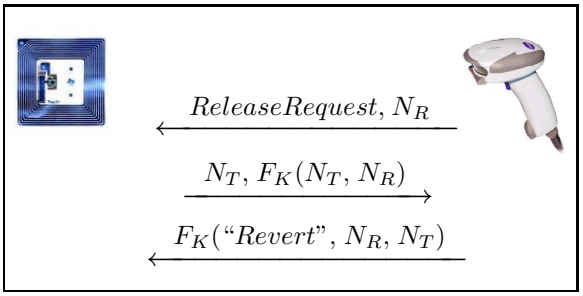

Figure 4: Restoring the original key of the tag.

In the protocol, K denotes the key of the current owner. The first two steps of the protocol follow the format of the previous ones although this operation could be made simpler. In the third message, the current owner (and *only* the current owner) instructs the tag to revert back to a key K_{-1} which is always stored in its memory, and constitutes the original key that can be used by Owner_0 to disambiguate the tag (thus tags have two keys; a permanent one that can be restored *only* by current owner during tag release, and a transient one that can be used to access the tag). When, finally, the user reclaims back the tagged product, she will also be handed a key K_0' to access the tag. Then she can update the key in the privacy of her home using the protocol of the previous section.

This scheme inherits the security properties of the previous protocols, and the secrecy of K_{-1} guarantees that an attacker will not be able to *simulate* tag responses and fool Owner_0 as she could do if the original tag secret was K_0 or any other key that could be handed over to Owner_0. Nor, Owner_0 can use K_{-1} to track the tag. Once the current owner is given a new access key, she can update this key, thus ensuring backward privacy.[3]

7. RELATED WORK

There are many works in the literature that deal with the problem of RFID privacy (see [4] for a recent survey) but only a few that consider RFID delegation and ownership transfer.

In [10], the authors propose a delegation scheme based on a tree of secrets. This scheme carries some of the inconveniences of tree based schemes ([11, 12]) since compromise of one tag may reveal information about others. Furthermore, delegation is limited to a number of times that is specified by the tree and the tags must be involved in more ex-

[3]Notice that this protocol is secure according to the model presented in Appendix A. When the adversary is given oracle access to the tag, she is not given access to the key K_{-1} that could be used to disambiguate the challenge tag. The adversary is only given access to updated keys that are not related to K_{-1}. In real life, however, even if an adversary is given access or finds about K_{-1} (e.g. Owner_0 distributes this key), it takes only one private update operation to restore privacy since K_{-1} is never used in ordinary tag transactions!

pensive computations such as deriving secrets and encoding pseudonyms. Additionally, the back-end system still knows the secrets of the tags.

In [22], the authors achieve ownership transfer by letting the owner broadcast a new symmetric key K' to the back-end database which then instructs the tag to refresh its secret ID to $E_{K'}(ID)$. However, we feel that this approach suffers from desynchronization problems since the database cannot be sure that the tag updated its secret. Additionally, the scheme requires an always *online* database for tag access and ownership transfer.

In [23], a protocol for RFID authentication is proposed which also enables ownership transfer as a byproduct. However, this protocol requires a large amount of tag computations and the maintenance of a number of hash chains. Furthermore, this approach is prone to desynchronization problems which may lead to traceability due to the finite length of the key chains. Additionally, the protocol does not support tag release.

The RFID Enhancer Proxy or REP [24] is a device that assumes the identities of tags and simulates them in the presence of reading devices. However, this approach suffers from a few shortcomings such as corruption of tag data, tag-to-REP desynchronization and difficulty in tag release that are attributed to the fact that tag identities need to be partially generated by the tag and match portions of its true ID.

Finally, the RFID Guardian [25], is a device that can enforce various policies when interacting with readers. One drawback of this scheme is that the Guardian must always be alert in protecting tag responses from unauthorized read attempts. Furthermore, the authors do not consider such issues as tag acquisition and ownership transfer.

8. CONCLUSIONS

In this work we have presented RFIDDOT, a simple protocol that allows users to securely own, transfer or release tags. The protocol ensures tag-reader authentication and guarantees protection against such attacks as invasion of privacy, tracking, tag or reader impersonation, tag cloning and so on. We have also designed the protocol with both forward and backward security in mind and we have proposed privacy definitions strong enough to capture the requirements that are necessary to make ownership transfer secure. Using RFIDDOT, ownership transfer and release is achieved by a simple update operation that ensures the privacy of past and future owners.

9. ACKNOWLEDGMENTS

The author would like to thank the reviewers for their many helpful comments.

10. REFERENCES

[1] Mark Weiser, "The computer of the 21st century," in *Scientific American*, Jan, 265(3), pp. 94–104, 1992.

[2] Stephen C. Bono, Matthew Green, Adam Stubblefield, Ari Juels, Aviel D. Rubin, Michael Szydlo. "Security Analysis of a Cryptographically-Enabled RFID Device." In *14th USENIX Security Symposium*, 2005

[3] Ziv Kfir and Avishai Wool. "Picking Virtual Pockets using Relay Attacks on Contactless Smartcard Systems," In *1st Intern. Conf. on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, 2005.

[4] Ari Juels, "RFID security and privacy: A research survey," IEEE Journal on Selected Areas in Communication, 2006.

[5] Tassos Dimitriou, "RFID Security: Attacks and countermeasures," Book Chapter in RFID Security: Techniques, Protocols and System-On-Chip Design, Paris Kitsos, Yan Zhang (eds), Springer Verlag, 2008.

[6] M. Jung, H. Fiedler, and R. Lerch. "8-bit microcontroller system with area efficient AES coprocessor for transponder applications," In *Ecrypt Workshop on RFID and Lightweight Crypto*, 2005.

[7] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. "AES Implementation on a grain of sand," In *Information Security*, IEE proceedings, 152(1):13-20, October 2005.

[8] P. Kaps and B. Sunar, "Energy comparison of AES and SHA-1 for ubiquitous computing," In *EUCŠ06*.

[9] Gerhard Hancke and Markus Kuhn (2005) "An RFID distance bounding protocol," In *1st Intern. Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, 2005.

[10] David Molnar, Andrea Soppera and David Wagner, "A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags," in *SAC*, 2005.

[11] Tassos Dimitriou, "A Secure and Efficient RFID Protocol that could make Big Brother (partially) Obsolete," in *4th IEEE Intern. Conference on Pervasive Computer and Communications (PerCom)*, 2006.

[12] Karsten Nohl, David Evans (2006) "Quantifying Information Leakage in Tree-Based Hash Protocols," in *8th International Conference on Information and Communications Security (ICICS)*, USA.

[13] Roy Want. "RFID: A Key to Automating Everything," In *Scientific American*, Jan, 290(1), pp. 56–65, 2004 1.

[14] "Nokia unveils RFID phone reader," IN *RFID Journal*, 17 March 2004. http://www.rfidjournal.com/article/view/834.

[15] http://www.nfc-forum.org

[16] Kenneth Fishkin and Sumit Roy, "Enhancing RFID privacy through antenna energy analysis," In *MIT RFID Privacy Workshop*, 2003.

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

[18] A. Juels and S. Weis. "Defining Strong Privacy for RFID," in *Proceedings of the 5th IEEE International Conference on Pervasive Computing and Communications Workshops*, 2007. Also in Cryptology ePrint archive, www.iacr.org

[19] Gildas Avoine. "Adversary Model for Radio Frequency Identification," in Cryptology ePrint archive, www.iacr.org

[20] Ivan Damgård and Michael Østergaard Pedersen. "RFID Security: Tradeoffs between Security and Efficiency," in Cryptology ePrint archive, www.iacr.org, July 2006.

[21] Mihir Bellare and Bennet Yee. "Forward Security in Private-Key Cryptography", in *Topics in Cryptology - CT-RSA 03*, Lecture Notes in Computer Science Vol. 2612, M. Joye ed., Springer-Verlag, 2003.

[22] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi. "An Efficient and Secure RFID Security Method with Ownership Transfer," in *IEEE Intern. Conf. on Computational Intelligence and Security (CIS'06)*, 2006.

[23] C. H. Lim and T. Kwon. "Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer," In *8th Intern. Conference on Information and Communications Security (ICICS '06)*, December 2006.

[24] A. Juels, P. Syverson, and D. Bailey, "High-Power Proxies for Enhancing RFID Privacy and Utility," In *Center for High Assurance Computer Systems - CHACS*, 2005.

[25] M. Rieback, B. Crispo, and A. Tanenbaum, "RFID Guardian: A Battery-powered Mobile Device for RFID Privacy Management," in *Australasian Conference on information Security and Privacy - ACISP 2005*, vol. 3574 of LNCS, pp. 184-194, July 2005.

APPENDIX

A. A MODEL FOR FORWARD AND BACKWARD PRIVACY WITH RESPECT TO OWNERSHIP TRANSFER OF TAGS

In this section we formalize the notion of forward and backward security. We follow the model proposed in [18] that is reminiscent of the classic indistinguishability games (a closely related model is the one proposed in [19]). The model allows for a set of tag and reader functionalities such as SetKey, TagInit and ReaderInit. Basically, SetKey models the ability of adversaries to compromise tags as well as manufacture/clone arbitrary tags. TagInit and ReaderInit messages associate a tag and a reader with a particular session id.

In accordance with the ownership transfer protocol presented in Section 4.2, we extend the model by including a tag functionality, which we call Update, in which the tag after receiving an appropriate message and challenge information from the reader, it *updates* its internal key as a function of the previous key and the challenges associated with a particular session. When the update is performed, the old key is erased from the memory of the tag. Notice that an update can be successful only if the corresponding tag key is also known to the reader. Otherwise, the update will be rejected by the tag.

An adversary $\mathcal{A}$ can issue its own SetKey, TagInit, ReaderInit, Update as well as challenge and response messages. The adversary is parameterized by the number r of the ReaderInit messages, the number of computational steps s and the number of TagInit messages t she sends. Such an adversary is denoted by $\mathcal{A}[r, s, t]$. The model does not explicitly mention the number of challenge-response messages since these are determined by the underlying protocol and the number of TagInit and ReaderInit messages. We follow the same approach with the Update messages since these are also a function of the challenge-response messages.

SetKey messages are not parameterized either. When a tag receives such a message it is considered compromised by the adversary. It is assumed that an adversary can corrupt any of a number of tags except the two that are part of the indistinguishability games. In [20], the number of corrupted tags is treated as a separate parameter since the goal there is to study tradeoffs between security and efficiency in tree-based systems where tags *share* secrets. In our case, however, this is not necessary so we will stick to the simpler model.

A.1 Privacy Games

In what follows we present the two games that are related to forward and backward security. These are similar to classic indistinguishability games in which an adversary tries to obtain partial information from encrypted messages. In our case, the goal of the adversary is to distinguish between two different tags with significant probability over the parameters of the system.

Each experiment will consist of two phases; a training and a challenge phase. In the *training* phase the adversary is allowed to send any number of TagInit, ReaderInit, Update and challenge-response commands within the appropriate time bounds, and corrupt via SetKey messages any set (up to $n-2$) of tags, where n denotes the number of tags in the system.

In the *challenge* phase, the adversary selects two uncorrupted tags as the challenge candidates. One of these tags is then randomly selected and presented to $\mathcal{A}$ via an appropriate oracle. In what follows, we will present two oracles that correspond to our notions of forward and backward privacy in conjunction with ownership transfer. Eventually, the adversary has to distinguish between the two tags in order to be successful. In the opposite case, the system is considered secure for the corresponding security notion.

The system is initially setup by running a probabilistic key generation algorithm $Gen(1^k)$ which produces a set of keys to be assigned to the n tags. These keys may evolve as a result of Update operations. Initially, these keys are not known to $\mathcal{A}$. Setting $\mathcal{S} = (Gen, \mathcal{R}, \{\mathcal{T}_i\})$, we let $\mathbf{Exp}^{FS-OT}_{\mathcal{A},\mathcal{S}}[k, n, r, s, t]$ and $\mathbf{Exp}^{BS-OT}_{\mathcal{A},\mathcal{S}}[k, n, r, s, t]$ denote the privacy experiments for forward and backward privacy with respect to ownership transfer (OT). These experiments are presented in more details in the sections that follow.

A.1.1 Forward Security

In general, *forward security* means that even if a secret key is compromised at the present, all past transactions still remain secure. In our setting, however, *we focus on providing forward security with respect to ownership transfer*. We want to make sure that even if a tag secret is compromised now (or the tag acquires a new owner who is given the current tag secret – recall Section 4.1), all tag transactions that took place with previous owners still remain secure. It should be clear that since the tag does not by itself update its secret with every read request, forward security in the classical sense [21] cannot be achieved since the compromised key will reveal all transactions that occurred with the same key. Thus we divide time in different ownership periods and we want to ensure that forward privacy is guaranteed across *different* periods. This models the requirement that current owners of a tag (or adversaries that compromised a tag now) cannot identify the past owners behaviors (say through readers logs).

We start with the same setup phase as in [18]. Furthermore, we let transactions of a tag be divided into periods $i = 1, 2, ...$, according to the key used in that period. Thus once a key is updated, a new period begins. The new key is obtained from the previous key through an Update operation. Forward security means that privacy of tag data at period j is maintained even if the adversary is in possession of a key at a period $i > j$. To capture this we define the experiment shown in Figure 5. In the experiment, a tag $\mathcal{T}$ operating in period i will be denoted by $\mathcal{T}^i$.

The experiment is similar in flavor to those presented in [21]. First the adversary $\mathcal{A}$ selects two uncorrupted tags and two periods i and j, with $i > j$. These tags are updated successively by the *oracle* until period i is reached. Then one of the two tags is selected at random and presented to $\mathcal{A}$ but with an extra twist: it is the j-th period tag that is given to $\mathcal{A}$ and not one of the i-th period tags. $\mathcal{A}$ is also presented with the i-th period keys of the tags (alternatively is given access to $\mathcal{T}_0^i$, $\mathcal{T}_1^i$). The adversary is considered successful if it guesses correctly which of the two tags $\mathcal{T}^*$ corresponds to.

Intuitively, this experiment captures the notion of forward privacy under tag ownership transfers. If the system is secure then privacy of *past* tag owners is still guaranteed even

Experiment $\mathbf{Exp}^{FS-OT}_{\mathcal{A},\mathcal{S}}[k,n,r,s,t]$:

- *Training phase*

  $\mathcal{A}$ may perform any number of ReaderInit and TagInit calls without exceeding r and t calls respectively, communicate, compute and Update without exceeding s steps overall, and compromise at most $n-2$ tags using SetKey calls.

- *Challenge phase*
  - $\mathcal{A}$ selects two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ that have not been compromised by SetKey messages and specifies two values i and j with $j < i$. $\mathcal{A}$ is then given access to an FS tag oracle.
  - The oracle updates the keys of *both* tags using Update operations until period i is reached. Let b be a random bit and let $\mathcal{T}^*$ be one of $\mathcal{T}_0^j$, $\mathcal{T}_1^j$. $\mathcal{A}$ is then given access to $\mathcal{T}^*$ and the i-th period keys of tags $\mathcal{T}_0$, $\mathcal{T}_1$.
  - $\mathcal{A}$ may perform any number of ReaderInit and TagInit calls without exceeding r and t calls respectively, communicate, compute and Update without exceeding s steps overall, and compromise at most $n-2$ tags using SetKey calls, but *not* $\mathcal{T}^*$.
  - $\mathcal{A}$ outputs a guess bit b'.

  Exp is successful if $b = b'$.

Figure 5: Forward privacy experiment for ownership transfer

if a tag has been compromised or changed hands. In Section 5, we explained how such guarantees can be achieved in practice. Below we give a formal definition of forward privacy along the lines of [18].

Definition 1. *A protocol initiated by $\mathcal{R}$ in an RFID system $\mathcal{S} = (Gen, \mathcal{R}, \{\mathcal{T}_i\})$ with security parameter k is (r,s,t)-Forward Secure with respect to Ownership Transfer (FS-OT) if:*

$$\forall \mathcal{A}[r,s,t] \quad \Pr[\boldsymbol{Exp}^{FS-OT}_{\mathcal{A},\mathcal{S}}[k,n,r,s,t] \text{ succeeds in guessing } b.] \leq \tfrac{1}{2} + 1/poly(k)$$

A.1.2 Backward Security

Backward security in RFID systems aims to protect *future* owners from current ones (or from key compromises that occurred in the past). As before, backward security makes sense only with respect to ownership transfer. This is because an adversary can still trace a tag during the authentication attempts that follow a tag compromise. So, we require that there is a time interval where the adversary *cannot* listen to tag-reader interactions. This is when an update occurs (this requirement is ensured by having the new owner perform an update in a private environment protected from eavesdropping – recall Section 4.2).

Again we divide time in different periods; a new period begins as a result of an Update operation on the current key. Thus, backward security means that privacy of tag data at period i is maintained even if the adversary is in possession of a tag at period j, where $j < i$. All this is modeled by the experiment shown in Figure 6.

Experiment $\mathbf{Exp}^{BS-OT}_{\mathcal{A},\mathcal{S}}[k,n,r,s,t]$:

- *Training phase*

  $\mathcal{A}$ may perform any number of ReaderInit and TagInit calls without exceeding r and t calls respectively, communicate, compute and Update without exceeding s steps overall, and compromise at most $n-2$ tags using SetKey calls.

- *Challenge phase*
  - $\mathcal{A}$ selects two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ and is then given access to a BS tag oracle.
  - The oracle updates the keys of *both* tags using Update operations (say until period i is reached). Let b be a random bit and let $\mathcal{T}^*$ be one of $\mathcal{T}_0^i$, $\mathcal{T}_1^i$. $\mathcal{A}$ is then given access to $\mathcal{T}^*$ and the j-th period keys of tags $\mathcal{T}_0$, $\mathcal{T}_1$.
  - $\mathcal{A}$ may perform any number of ReaderInit and TagInit calls without exceeding r and t calls respectively, communicate, compute and Update without exceeding s steps overall, and compromise at most $n-2$ tags using SetKey calls, but *not* $\mathcal{T}^*$.
  - $\mathcal{A}$ outputs a guess bit b'.

  Exp is successful if $b = b'$.

Figure 6: Backward privacy experiment for ownership transfer

Initially the adversary selects two tags. These tags can be owned by the adversary, which means $\mathcal{A}$ may know their corresponding keys (these tags can be thought as period j tags). The keys of both tags are updated at least once until (say) period i is reached. These updates are performed by the BS-oracle without $\mathcal{A}$ having access to the data exchanged. Then $\mathcal{A}$ is presented with one of the i-th period tags and her goal is to determine which tag was selected by the oracle.

Intuitively, this experiment captures the notion of backward privacy under tag ownership transfers. If the system is secure then privacy of *future* tag owners is still guaranteed even if a tag has been compromised or changed hands in the past. A formal definition of backward privacy follows:

Definition 2. *A protocol initiated by $\mathcal{R}$ in an RFID system $\mathcal{S} = (Gen, \mathcal{R}, \{\mathcal{T}_i\})$ with security parameter k is (r,s,t)-Backward Secure with respect to Ownership Transfer (BS-OT) if:*

$$\forall \mathcal{A}[r,s,t] \quad \Pr[\boldsymbol{Exp}^{BS-OT}_{\mathcal{A},\mathcal{S}}[k,n,r,s,t] \text{ succeeds in guessing } b.] \leq \tfrac{1}{2} + 1/poly(k)$$