Book Title: XXXXXXXXXXXXXXXXXXXXXXXXXXX

Editors

October 5, 2010

Contents

Chapter 1

DoS Attacks on RFID Systems: Privacy vs. Performance

Dang Nguyen Duc
Auto-ID Lab Korea, KAIST, Republic of Korea
Kwangjo Kim
Auto-ID Lab Korea, KAIST, Republic of Korea

ABSTRACT

In this chapter, we discuss the impact of providing tag privacy on the performance of an RFID system, in particular the complexity of identifying the tags being queried at the back-end server. A common technique to provide tag privacy is to use pseudonyms. That is, for each authentication session, a tag uses a temporary and random-looking identifier so that it is infeasible for attackers to relate two authentication sessions. A natural question which should arise here is how the server can identify a tag given that the tag's identity is changing all the time. This problem becomes even more serious when the shared secret key between a tag and the server is updated after every authentication session to provide forward privacy. In the first part of this chapter, we review different techniques to deal with this problem. We then point out that most of the existing techniques lead to vulnerability of the server

against Denial-of-Service (DoS) attacks. We illustrate some of these attacks by describing methods which attackers can use to abuse the server's computational resources in several popular RFID authentication protocols. Finally, we discuss some techniques to address the privacy-vs-performance dilemma so that DoS attacks can be prevented.

INTRODUCTION

RFID is an emerging technology which promises many powerful applications in supply chain management, smart home appliances and ubiquitous computing, *etc.* The key idea is to attach to each and every object a low-cost, wirelessly readable RFID tag (by the so-called RFID readers). Each RFID tag carries a unique string, serving as an object identifier so that this identifier can be used as a pointer to look up detailed information on the corresponding object (at some database server or simply the back-end server where all the detailed information on tagged objects are stored and managed). The identifier that a tag carries is usually called the *Electronic Product Code* (EPC). Unfortunately, this very operation of an RFID tag also causes serious security concerns. These concerns are two-fold.

- *Counterfeiting product*: When RFID is widely deployed, we will come to depend on it to recognize surrounding objects, especially merchandized objects. However, the object identifier stored in an RFID tag can be read by any compatible RFID reader. In addition, wireless communication is inherently insecure against eavesdropping attacks which might help attackers capture the object identifier without querying the tag. Malicious parties can collect legitimate tag identifiers and create tags that emit the same identifiers. We call these tags *cloned tags*. The cloned tags can be placed on counterfeiting products to avoid being detected.

- *Consumer privacy*: The uniqueness and availability of object identifiers raise another side effect for consumers, their privacy. With the vision of RFID tags being everywhere, it will become common for a person to carry with him/her several RFID-tagged objects. Therefore, a malicious hacker equipped with a compatible RFID reader, can identify objects carried by RFID holders as well as trace their movements. For the rest of this

chapter, we will use consumer privacy and tag privacy as two equivalent terms.

To deal with counterfeiting products, one can implement an authentication protocol between the reader and the tag so that fake tags are detected and malicious readers cannot collect useful information from legitimate tags. This can be done easily, even on low-cost hardware like RFID tags. Many RFID authentication protocols (Ohkubo et al. (2004)) which employ just a cryptographic hash function or other lightweight cryptographic primitives have been proposed so far. However, dealing with privacy issues is much more difficult. This is not because privacy is a new problem in cryptography. Rather, it is due to the cost of providing privacy which in some cases could require the back-end server to go through the whole tag database in order to identify a single tag. A common technique for providing privacy is to use pseudonyms for RFID tags instead of their true identifiers. That is, for each authentication session, a tag uses a temporary, random-looking identifier (thus, it is called pseudonym) so that it is infeasible for attackers to relate two authentication sessions, even of the same tag. However, if a tag emits a different identifier whenever it is queried, the back-end server will have difficulty in identifying the tag. This problem becomes even more serious when forward privacy is required. Forward privacy is a security notion which specifies that the privacy of a tag is still partially preserved even if the secret key of the tag is exposed. By partially preserved, we mean the querying sessions which happened before the secret key exposure remain anonymous (or sometimes referred to as unlinkable). In other words, the privacy of the tag is guaranteed up to the point of the secret key exposure. To provide forward privacy, a common practice is to refresh the secret key after every querying session. Unfortunately, this could add additional burdens on the server when identifying and authenticating a tag because the server has to take care of situations where the secret keys shared between the server and tags were inconsistently refreshed (we refer to this problem as *de-synchronization of secret*).

In this chapter, we will demonstrate how malicious parties can exploit the privacy-performance dilemma in several privacy-preserving RFID authentication protocols to abuse the back-end server's computational resources. This means if an attacker can mount the attack on a large scale, it can cause denial-of-service attack on the back-end server. We then propose a few approaches to prevent this type of attack.

BACKGROUND

In this section, we briefly review several RFID authentication protocols with privacy-preserving property. Since protocols mentioned in this chapter use different approaches to achieve privacy-preserving property, we categorize them based on the complexity of looking up a tag in a tag database. The notations used to describe these protocols are depicted in Table. 1.1.

Table 1.1: Notations

Notation	**Description**
S	Back-end server
D	Tag database
N	Number of tags in D
T	RFID tag
$e(.)$	A one-way trapdoor function
$f(.)$	A pseudo-random function
$h(.), g(.)$	Two cryptographic hash functions
PRNG(.)	A pseudo-random number generator
$\|\|$	Bit string concatenation

It is important to mention some conventions which we will use to describe several different protocols in a consistent manner. For a tag being queried or an unknown tag, we do not use any subscripts in the notations related to the tag like its secret key and pseudonym. However, the same notations for tags in the database will have subscripts. By default, we assume that there are N tags in the back-end database and therefore the subscripts will range from 1 to N. We will also use some C-language conventions when describing how an entity processes information. For example, "return ACCEPT(i)" means a tag is successfully authenticated and identified as the tag numbered i in the database while "return REJECT" implies a failed authentication.

Note that, when describing each RFID authentication protocol below, we omit the role of RFID readers. We assume that an RFID reader just acts as an intermediate party which relays messages between the back-end server and an RFID tag. This assumption does not affect the point we are going to make in this chapter. However, when the role of RFID readers is required, we will mention it specifically.

Exhaustive-search Protocols

The most popular exhaustive-search protocol is the OSK protocol by Ohkubo et al. (2004). The protocol assumes that a tag can compute two cryptographic hash functions, $g(.)$ and $h(.)$. A tag is given an initial identifier s^1 which is also stored in the database at the back-end server. After each interrogation session, the tag identifier is updated in a chaining fashion, that is $s^{k+1} = h(s^k)$ after the k-th authentication session.

During the k-th authentication session, a tag computes its authentication token r^k as the hash of its current identifier, *i.e.,* $r^k = g(s^k)$. To verify a tag's authentication token, the server tries to match the tag being queried with all tags in the server's database. In particular, for an initial identifier of the i-th tag in the database, the server computes $g(s_i^1), g(s_i^2), \cdots$ for $i = 1, 2, \cdots, N$ until a match is found. We illustrate the OSK protocol in Fig. 1.1 where M is the maximum number of times the back-end server will try to match a tag being queried with another tag in the database.

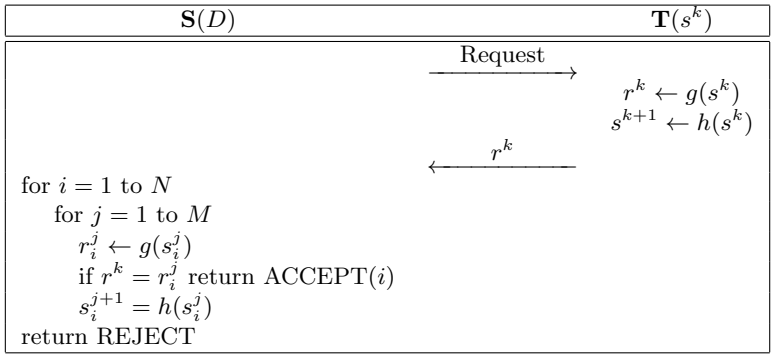

Figure 1.1: The k-th Authentication Session of a Tag in OSK Protocol

As we can see, the back-end server in the OSK protocol needs to perform $O(NM)$ hash operations to identify one tag in the database. This is the worst overhead for the sake of providing privacy among all RFID privacy-preserving authentication protocols and therefore we call the OSK protocol a brute-force protocol.

Tree-based Protocols

Tree-based protocols are among the first attempts to reduce the cost of privacy suffered by the OSK protocol. We review here a protocol by Molnar et al. (2005) which achieves $O(\log_2(N))$ instead of $O(NM)$ for the cost of identifying a tag. In order to achieve such a

reduced cost, a tree-based protocol in which a tree is employed to store the secret keys that the server shared with tags in the tag database. We call this protocol the Molnar-Soppera-Wagner protocol.

In the Molnar-Soppera-Wagner protocol, the tree of secrets is prepared as follows: First, a complete binary tree of depth d_1, in which each node is labeled with an independently and randomly chosen secret key, is generated. In this tree, each leaf node corresponds to a tag in the tag database; each tag is uniquely identified by a list of d_1 secret keys collected from the nodes along the path from the root to the leaf node representing the tag. A tag is initialized with these d_1 secret keys when it is admitted to the system. The tree is then expanded with $N = 2^{d_1}$ complete binary subtrees, each is of depth d_2 and rooted at a node corresponding to one tag. Each node in these subtrees is also assigned a secret key which is not chosen at random but computed from a deterministic pseudo-random number generator seeded with the secret key of the parent node. For example, let K be the secret key associated with the parent node and PRNG(.) be the pseudo-random number generator, then the secret keys associated with the left and the right child nodes are computed as the first and the second halves of the output of PRNG(K), respectively. It is easy to see that knowing the secret key associated with a node at depth d_1 is sufficient to compute all secret keys associated with this node's descendants. Each leaf node in a subtree is also assigned with a d_2-bit number as there are 2^{d_2} such leaf nodes per subtree; For our convenience, we assign numbers to leaf nodes in increasing order from left to right, *e.g.,* the left-most leaf node's number is 0 and the right-most leaf-node's number is $2^{d_2} - 1$, *etc.* It is also convenient to view the number assigned to a leaf node as a one-to-one correspondence to the path from the root of the subtree to the leaf node. Indeed, a tag pseudonym is computed from d_2 secret keys associated with nodes along such a path and d_1 secrets associated with nodes along the path from the root of the expanded tree to the root of the subtree. It is easy to see that each tag has at most 2^{d_2} pseudonyms.

Without loss of generality, assume that a tag is initialized with d_1 secret keys $(K_1, K_2, \cdots, K_{d_1})$ and an appropriate counter c. The tag database D contains the whole tree of secrets, whose depth is $d_1 + d_2$. A detailed description of the Molnar-Soppera-Wagner protocol is depicted in Fig. 1.2.

Note that the Molnar-Soppera-Wagner tree-based protocol does not aim to provide authen-

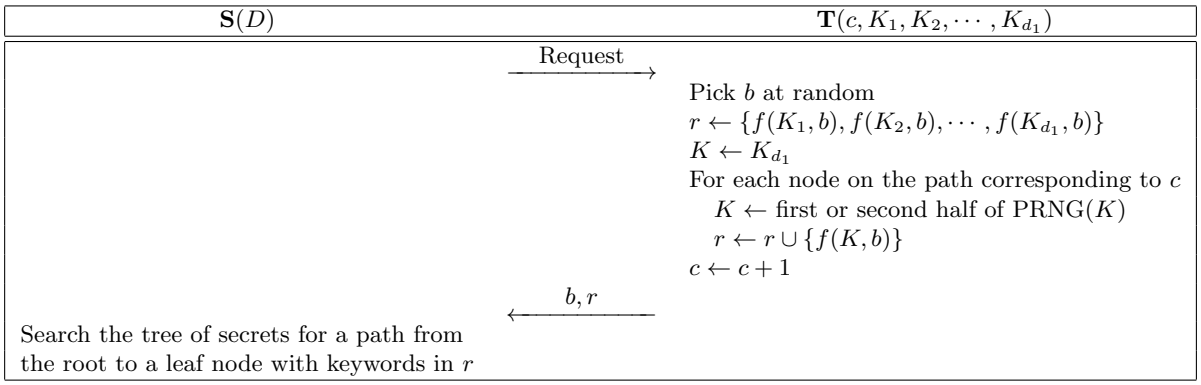

Figure 1.2: Molnar-Soppera-Wagner Tree-based Protocol

tication. Instead, its goal is only privacy and identification of tags.

Constant-cost Key-lookup Protocols

In this section, we will review several privacy-preserving RFID authentication protocols which claim to achieve constant cost of key lookup. These schemes represent the most advanced techniques in achieving privacy without degrading performance.

O-FRAP Protocol

O-FRAP, which stands for *Optimistic Forward secure RFID Authentication Protocol*, is one of the first protocols to claim constant-cost key-lookup proposed by Le et al. (2007). O-FRAP achieves privacy by giving each tag a random pseudonym (denoted by r_{tag} and updating this pseudonym after every authentication session. The pseudonym is maintained at both the tag and the back-end server which also uses the pseudonym to index the tag database. Each tag and the back-end server also share a secret key (denoted by k_{tag}). This secret key is also updated after every authentication to provide forward privacy. To deal with the de-synchronization of the shared secret key, the server keeps two versions of the secret key for each tag. One version is the secret key used in the last *normal* authentication session (denoted by k_{tag}^{prev}) and another version is the current secret key (denoted by k_{tag}^{cur}). The server updates this pair of secret keys by executing the $D.update(.)$ procedure as follows:

- If the tag is authenticated with k_{tag}^{cur}, *i.e.,* a normal authentication session, the server does: $k_{tag}^{prev} = k_{tag}^{cur}$ and $k_{tag}^{cur} = k_{tag}^{new}$ where k_{tag}^{new} is a newly generated key.

- If the tag is authenticated with k_{tag}^{prev}, *i.e.,* an abnormal authentication session, the server preserves k_{tag}^{prev} and lets $k_{tag}^{cur} = k_{tag}^{new}$. The reason that the server does not update k_{tag}^{prev} is to prevent further de-synchronization-of-secret attacks on this particular tag from de-legitimating the tag because this tag may no longer has the secret key that is actually kept in the back-end database.

In addition to two versions of the secret key for each tag, the server also keeps two corresponding versions of the tag pseudonym. Let $Prev_j$ and Cur_j denote two pairs ⟨pseudonym, secret key⟩ kept in the back-end database for one of previous session and the current session, respectively. A detailed description of O-FRAP is given in Fig. 1.3.

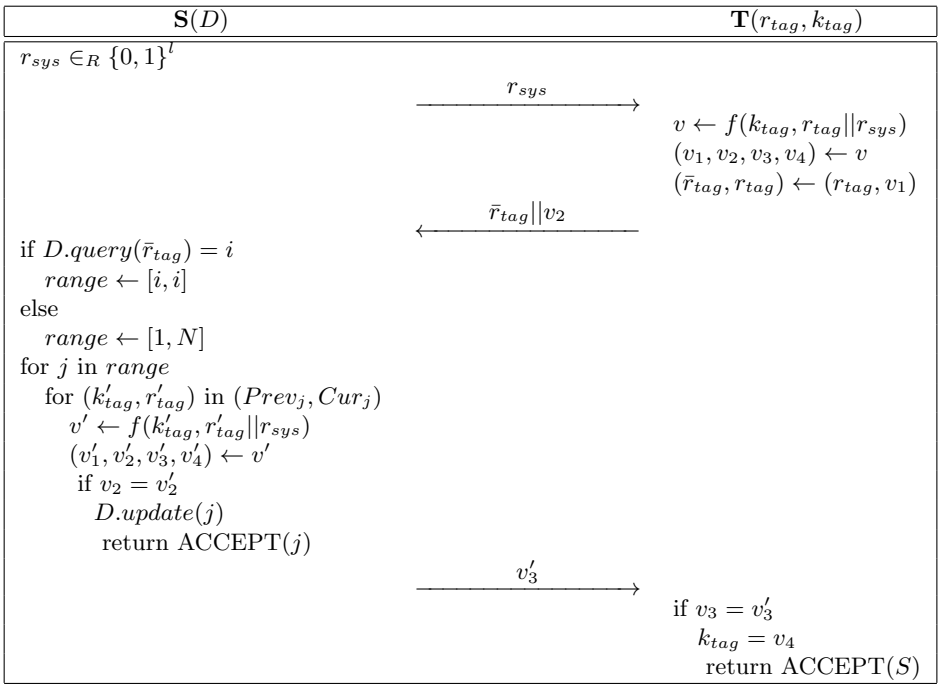

Figure 1.3: The O-FRAP Protocol

The O-FRAP protocol requires additional costs to update the database index as the tag pseudonym is updated after every authentication. For this reason, we place O-FRAP below two other protocols in the category of constant-cost key-lookup schemes.

Ryu-Takagi Protocol

Ryu & Takagi (2009) proposed a unique approach to addressing the cost-of-privacy problem.The pseudonyms for a tag are created by probabilistically encrypting the tag identifier so that when presented with a pseudonym, the back-end server can recover the unique and fixed tag identifier and quickly look-up the tag in the database. We call this protocol the Ryu-Takagi protocol. In the Ryu-Takagi protocol, a trusted party prepares tags and populates the tag database as follows: Each tag is assigned a unique identifier ID and a secret key K; The tag identifier is then encrypted m times to produce a set of encrypted tag pseudonyms $\Delta = \{r_1, r_2, \cdots, r_m\}$ where r_i is a ciphertext of ID; Lastly, the pair $\langle \Delta, K \rangle$ is stored in the tag's memory and the tag database D is populated with N pairs $\langle ID, K \rangle$ for N tags. A detailed description of Ryu-Takagi protocol is illustrated in Fig. 1.4.

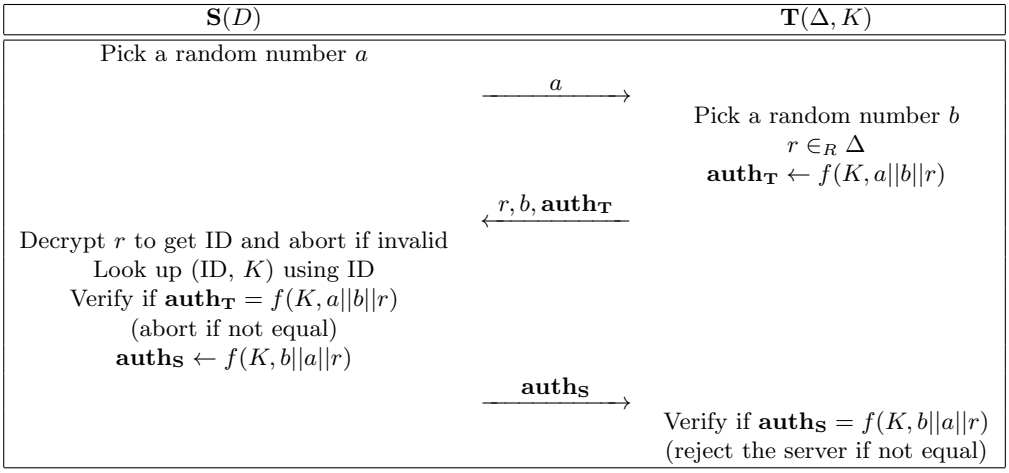

Figure 1.4: Ryu-Takagi Protocol

Note that, according to Duc et al. (2010), the Ryu-Takagi protocol does not satisfy the privacy-preserving property as claimed by its authors. The reason is that malicious parties can collect the list of m pseudonyms of a tag by repeatedly querying the tag. Duc et al. (2010) also suggested an improved protocol which uses the same idea that we will present later in this chapter.

Burmester-Medeiros-Motta Protocol

Burmester-Medeiros-Motta protocol proposed by Burmester et al. (2008) is not a new protocol per se. It is a generic method to convert any secure challenge-response protocol to

a secure privacy-preserving RFID authentication protocol with constant-cost key lookup. We describe Burmester-Medeiros-Motta construction by using a secure challenge-response authentication protocol depicted in Fig. 1.5.

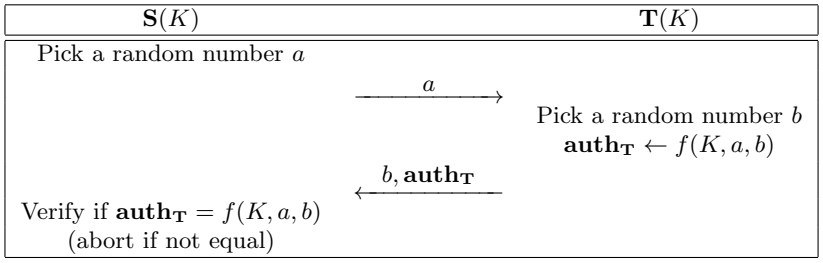

Figure 1.5: A Basic Challenge-Response Authentication Protocol

The key idea of Burmester et al. (2008) is somewhat similar to that of Ryu-Takagi protocol. More specifically, the tag identifier is encrypted to create tag pseudonyms. However, the different part here is that the tag is the one to encrypt its identifier, not the back-end server. As a result, a tag does not need to store a list of its pseudonyms because it can generate these pseudonyms on-the-fly. The encryption algorithm should work like a public-key cryptosystem so that the tag can encrypt its identifier and only the server can decrypt the tag pseudonym to get back the tag identifier. A symmetric cipher would be better suited for low-cost tags but there is a risk of leaking the secret key used to encrypt the tag identifier once a tag is corrupted. The question remained is how to design an efficient encryption scheme (or more technically, a one-way trapdoor function where the trapdoor can be used to invert/decrypt the function) that fits on the limited hardware of low-cost RFID tags. Fortunately, there is one candidate for such a scheme called SQUASH suggested by Shamir (2008). SQUASH is designed as a hash function and simply a modular squaring operation with modulus n as the product of two large prime numbers. The one-way property of SQUASH is equivalent to the security of Rabin's encryption scheme and the trapdoor information is the two prime numbers. As estimated in Burmester et al. (2008), when adapting SQUASH to some RFID authentication protocols, the cost of hardware should be less that 3,000 Gate-Equivalents for computation and storage.

Let $e(.)$ be an one-way trapdoor function and t be the trapdoor information, we describe the privacy-preserving RFID authentication protocol with constant-cost key-lookup converted from a basic challenge-response authentication protocol in Fig. 1.6.

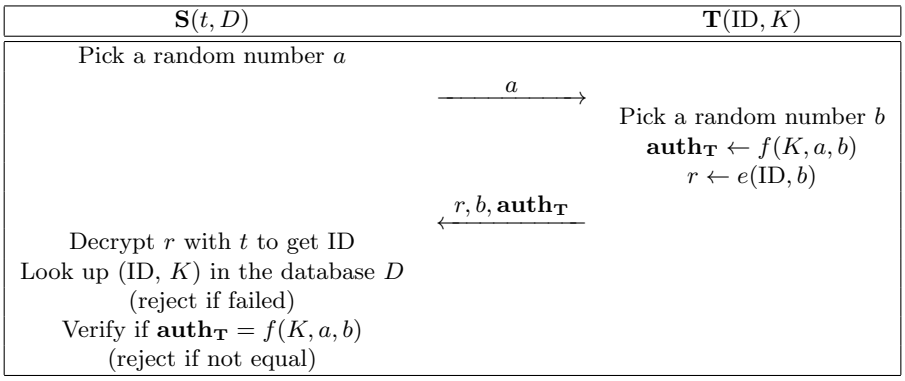

Figure 1.6: A Privacy-Preserving RFID Authentication Protocol with Constant-cost Key-lookup

DENIAL-OF-SERVICE ATTACKS ON PRIVACY-PRESERVING RFID AUTHENTICATION PROTOCOLS AND COUNTERMEASURES

Denial-of-Service Attacks on Privacy-Preserving RFID Authentication Protocols

We now describe how malicious parties can abuse the back-end server's computational resources by exploiting the server's behavior in handling the identification of tags being queried. First of all, in the OSK protocol, malicious parties can abuse the back-end server's computational resources as follows:

- Malicious parties send query requests to a large number of tags in order to make these tags update their identifiers. When these tags are actually scanned by the server, the server will have to compute a longer hash chain to find a match. Note that, if a tag is queried by malicious parties more than M times, this tag will no longer be identified by the server because the server only compute at most M values per hash chain. This constitutes a valid denial-of-service attack on the RFID system.

- Malicious parties themselves deploy a large number of fake tags. These tags simply backscatter random numbers when interrogated by the server. Since it is unlikely that the random numbers would match with some hash values, the server will have to go through all N hash chains for the N tags, each with M hash values, in the database before rejecting these tags. In other words, malicious parties can make the server run

into the worst-case scenario and therefore use the most of its computational resources.

In the case of the Molnar-Soppera-Wagner tree-based protocol, malicious parties can also mount similar attacks to abuse the server's computational resources. The difference here is that the attacker needs to eavesdrop a valid pseudonym and replay it to the server so that the server will waste most of its computational resources. Simply sending random pseudonyms is not effective because the server would stop the search early when there is invalid information on the pseudonym.

In Duc & Kim (2010), the authors presented two methods that malicious parties can use to abuse the back-end server's computational resources in the O-FRAP protocol. We summarize their observations here. In the O-FRAP protocol, the server tries match a tag being queried with every tag in the tag database ($range \leftarrow [1, N]$) when it fails to find a tag given a tag pseudonym ($\bar{r}_{tag}$). The reason for this behavior is that the tag pseudonym is updated by a tag whenever it receives a query request (r_{sys}). As a result, the tag pseudonym is an easy target for a de-synchronization attack which leads to failure in looking up a tag by its pseudonym. To identify tags that have been subject to de-synchronization, the server needs to go through the whole tag database. Similar to DoS attacks on the OSK protocol, malicious parties can mount the attack either by querying a large number of tags or deploying fake tags. We illustrate the two attack scenarios in Fig. 1.7.

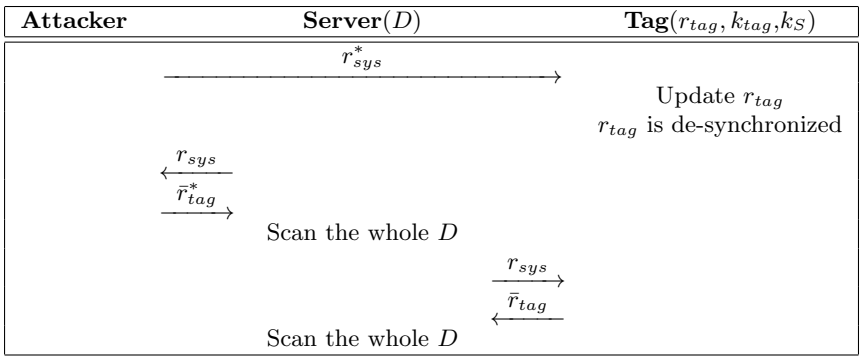

Figure 1.7: DoS Attack on O-FRAP

It is much harder to mount DoS attacks on Ryu-Takagi and Burmester-Medeiros-Motta protocols because the server in these two protocols never has to go through the whole tag database. Nevertheless, the server still has to decrypt the tag pseudonym each time it

interrogates a tag. Note that the server does not verify the integrity of tag pseudonyms backscattered by tags at all. This poses a potential weakness for malicious parties to exploit. For example, malicious parties can deploy a large number of fake tags at sensitive locations where the responsiveness of scanning activities are critical so that the server has a significant load to handle including decryption of tag pseudonyms in a short period of time. In such a scenario, it is really beneficial if fake tags are detected and filtered out before arriving at the back-end server. An issue with the Burmester-Medeiros-Motta protocol is the cost of implementing SQUASH on RFID tags, especially passive tags.

Countermeasures

We observe that a common problem leading to abuse of the server's computational resources is the lack of integrity checking on tag pseudonyms. That is, the server has no idea whether a pseudonym backscattered by a tag is valid or not. Therefore, we come to the conclusion that the server should be able to verify the validity of a tag pseudonym before actually using it for searching the tag in the database. More importantly, if this task can be done before the tag pseudonym reaches the server (*e.g.,* by the RFID readers who actually do the interrogation of tags), then the malicious party's attempt to push an illegitimate load on the back-end server will be stopped before the attacking traffic arrives at the server. To be able to verify a tag pseudonym even before identifying the tag, we propose that the server shares a common secret key K_S with all the tags in the database. The reasons are two-fold.

- Before a tag is identified, the server does not know the secret shared with the tag and therefore cannot use this secret key to verify the tag pseudonym.

- The key, K_S can also be used by the tag to verify the server before sending its pseudonym. This can prevent malicious parties from harvesting tag pseudonyms. This is how Duc et al. (2010) addressed the privacy issue of the Ryu-Takagi protocol.

We call our approach *two-phase authentication* because a tag is authenticated and identified in two phases:

- In the first phase, the tag pseudonym is authenticated so that the server is guaranteed that this particular tag is actually in the server's tag database. This phase can be done by the RFID readers provided that the server gives the RFID readers the secret key shared with all tags.

- In the second phase, the secret key shared between the server and only this tag is located in the tag database. At this point, the tag is identified and verified by this private secret key.

We apply this approach to a privacy-preserving RFID authentication protocol with constant-cost key-lookup in Fig. 1.6. We make the following modification: the server now shares with all tags a common secret key K_S; Before sending its response, a tag computes an authentication token for its pseudonym r as $\mathbf{auth_r} = f(K_S, a, r)$; Upon receiving the tag's response, the server (or the RFID readers in practice) first verifies the validity of the tag pseudonym r using $\mathbf{auth_r}$ and proceeds only if $\mathbf{auth_r}$ is correctly verified. The resulting protocol is illustrated in Fig. 1.8.

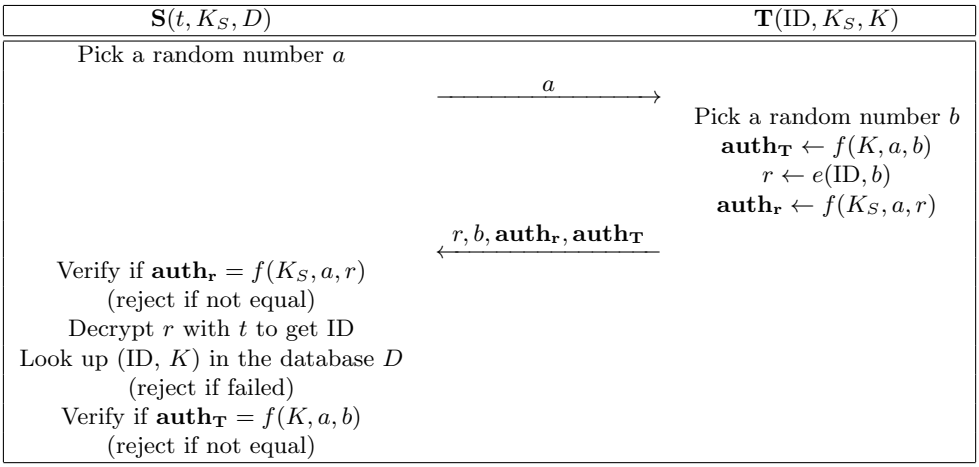

Figure 1.8: Our Proposed Privacy-Preserving RFID Authentication Protocol with Constant-cost Key-lookup

This same idea can be applied to the O-FRAP and Ryu-Takagi protocols as we can see in Duc & Kim (2010) and Duc et al. (2010), respectively. Indeed, the two-phase authentication approach can be used by most RFID authentication protocols which require detection of

illegitimate tags at the reader level.

FUTURE RESEARCH DIRECTIONS

Our proposed two-phase authentication approach is not without drawbacks. In particular, the same secret key has to be stored on every tag in the tag population. As a result, if one tag is corrupted and the secret key is leaked, it will affect all other tags in the system. Technically speaking, our two-phase authentication is not secure against key exposure since there is no way the key K_S can be updated and synchronized in all tags and the server. While security against the risk of key exposure is an overkilled feature in many applications, this is still a valid concern and needs to be addressed. We propose the following directions for further investigation:

- One may use some trusted computing primitives to protect the secret keys against leakage.

- The tag database can be partitioned into different parts and all tags in each part will share a different secret key. Then, when a tag is interrogated, the server and/or the RFID readers are required to identify which partition of the tag database the tag belongs to.

Another issue with the current situation in RFID security is the lack of research on cryptographic primitives that are suitable for low-cost RFID tags. For example, SQUASH is the only efficient one-way trapdoor function designed specifically for RFID tags. We need even more efficient schemes for really low-cost tags like passive tags.

CONCLUSION

In this chapter, our goal is to show that privacy and performance are often two conflicting goals in RFID system. We reviewed different privacy-preserving RFID authentication protocols and pointed out that they suffer from different levels of vulnerability against DoS

attacks. We then presented a generic method to address DoS attacks on privacy-preserving protocols, which we called two-phase authentication. The goal of two-phase authentication is to detect and filter out illegitimate tags at the reader level so that the server does not waste computational resources in processing these tags. We believe that privacy-vs-performance is a central issue in RFID security and much more research effort is needed in order to find an optimal solution.

ADDITIONAL READING SECTION

Stephen Weis, Sanjay Sarma, Ronald Rivest & Daniel Engels (2003). Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. *In the Proceedings of Security in Pervasive Computing*, Springer-Verlag, LNCS 2802, 201-212.

Stephen Weis (2003). *Security and Privacy in Radio-Frequency Identification Devices*, Master thesis, MIT, USA. Also available at `http://saweis.net/pdfs/weis-masters.pdf`.

Ari Juels, Ronald Rivest & Michael Szydlo (2003). The blocker tag: selective blocking of RFID tags for consumer privacy. *In the Proceedings of the 10th ACM conference on Computer and communications security*, ACM Press, 103-111.

Gildas Avoine & Philippe Oechslin (2005). A Scalable and Provably Secure Hash-Based RFID Protocol. *In the Proceedings of Workshop on Pervasive Computing and Communications Security - PerSec'05*, 110-114.

Gildas Avoine, Etienne Dysli & Philippe Oechslin (2005). Reducing Time Complexity in RFID System. *In the Proceedings of Selected Areas in Cryptography'05*, Springer-Verlag, LNCS 3897, 291-306.

Ari Juels (2007). The Vision of Secure RFID. *Proceedings of the IEEE*, Volume 95, Issue 8, 1507-1508.

Tadayoshi Kohno (2008). An Interview with RFID Security Expert Ari Juels. *IEEE Pervasive Computing*, 7(1):10-11.

Ari Juels (2006). RFID Security and Privacy: A Research Survey. *In the Journal of Selected Areas in Communication (J-SAC)*, Volume 24, Issue 2, pp. 381-395.

Serge Vaudenay (2007). On Privacy Models for RFID. *In the Proceedings of ASIACRYPT'2007*, Springer-Verlag, LNCS 4833, 68-87.

Mike Burnmester, Tri Van Le, Brene De Medeiros & Gene Tsudik (2009). Universally Composable RFID Identification and Authentication Protocols. *ACM Transactions on Information and Systems Security*, Vol. 12, No. 4, Article 21.
Dang Nguyen Duc (2010). *A Study on Cryptographic Protocols for RFID Tags*. Doctoral dissertation, KAIST, Republic of Korea. Also available at `http://caislab.kaist.ac.kr/publication/thesis_files/2010/Duc_Thesis.pdf`.
Gildas Avoine. (2010). RFID Security & Privacy Lounge. `http://www.avoine.net/rfid/`.

KEY TERMS & DEFINITIONS

EPC: EPC stands for *Electronic Product Code* which is a unique number stored in an RFID tag's memory. The EPC serves as the identifier for a product on which the tag is attached. An EPC can be retrieved via radio communication by RFID readers.

Authentication: Authentication is a process of proving one's identity (the prover) to another party (the verifier) over an insecure communication channel. A typical approach in designing an authentication protocol is to use a challenge-response protocol. That is, the verifier sends a random challenge and the prover replies with a response which is computed as a function of the verifier's challenge and secret information, *e.g.,* a secret key shared in advance between the prover and the verifier. The response can then be checked by the verifier in such a way that if the response is correctly verified, the verifier is confident that the prover indeed knows the secret information used to compute the response.

Privacy: Privacy has a broad meaning in cryptography. In case of RFID, privacy refers to the privacy of RFID tags which in turn means the privacy of people carrying tagged items. Intuitively speaking, the privacy of an RFID is guaranteed if its identity, movements and sometimes even its existence are hidden from unauthorized parties.

Forward Privacy: Forward privacy is a security notion which addresses the key exposure problem. As an RFID tag is generally a low-cost hardware, it is an easy target to be captured and dissected to reveal the secret key stored in the tag's memory. Forward privacy refers

to the ability to preserve the privacy of a tag up to the point of the secret key exposure. A common method to achieve forward privacy is to update the secret key and/or the tag pseudonym after every authentication session.

Unlinkability: Unlinkability is a frequent interpretation of privacy in RFID. Unlinkability usually means that it is infeasible for unauthorized parties to deicide whether two arbitrary authentication sessions involve a same tag. In other words, to the eyes of an outsider, an authentication session could be of any tag and therefore the outsider cannot learn any useful information and violate the privacy of tags.

Untraceability: Untraceability is an alternative interpretation of tag privacy. In simply means the impossibility to trace a tag by unauthorized parties.

Denial-of-Service (DoS) Attack: DoS attack refers to a type of attack in which the victim's computational resources are abused to the point where no legitimate activities can be served. In RFID, the DoS attack is usually targeted at the back-end server where all the detailed information about tagged objects are stored and queries are processed. The unavailability of the back-end server would mean a total break down of an RFID system.

De-synchronization Attack: De-synchronization attack refers to the acts by malicious parties which cause inconsistency in updating shared secret keys and sometimes pseudonyms between RFID tags and the back-end server. For example, an attacker may interfere in an authentication session to cause the server to update the shared secret key but the RFID tag. As a result, the tag might be no longer accepted by the server since the twos possess two different secret keys.

Pseudonym: Pseudonym is a temporary name used by an entity instead of its real name. The goal of using pseudonym is to preserve the anonymity of the entity when communicating over an insecure channel or to possibly other un-trusted entities.

REFERENCES

Mike Burmester and Breno de Medeiros & Rossana Motta (2008). Anonymous RFID authentication supporting constant-cost key-lookup against active adversaries. *International Journal of Applied Cryptography*, Volume 1, Issue 2, 79-90.

Dang Nguyen Duc & Kwangjo Kim (2010). Defending RFID Authentication Protocols against DoS Attacks. *To Appear in Elsevier's Journal of Computer Communications.*

Dang Nguyen Duc, Chan Yeon Yeun & Kwangjo Kim (2010). Reconsidering Ryu-Takagi RFID Authentication Protocol. *To Appear in the 2nd International Workshop on RFID / USN Security and Cryptography (RISC).*

David Molnar, Andrea Soppera & David Wagner (2005). A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags. *In the Proceedings of Selected Areas in Cryptography'05*, Springer-Verlag, LNCS 3897, 276-290.

Tri Van Le, Mike Burnmester & Breno de Medeiros (2007). Universally Composable and Forward Secure RFID Authentication and Authenticated Key Exchange. *In the Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, 242-252.

Miyako Ohkubo, Koutarou Suzuki & Shingo Kinoshita (2004). Efficient Hash-Chain Based RFID Privacy Protection Scheme. *In the Proceedings of International Conference on Ubiquitous Computing, Workshop Privacy.*

Adi Shamir (2008). SQUASH - A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. *In the Proceedings of Fast Software Encryption 2008*, Springer-Verlag, LNCS 5086, 144-157.

Eun-Kyung Ryu and Tsuyoshi Takagi (2009). A Hybrid Approach for Privacy-preserving RFID Tags. *Journal of Computer Standards and Interfaces*, Volume 31, 812-815.