Attacks on and Countermeasures for an RFID Mutual Authentication Scheme in Pervasive Computing Environment

Abedelaziz Mohaisen[1], Ku-Young Chang[2] and Dowon Hong[2]
[1] Computer Science and Engineering Department, University of Minnesota – Twin Cities
Minneapolis, MN 55455 – USA
[e-mail: mohaisen@cs.umn.edu]
[2] Information Security Research Division, Electronics and Telecommunication Research Institute
Daejeon, Repubic of Korea
[e-mail: {chang1090, dwhong}@etri.re.kr]
*Corresponding author: Dowon Hong

Received June 16, 2011; revised July 23, 2011; August 16, 2011; accepted August 28, 2011; published September 29, 2011

Abstract

We show that two protocols for RFID mutual authentication in pervasive computing environments, recently proposed by Kang et al, are vulnerable to several attacks. First, we show these protocols do not preserve the privacy of users' location. Once a tag is authenticated successfully, we show several scenarios where legitimate or illegitimate readers can trace the location of that tag without any further information about the tag's identifier or initial private key. Second, since the communication between readers and the database takes place over an insecure communication channel and in the plaintext form, we show scenarios where a compromised tag can gain access to confidential information that the tag is not supposed get access to. Finally, we show that these protocols are also vulnerable to the replay and denial-of-service attacks.

While some of these attacks are due to simple flaws and can be easily fixed, others are more fundamental and are due to relaxing widely accepted assumptions in the literature. We examine this issue, apply countermeasures, and re-evaluate the protocols overhead after taking these countermeasures into account and compare them to other work in the literature.

Keywords: RFID, mutual authentication, attacks and countermeasures, pervasive computing environments

This work was supported by the IT R&D program of MKE, Korea (Development of Privacy Enhancing Cryptography on Ubiquitous Computing Environment). The work of the first author was mainly done while he was a member of engineering staff at the Electornics and Telecommunication Research Institute.

DOI: 10.3837/tiis.2011.09.011

1. Introduction

Radio frequency identification (RFID) has been applied or incorporated into products for the purpose of identification and tracking. Because RFID tags are meant to identify objects, their authentication is considered one of the most challenging issues in the RFID security study avenue nowadays. Recently, to meet this goal Kang et al. [1] introduced a mutual authentication scheme that consists of two protocols for authenticating RFID tags. Both protocols are explicitly intended for pervasive computing environments. The authors claim that both protocols are secure against eavesdropping, traffic analysis, and the replay attack. Also, they claim that the protocols preserve the privacy of RFID tags so that no other tag holders, or tag readers can track the location of any legitimate tag in the system. The main assumption being altered, which is claimed by the authors to be an additional feature required in pervasive computing environments, is that they assume an insecure channel between the reader and the tag. This deviation from what is being conventionally used and widely accepted in literature (e.g., [2], [3], [4], [5], and [6]) opens the door wide open for several attacks and vulnerabilities.

In this paper, we show that these protocols are vulnerable to privacy breaching, confidentiality leakage, replay, and denial-of-service attacks[1]. Some of these attacks are fundamental and are due to the assumptions the authors make, and the guarantees they try to achieve. For these attacks, no absolute defenses are possible – but rather mitigations. These mitigations are explained in this paper. On the other hand, other attacks are preventable using some simple techniques, which we also explain in this paper. Though the attacks themselves are well known in literature, in other and similar contexts, this paper is meant to highlight that relaxing assumptions is always a threat to security guarantees.

While these attacks are specific to the protocols in hand, some others are known in the literature and have been applied to several prior works on designing secure authentication protocols with certain characteristics in similar contexts. Also, such specific attacks proposed in this paper are generic in the message they convey on the design and evaluation of secure authentication protocols for resources-conservative networks like RFIDs. In particular, the main take-away message of this work is that certain assumption accepted in the security and cryptography community, while annoying from a system point of view and could be hard to handle in practical contexts, are made that way for a wider benefit. Such benefits are made obvious in proving lower bounds on the security in these practical scenarios. Altering these assumptions would definitely downgrade both security and utility of designs and protocols. To recap, our attacks are not intended for a novel attacking methodology, but to show that simple known attacks in the literature are applicable, in many contexts powerful on designs that try to relax widely accepted assumptions in the literature. Indeed, the design we are considering here is not marginal, it has been recently cited in other refereed work, without any attempt to point out these shortcomings and attacks [7][8].

The rest of this paper is organized as follows. In section 2 we describe the second protocol introduced in [1] including our correction, which enables the protocol to work correctly, followed by overhead evaluation. In section 3 we introduce our attacks on the protocol. In section 4 we define countermeasures and mitigation strategies for the attacks followed a

[1] Though we consider the second protocol in this article with most of the details, the attacks in sections 3.1, 3.2, 3.3, and 3.4 as well as correction 5 in section 2.5 apply to the first protocol as well.

conclusive comparison. In section 5, we draw some concluding remarks. The rest of this section describes the system model, attack model, and major notation used in this paper.

2. Preliminaries

In this section, we review the preliminaries required to understand the context of the rest of the paper. We review the system and attacker model.

2.1 System Model

The system model consists of three entities which are tags, readers and the database. The communication channels between the tag and the reader and between the reader and the database are assumed to be insecure. Particularly, as suggested by [1], the insecurity of the channel between the reader and database motivates for the use of the proposed protocols in pervasive computing environments, in which communication is performed over open unauthenticated wireless channels. Each tag has its own unique identifier and each tag shares a group key with both of the database and the reader. Each tag also shares a private key with the database. Each reader shares a private key with the database. Tags are intuitively assumed not to have tamper-proof modules for cost restrictions. The environment in which the scheme is used is a typical pervasive computing environment.

2.2 Attacker Model

The attacker model is derived from the system model in section 2.1 and the general pervasive computing environment's assumptions – which is brought from the original paper. Particularly, the attacker has the ability to observe the communication that takes place between tags, readers, and database. The attacker has the ability to eavesdrop messages not directed to him and use them later for gaining access to information that he is not authorized to know. The attacker also can act as a legitimate reader and *try* to fool a tag to know its location or information that can lead to its location. Because the link between the reader and the database is assumed to be insecure, an attacker can be a tag, a reader, or a combination of them. Finally, it is intuitively understood in the pervasive computing environment that an attacker may take control over (compromise) some tags and statically analyze their contents. This applies not only to RFID tags (where tag compromise is possible [9]) but also to other components in pervasive computing environments, such like sensor nodes [10].

2.3 Notation

We use the notation used in [1] with some modification to ease the readability of the main protocol and its description. The notation used in this work is as follows:

- r is a random value generated using a random number generator. The values TS_1, TS_2, T_1, and T_2 are time stamps.
- t_i is a tag with an identifier TID_i. Each tag has a unique private key T^i_{key} shared with the database, and a unique group key GK_i shared with both of the reader and the database. The total number of tags in the network is n.
- $MetaID_i$ is a hash-locked ID of t_i expressed as $MetaID_i = H(TID_i \oplus r)$.
- $V_1, V_2, V_3, T_V, S_1, S_2, S'_1, S'_2, T'_V$ are values exchanged over the links tag-reader, reader-database, database-reader, and reader-tag. These values are computed as described

in section 2.4.

- *R* is a reader with an identifier *RID* and a unique secret key, *RK*, shared with the database. The reader shares group keys with each tag (i.e., reader stores GK_i for $1 \le i \le n$).
- $L[X]$ is the left half and $R[X]$ is the right half of a binary string X. The operator $A \| B$ appends the binary string B at the end of the binary string A.

2.4 Protocols for Mutual Authentication

In this section we describe the second authentication protocol introduced in [1], which is the advanced protocol, and motivate the attacks. All attacks described on this protocol also apply to the other protocol in [1], which is weaker than protocol described here.

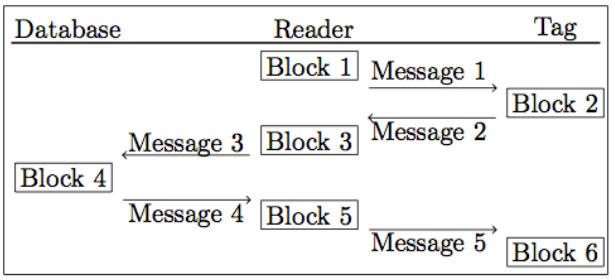

Fig. 1. A block and message flow diagram of the mutual authentication protocol.

This protocol, as shown in **Fig. 1** and explained in more details below, consists of 5 pairs of (Block, Message) and one termination (Block). In each block, processing is performed to authenticate one party at a time. Except in the last block which terminates the protocol, each block is followed by a message directed to another party to challenge and test his ownership of a shared secret. When the protocol is terminated successfully, the tag-reader, reader-database, and database-tag pairwise links are mutually authenticated. Because tags cannot communicate directly with the database, they communicate with the reader which passes their messages to the database on behalf of them. The protocol is assumed to resist eavesdropping, traffic analysis, replay attack, and location tracking. In addition to authentication, the protocol is claimed to provide integrity, anonymity, and confidentiality [1]. In this section we describe the second protocol of authentication in [1], bearing in mind that all attacks described later apply to this protocol. After describing the protocol, we provide a correction that ensures correctness of operation of the potocol.

- **(Block 1, Message 1):** This block initiates the protocol as follows

  1.1) $RID \oplus GK_i = V_1$

  1.2) $r \oplus GK_i = V_2$

  1.3) $S_1 = H(SID \| r \| TS_1)$

  Message 1 consists of $(V_1 \| V_2 \| L[S_1] \| TS_1)$. This message checks if the tag t_i owns the proper group key GK_i or not. GK_i is required for for the verification in Block 2.

  Notice that, before sending message 1 in this protocol the reader must know which tag it is trying to authenticate in order to compute the correct parameters (which include a group key shared with the proper tag). Accordingly, prior to the first block of the authentication protocol, the tag needs to proactively provide his identity which violates the purpose of the protocol. While this is not handled by any means in the original work in

[6], we suggest that group key is shared between the reader, the databse, and a set of tags identified by, for examples, a range of identifier. When a tag is to be authenticated, it provides the range of its identifier for which the reader uses the proper group key. In the extreme case (when the range covers the entire set of identifers used in the system) a single group key is used, which weakens our attack in section 3.2. In either of both scenarios, we assume that the reader has the ability to know which group key is to use without knowing the exact tag it is trying to authenticate, and thus not violating the prupose of the protocol in preserving privacy of the authenticated tag.

- **(Block 2, Message 2):** If this block is performed successfully, it proves that the tag owns the proper key GK_i, which he uses to compute both RID and r. This block consists of the following (the verification is step 2.4)
  - 2.1. $V_1 \oplus GK_i = RID$
  - 2.2. $V_2 \oplus GK_i = r$
  - 2.3. $S'_1 = H(RID \| r \| TS_1)$
  - 2.4. $L[S_1] \stackrel{?}{=} L[S'_1]$
  - 2.5. $MetaID_i = H(RID \oplus r)$

  Message 2 consists of $(MetaID_i \| L[T_V] \| R[S'_1])$. It is used to verify that a tag t_i owns the proper group key GK_i. Also, it challenges the database's ownership of the tag key T^i_{key} (see the correction below).
- **(Block 3, Message 3):** In this block, if the ownership of GK_i is verified, the reader-tag link is mutually authenticated. Also, in this block the reader challenges if the database owns the proper reader key RK. This block consists of the following:
  - 3.1. $R[S_1] \stackrel{?}{=} R[S'_1]$
  - 3.2. $RID \oplus GK_i = V_i$
  - 3.3. $V_3 = r \oplus RK$
  - 3.4. $S_2 = H(RID \| r \| TS_2)$

  Message 3 consists of $(MetaID \| L[T_V] \| V_1 \| V_3 \| L[S_2] \| TS_2)$, which challenges the database to verify if it owns the proper keys.
- **(Block 4, Message 4):** In this block, the database verifies its ownership of the proper reader key RK and authenticates the reader, if the verification is performed successfuly. It also verifies the ownership of T^i_{key} and authenticates the tag.
  - 4.1. $RID = V_1 \oplus GK_i$
  - 4.2. $r = V_3 \oplus RK$
  - 4.3. $S'_2 = H(RID \| r \| TS_2)$
  - 4.4. $L[S_2] \stackrel{?}{=} L[S'_2]$
  - 4.5. $All[TID \oplus r] \stackrel{?}{=} MetaID_i$
  - 4.6. $H(TIDi \| r \| T^i_{key}) = T'_V$
  - 4.7. $L[T_V] \stackrel{?}{=} L[T'_V]$

Message 4 consists of $(R[T_V] \| R[S'_2])$ which is used to verify the ownership of RK and T^i_{key} at the database side. The message's parts are directed to the tag and the reader respectively.

- **(Block 5, Message 5):** The reader checks the validity of $R[S_2] \stackrel{?}{=} R[S'_2]$ and passes Message 5 to the tag. Message 5 includes $R[T_V]$, which is used for the other side of the mutual authentication at the side of the reader. If this block is performed successfully, a mutual authentication is established between the reader and the database.
- (Block 6): this block consists of a single verification operation in which the tag authenticates the database by checking the validity of $R[T_V]$ which he received from the reader and verifying the correctness of $R[T_V] \stackrel{?}{=} R[T'_V]$. If this block is performed successfully, a mutual authentication is established between the tag and the database.

2.5 Correctness

Before discussing the protocol and addressing vulnerabilities and attacks, the following corrections are necessary for the correctness of this protocol's operation[2].

- In Block 1 at step (1.3), the operation $S_1 = H(SID \| r \| TS_1)$ should be changed into $S_1 = H(RID \| r \| TS_1)$, since the tag does not have *SID* for verification.
- In Block 2, $T_V = H(TID \| r \| T^i_{key})$ has to be computed and passed to reader. This is noted in Block 4 at step (4.6) for verifying the tag's ownership of the key T^i_{key}.
- In Block 2 at step (2.5), $MetaID_i = H(RID \oplus r)$ needs to be changed into $MetaID_i = H(TID \oplus r)$. This is observed in (4.5) since the comparison is performed over TID but not the RID for the verification part.
- In Block 3 step (3.2) needs to removed because it is already computed at the same side earlier in step (1.1), and could be chached for the latter verification operation.
- $All[TID \oplus r] \stackrel{?}{=} MetaID_i$ in (4.5) is modified into $All[H(TID \oplus r)] \stackrel{?}{=} MetaID_i$. That is, hashing operations are required for all identifiers of tags stored in the database after performing an exclusive-OR operation on each identifier with the random number generated by the reader in order to find out whether a tag is registered or not and retrieve the proper TID required for computing T'_V.

3. Attacks on Mutual Authentication Protocols

In this section we describe several vulnerabilities in the protocol explained in section 2.4. These vulnerabilities challenge the guarantees explicitly claimed in [1]. This includes the privacy of location claimed for tags and the confidentiality of secret information of the reader. It also shed the light on the replay and denial-of-service (DoS) attacks; two well-known attacks in the literature that could be easily applied to this protocol.

[2] Through a correspondence with the authors of [1], they acknowledged these corrections, which are not related to our attacks but are required for rather proving correctness of the protocols in hand.

3.1 Privacy-Breaching Attack

The first vulnerability challenges the privacy of a tag location, which the authors claim that their protocol preserves. According to [1], because the tag passes its own identifier, TID, to the reader in the form of $MetaID = H(TID \oplus r)$, the reader will not be able to track the tag even though the reader has r. The intuition behind this assumption and claim is that the reader needs to invert the hash function in order for a reader to compute TID, which is *computationally infeasible*. To this end, we introduce an attack that utilizes eavesdropping on the exchanged messages between a tag and a reader to track the location of a specific tag. In this attack, the reader does not need to know the identity of that tag in order to know its location. Notice that all of the proposed scenarios here, the intended attack breaks what is known in the literature as "unlinkability" which is the metric used in the design of these protocols for privacy preservation. The attack can be performed according to any of the following scenarios.

Scenario 1: This scenario of the attack, and its success chances, depend in part on the way according to which tags behave when error is faced. For example, it makes some difference to the outcome of this attack if the tag responds with an error message to the reader when the reader passes an invalid parameter to the tag, whereas the lack of response can be used to infer some information about the tag in hand. Even if GK is not known to the attacker, the attacker will still be able to distinguish a tag t_i with GK_i according the following scenario. If checking $L[S_1] = ? L[S'_1]$ does not hold true at the side of the tag, the tag t_i will not respond to the attacker since $GKi \oplus (GKi \oplus RID) \neq RID$. The attacker takes the lack of response as a positive sign about the potential of that the tag itself is the previously known tag to the reader, or if an error message is sent back to the reader, the reader knows for sure that the tag is potentially not among those being tracked. On the other hand, if the tag responds to the query issued by the reader, confirming that the equality above holds, certain leakage of privacy about the identity of the tag in question is realized, indicating that the reader has previously met the tag, and it is the tag of interest.

Scenario 2: Even worse, privacy is not concerned by only outsider attackers but also by the legitimate participants who may gain access to information of other participants, which they are not supposed to know [11]. For instance, a legitimate reader can try to fool tags by sending the same random nonce *r* every time and comparing the responses with previously registered values in order to breach the privacy of these tags by observing their locations. While the tag is supposed to be anonymous to the reader, as implied by the guarantees claimed in [1], this scenario even enables a further *anonymity leakage* attack on the tag in addition to enabling a reader to know the location of a tag.

Scenario 3: An illegitimate reader R' that controls a compromised tag t_a computes all possible group keys according to the method in section 3.2. At the running time of the protocol, R' registers both Message 1 and Message 2 shown in **Fig. 1**. Later, R' challenges a tag t_i, which has a group key GK_i, by sending Message 1, which contains $(V_1 \| V_2 \| L[S_1] \| TS_1)$ to that tag. In response to this challenge, the tag t_i performs the procedure in Block 2 and obtains RID and r. Also, t_i computes S'_1. Using these values, the tag t_i verifies the reader by checking $L[S_1] = ? L[S'_1]$. The result of this process will always hold given that the group-wise key used by the tag for verification is the same key used by the attacker for challenge. After that, t_i computes $MetaID' = H(TID' \oplus r)$ and sends it back to R' as in

Message 2, along with other values. Finally, R' checks $MetaID' = MetaID$. If the left-hand side equals the right-hand side, R' can be sure that this tag is the one met at the first place. Otherwise, the tag t_i is not the one met earlier. The fact that the potential attacker will be able to pass a query to an oracle (conceptualized by this attack) which will answer the query of whether the questioned tag in the query is the one known previously to the attacker or not fulfill the attack purpose, and breach tag's privacy.

Finally, to maximize the benefit of this attack, particularly in its first and second scenarios, the illegitmate reader R', representing the attacker, can potentially monitor the interaction between different tags and the legitimate reader for enough time and stores different pairs of (Message 1, Message 2) which correspond to different authentication rounds in relation with each tag's location. In a later phase, the illegitmate reader R' performs the above procedure of attack until $MetaID' = MetaID$ for the stored values obtained by observing the normal operation of tags. When both meta identifiers are equal, R' updates the location of the tag in question, and thus breach its privacy.

These different forms of the attack are only possible because the tag's rule is limited to verifying whether a reader owns a group-wise key, which is pre-shared between the reader and other tags in the system, where other tags do not participate in the challenge and response part of the protocol. As we will see later, a possible fix for this problem is done by involving the tags in the mutual authentication process; thus no single party dominates the authentication process by being the only entity generating the challenge. This type of fix for this problem is classically known in the literature of mutual authentication protocols.

3.2 Gaining Access to Confidential Information

The reader shares a key with the database, which the reader uses for authenticating himself to the database. We show that any tag can gain access to that key. Given that any tag can be compromised according to the attack model in 2.2, which is rationalized by the attacker model considering earlier in the literature, such information can be exposed to the attacker and endanger the traffic that takes place between the reader and the database for authenticating the reader and other tags as well. Also, by gaining access to the reader identifier, an attacker can compromise all group keys shared between the database, reader and all other tags without physically capturing them. Both scenarios violate the *confidentiality* of the protocol. Below, we elaborate on how this is possible for attacker using pieces of information exchanged in the protocol between honest entities.

In the session of authenticating tag t_i which is compromised by an attacker, $V_3 = r \oplus RK$ is exchanged in the *plaintext form* and can be eavesdropped by that attacker. The attacker who already received V_1 and V_2 for authenticating t_i defines $V_4 = V_2 \oplus V_3 = r \oplus GK_i \oplus r \oplus RK = GKi \oplus RK$. Because the attacker already knows GK_i, he can compute $RK = V_4 \oplus GK_i$.

Also, because the attacker already knows the reader's identifier RID at the time of his tag's authentication, the attacker can simply use RID to compromise the group keys shared between other tags, the reader, and the database. Suppose that GK_j is the group key assigned to tag t_j and shared with the reader and the database, the attacker can obtain the group key of

t_j by eavesdropping on t_j's authentication session, obtaining V_1 of t_j and finally computing $RID \oplus V_1 = GK_j$.

Beside putting the authentication of the reader at jeopardy, if an attacker knows *RK* he can compute the random nonce r used for authenticating a specific tag by eavesdropping V_3 of the tag's authentication session and computing $V_3 \oplus RK = r$. Revealing this value can be exploited in several ways. For instance, the attacker can try to compute that tag's identifier TID from the $MetaID$. If successfully computed, the attacker also can try to compute T_{key} via a brute force search on T_V. Practically, the efficiency of such attack will entirely depend on the length of TID and T_{key}, which are likely to have a small domain for efficiency reasons. These revealed values can be utilized to enhance the attack in section 3.1. If TID and T_{key} are computed successfully, an attacker can impersonate the tag in question easy and act as if it is the tag in hand.

It is worth noting that, unlike the other attacks explained in section 3.1 and those explained later in section 3.3 and 3.4, the attack on confidentiality is the most powerful among them because it reveals "secret credentials" to the attacker. Using these credentials, chances of success for a launched attack are independent of the way according to which honest tags behave. Furthermore, unlike other attacks which are limited in their applicability to the power of the attacker, and his ability to utilize credentials obtained using the used attack within the proper time (like the replay attack discussed below, for example), the credentials obtained in this attack have longer life, since they are secret information used mainly for authentication, making the attacker free from a lot of constraints.

3.3 Replay Attack

In a similar fashion to that used for determining the location of a specific tag, an illegitimate reader can utilize the replay attack. This illegitimate reader by nature does not own any legitimate identifier or reader key to be authenticated. To perform this attack, the illegitimate reader R' monitors the communication that takes place between legitimate reader R and the database and registers the exchanged values between the reader and the database $(MetaID \| L[T_V] \| V_1 \| V_3 \| L[S_2] \| TS_2)$. Even though R' does not own RK or GK, he can always utilize these received values to perform a replay attack and authenticate himself to the database. In addition to authenticating himself as a reader (first part of block 4 that uses $V_1, V_3, L[S_2], TS_2$), the attacker can act as a legitimate tag and authenticate himself as a tag without owning the proper T_{key} by simply forwarding the typical same values obtained from a previous authentication session to the database (i.e., $MetaID$ and $L[T_V]$).

Notice that, while the original protocol design does not make use of the timestamps for verification of validity and data exchanged between the reader and the database, we assume that this process is implicitly in use. Accordingly, for the attacker to get through this process, he needs to replay the exchanged authentication messages eavesdropped on the channel between the legitimate reader and the database within the allowed period. By doing so, we illustrate two interesting main issues. First, insecure channels between readers and the database allow for eavesdropping and, second, eavesdropped data can be used to thwart the utility of the authentication protocols.

3.4 Denial-of-Service Attack

To perform a strong denial-of-service attack on the database, an attacker needs to have eavesdropping capabilities only. An attacker, R', first eavesdrops Message 3 in **Fig. 1** which contains $(MetaID_i \| L[T_V] \| V_1 \| V_3 \| L[S_2] \| TS_2)$. After that, R' alters Message 3 into $(MetaID' \| L[T_V] \| V_1 \| V_3 \| L[S_2] \| TS_2)$ where $MetaID'$ is a fabricated identifier by R' and has same length as $MetaID_i$. In response to that, the database will perform the procedure in Block 4 of **Fig. 1**. Particularly, authenticity of R' will be verified for its first part as a reader successfully by executing steps (4.1) through (4.4). However, an abortion will occur after checking $All[H(TID \oplus r)] \stackrel{?}{=} MetaID'$ by computing $H(TID_i \oplus r)$ for $1 \le i \le n$. This abortion will occur because the database will fail to find a tag identifier that produces the same $MetaID'$. Though the process aborts in the end, the cost accumulated at the time of abortion is n hash operations where n is the number of identifiers for valid tags registered in the database. Since n can be millions of tags, the overhead of computing (4.5) for one time is large. By performing this attack frequently, the database will not be able to reply to authentication requests for genuine tags and denial-of-service will occur.

4. Countermeasures and Comparison

4.1 Countermeasures

Some of the attacks pointed in section 3 are fixable, some can be mitigated, and some are incurable. In this section we explore countermeasures and mitigation techniques for some of these attacks, and point out technical reasons behind incurability of other attacks.

4.1.1 Privacy-breaching attack

While the first scenario that assumes an attacker that compromises a tag is incurable for determining the location information[3], both of the second and the third scenarios of the attack are fixable as pointed in [11] for different protocols with similar vulnerability. For that, both of the tag and the reader are supposed to participate in generating the random nonce. Particularly, the tag can participate performing the following modifications in Block 2 of Figure 1: generate r_i, compute $MetaID_i = H(TID_i \oplus r \oplus r_i)$, Compute $V_t = T_{key} \oplus r_i$ and forward them in Message 2 to the reader. Now that the reader does not own r_i, which changes frequently, the reader cannot track the location of the tag. For authentication purpose, the database can compute r_i after receiving V_t and compute $MetaID_i$ to perform the search of $MetaID_i$. Note that exchanging the T_{key} after performing an XOR between it and a random nonce is yet dangerous so it need to be considered carefully as the only possible countermeasure.

4.1.2 Confidentiality violation

Confidentiality violation is possible because a tag can gain access to confidential data that the tag is not supposed to know. While RK is revealed because Message 3 is transmitted in plaintext form, other tags' keys are revealed because the same RID is used as part of V_1 for

[3] This case can be mitigated if the tag's random nonce used in *MetaID* is different from that used between the tag and the reader.

sending these keys as a challenge in the authentication process. To fix the first case, we modify the contents of Message 3. In order to hide PK in V_3, the reader must use a different random nonce from that used on the channel between the reader and the tag. To overcome the second case of confidentiality violation RID must be replaced by a random nonce generated by the reader, and this nonce needs also to be changed every time an authentication is performed for a specific tag. Note that the tag authenticates the reader for owning GK but not RID. For the first case of violation, the reader can alternatively encrypt Message 3 and forward it to the database. Because no tag knows the shared key between the reader and the database, this violation will be blocked.

4.1.3 Replay attack

To mitigate the replay attack, both of the legitimate reader and the database must be synchronized. Before performing authentication, the time stamp transmitted by the reader should be compared to the local time and taking sufficient transmission delay into account. Though this countermeasure mitigates the replay attack to a great extent, it does not prevent it entirely. Particularly, if an attacker knows the current time stamp at the reader or the database, he can easily fool the database and perform the replay attack correctly by replacing the time stamp of the previous authentication with a recent time stamp. To avoid that entirely, time synchronization and encryption of Message 3 in Figure 1 are to be performed by the reader. If Message 3 is encrypted, the attacker will not be able to distinguish the time stamp and will not be able to alter it. At the same time, if the decrypted time stamp is drifted from the current time of database after taking the transmission delay into account, the database will easily detect the existence of a replay attack and subsequently disregard the received authentication message.

4.1.4 Denial-of-service attack

The DoS attack is resistible if the reader encrypts Message 3 transmitted to the database so that an attacker cannot distinguish $MetaID_i$ from other random bits in the ciphertext. If the attacker cannot inject his forged $MetaID'$, only legitimate authentication messages will be passed to the database and unnecessary overhead will not be incurred at the database side, resisting the attack in entirety.

It is worth noting that three among four of the attacks introduced in this article can be fixed by enabling encryption of communication taking place at the link between the database and the reader, as conventionally assumed and used in the literature. This particularly challenges the assumption of [1], which claims that the protocol is secure even when the channel between a reader and a database is made insecure.

Table 1. A comparison between the introduced protocols in [1] and other protocols from literature after re-evaluating the overhead in [1] considering our correction and countermeasures. *r* stands for random number generator, *h* stands for hash functions, XOR stands for exclusive-OR, and ED stands for encryption/decryption capabilities. Protocol 1 and Protocol 2 are the protocols in question

Protocol	Operations and functions			Hashes		
	t	R	DB	t	R	DB
Protocol 1 [1]	*r*/*h*/XOR	ED/*r*/*h*/XOR	ED/*r*/*h*/XOR	3	3	$n + 3$
Protocol 2 [1]	*r*/*h*/XOR	ED/*r*/*h*/XOR	ED/*r*/*h*/XOR	2	2	$n + 2$
Reference 1 [12]	*h*/XOR	-	*r*/*h*/XOR	2	-	4
Reference 2 [13]	*h*/XOR	*r*	*h*/XOR	-	-	3
Reference 3 [14]	*h*/XOR	*h*/XOR	*h*/XOR	2	1	$n + 2$

4.2 Comparison

After fixing the protocol using our countermeasures in section 4 and considering the corrections in section 2.5, we compare the protocol in question to other protocols from literature. These protocols are particularly the related works described in [1]. For more details on these works, please refer to the original description in [1]. The comparison shown in Table 1 particularly considers the correction in step (4.5) that necessitates hash operation for each tag identifier in the database in the search process. It also adapts the functions required for countermeasures, which are random number generation r at the tag and encryption/decryption (ED) capabilities at both of the reader and the database. Note that Table 1 does not include the number of ED operations, which is one pair of operations at both of the reader and the database. At the side of the database, random number generation for defending against confidentiality leakage can be excluded if encryption is enabled since confidentiality leakage is implicitly curable by the ED operations. Notice that many of these works to which we compare the two protocols in question are quite old; however, the point is made clear and the purpose of the argument is well served since these protocols outperform the protocols in hand.

5. Conclusion

In this article we showed several vulnerabilities in the security of the RFID mutual authentication scheme in pervasive computing environment. Particularly, we showed that the scheme does not preserve the privacy of a tag location, does not maintain confidentiality, and vulnerable to replay and denial-of-service attacks. We introduced several mechanisms as countermeasures, which can successfully defend against these attacks. As a result, we re-evaluate the overhead of the modified scheme and compare it to other schemes in literature. While some of these attacks are specific to the protocols in hand, some are very generic and have been exploited previous on other protocols. We find that, while some assumptions are hard to deal with from a system point of view, they are necessitated by a rigorous theoretical examination, and getting rid of these assumptions would thwart the utility of systems and downgrade the security of built protocols.

References

[1] S.-Y. Kang, D.-G. Lee, I.-Y. Lee, "A Study on Secure RFID Mutual Authentication Scheme in Pervasive Computing Environment," *Computer Communications*, vol. 31, no. 18, pp. 4248-4254, 2008. Article (CrossRef Link)

[2] B. Alomair, L. Lazos, R. Poovendran, "Passive Attacks on a Class of Authentication Protocols for RFID," in *Proc. of the 10th International Conference on Information Security and Cryptology*, pp. 102-115. 2007. Article (CrossRef Link)

[3] H.-Y. Chien, C.-W. Huang, "A Lightweight Authentication Protocol for Low-cost RFID," *Journal of Signal Processing Systems*, vol. 59, no. 1, pp. 95-102. Article (CrossRef Link)

[4] H.-Y. Chien, C.-S. Laih, "ECC-based Lightweight Authentication Protocol with Untraceability for Low-cost RFID, *Journal of Parallel Distributed Computing*, vol. 69, no. 10, pp. 848-853, 2009. Article (CrossRef Link)

[5] P.D.Arco, A.De Santis, "Weaknesses in a Recent Ultra-lightweight RFID Authentication Protocol," in *Proc. of the Cryptology in Africa 1st International Conference on Progress in Cryptology*, pp. 27-39, 2008. Article (CrossRef Link)

[6] C.Y. Ng, W. Susilo, Y. Mu, R. Safavi-Naini, "New Privacy Results on Synchronized RFID Authentication Protocols Against Tag Tracing," in *Proc. of 14th European Conference on Research in Computer Security*, pp. 321-336, 2009. Article (CrossRef Link)
[7] T. Yeh, C. Wu, Y. Tseng, "Improvement of the RFID Authentication Scheme based on Quadratic Residues, *Computer Communications*, vol. 34 no. 3, p. 337-341, Mar. 2011. Article (CrossRef Link)
[8] T. Yeh, Y. Wang, T. Kuo, S. Wang, Securing RFID Systems Conforming to EPC Class 1 Generation 2 standard," *Expert Systems with Applications*, vol. 37 no. 12, pp. 7678-7683, Dec. 2010. Article (CrossRef Link)
[9] B. Song, C.J., Mitchell, "RFID Authentication Protocol for Low-cost Tags, in *Proc. of the first ACM Conference on Wireless Network Security*, pp. 140-147, 2008. Article (CrossRef Link)
[10] M. B. Paterson, D. R. Stinson, "Two Attacks on a Sensor Network Key Distribution Scheme of Cheng and Agrawal," *Journal of Mathematical Cryptology*, vol. 2, pp.393-403, 2008. Article (CrossRef Link)
[11] D. Molnar, D. Wagner, "Privacy and Security in Library RFID:Issues, Practices, and Architectures, in *Proc. of ACM Conference on Computer and Communications Security*, pp. 210– 219, 2004. Article (CrossRef Link)
[12] D. Henrici, P Muller, "Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers, in *Proc. of IEEE PerCom Workshops*, pp. 149-153, 2004. Article (CrossRef Link)
[13] T.V. Le, M. Burmester, B. Medeiros, "Universally Composable and Forward-secure RFID Authentication and Authenticated Key Exchange," in *Proc. of ACM Symposium on Information, Computer and Communications Security*, pp. 242-252, 2007. Article (CrossRef Link)
[14] J. Yang, J. Park, H. Lee, K. Ren, K. Kim, "Mutual Authentication Protocol for Low-cost RFID, in *Proc. of Workshop on RFID and Lightweight Cryptography*. pp. 17-24, 2005.

Abedelaziz Mohaisen is a Ph.D. candidate at the University of Minnesota – Twin Cities, where he received the M.S. degree in Computer Science in 2011. From 2007 to 2009, he worked as a member of engineering staff at the Electronics and Telecommunication Research Institute (ETRI) in Korea. In the summer of 2011, he was visiting the networking group at ICSI, a private non-profit research institute closely affiliated with the University of California, Berkeley, where he worked on global inference undermining data privacy. His research interest is in systems security, data privacy, applied cryptography, and social networks.

Ku-Young Chang received his B.S., M.S. and Ph.D. degrees in mathematics from Korea University, Seoul, Korea on 1995, 1997, and 2000. He is currently a principal member of engineering staff of Cryptography Research team at the Electronics and Telecommunication Research Institute, Korea where his research interests are broadly in the area of applied cryptography and finite field theory.

Dowon Hong received his B.S., M.S. and Ph.D. degrees in mathematics from Korea University, Seoul, Korea on 1994, 1996, and 2000. He is currently a principal member of engineering staff and the team leader of Cryptography Research team at the Electronics and Telecommunication Research Institute, Korea where his research interests are broadly in the area of applied cryptography, networks security, and digital forensics.