6

Security Analysis of the RFID Authentication Protocol using Model Checking

Hyun-Seok Kim, Jin-Young Choi, and Sin-Jae Lee
Korea University
Republic of Korea

1. Introduction

In RFID security(Gildas), few mechanisms focus on data protection of the tags, message interception over the air channel, and eavesdropping within the interrogation zone of the RFID reader(Sarma et al.a)(Weis et al.). Among these issues, we will discuss two aspects on the risks posed to the passive party by RFID , which have so far been dominated by the topics of data protection associated with data privacy and identity authentication between tag and reader.

Firstly, the data privacy problem states that storing person-specific data in an RFID system can threaten the privacy of the passive party. This party might be, for example, a customer or an employee of the operator. The passive party uses tags or items that have been identified as tags, but the party has no control over the data stored on the tags.

Secondly, the authentication will be carried out when the identity of a person or a program is checked. Then, on that basis, authorization takes place, i.e. rights, such as the right of access to data are granted. In the case of RFID systems, it is particularly important for tags to be authenticated by the reader and vice-versa. In addition, readers must authenticate themselves to the backend, however in this case there are no RFID-specific security problems.

There have been some approaches focusing on the RFID privacy and authentication issues, including killing tags at the checkout, renaming the identifier of the tag, physical tag password, hash encryption, random access hash and hash based ID variation. The last three approaches of these will be discussed in detail in this chapter. We will not discuss the remaining approaches in this chapter as they are physical solving approaches. The last three approaches are security protocols(Ryan & Schneider) that play the essential role of minimizing the burden of privacy and authentication problems. As with any protocol, the security protocol comprises a prescribed sequence of interactions between entities, and is designed to achieve a certain end. Security protocols are, in fact, excellent candidates for rigorous analysis techniques: they are critical components of distributed security architecture, very easy to express, however, extremely difficult to evaluate by hand.

Formal methods play a very critical role in examining whether a security protocol is ambiguous, incorrect, inconsistent or incomplete. Hence, the importance of applying formal methods, particularly for safety critical systems, cannot be overemphasized. There are two main approaches in formal methods, logic based methodology (Gong et al.), and tool based methodology (Hoare)(Lowe)(FDR). In this chapter, we specify hash based RFID security protocols(Sarma et al.a) as the previous work that employs hash functions to secure the RFID

communication using Casper (A Compiler for Security Protocol Analyzer)(Lowe) specifying tool. Then we verify whether or not it satisfies security properties such as secrecy and authentication using the FDR (Failure Divergence Refinement) model checking tool(FDR). After running the FDR tool, we reconfirm the existence of known security flaws in this protocol and propose the schemes of two security protocols for secure RFID communication. The contribution of this chapter is in analyzing the secure authentication protocols and designing new security protocols that could be widely researched in RFID systems. Therefore we provide a way for all methods, specifically Casper and FDR, which have been developed over the last decade by the theoretical community for the analysis of cryptographic protocols to be able to analyze RFID privacy and authentication problems. This especially applies to the new security protocols proposed in this chapter, which require read access control. If a reader requests a tag's ID, then the tag has to firstly identify that the reader is authenticated. In the process of authentication, the tag sends out a random number. The reader then responds to the tag with a function value of the random number and its own ID. The reader's output for each query changes, meaning that even if the output is eavesdropped, the adversary can not pass the authentication in the next query. The random number based authentication can prevent spoofing and man-in-the-middle attacks.
This chapter is organized as follows. In brief, Section 2 describes related work on RFID privacy and authentication schemes. Section 3 describes how RFID security protocols can be modeled in CSP (Communication Sequential Processes)(Hoare), generated by using Casper. Our analyzed results of the protocols will be described in Section 4. The proposed hash based and challenge response based security protocols are presented in Section 5. Finally, the conclusion and future work are addressed in the last section.

2. Background

2.1 The components of RFID system

RFID systems(Sarma et al.a)(Weis et al.) are made up of three main components, that we briefly describe as follows:

1. ***Transponder or RFID Tag*** In an RFID system, each object will be labeled with a tag. Each tag contains a microchip with some computation and storage capabilities, and a coupling element, such as an antenna coil for communication. Tags can be classified according to two main criteria: - The type of memory: read-only, write-once read-many, or fully rewritable. - The source of power: active, semi-passive, and passive.
2. ***Transceiver or RFID Reader*** RFID readers are generally composed of an RF module, a control unit, and a coupling element to interrogate electronic tags via RF communication. Readers may have better internal storage and processing capabilities, and frequently connect to back-end databases. Complex computations, such as a variety of cryptographic operations, may be carried out by RFID readers, as they usually do not suffer from the same limitations as those found in modern handheld devices or PDAs.
3. ***Back-end Database or Host*** The information provided by tags is usually an index to a back-end database (pointers, randomized IDs, etc.). This limits the information stored in tags to only a few bits, typically 96, which is a sensible choice, due to severe tag limitations in processing and storing. It is generally assumed that the connection between readers and back-end databases is secure, because processing and storing constraints are not as constrained in readers, and therefore common solutions such as SSL/TLS can be used.

2.2 Related work with security problem in RFID

There have been many papers in the literature that attempt to address the security concerns raised by the use of RFID tags in RFID system. The last three approaches outlined below are discussed in details in this chapter. We will not discuss the remaining approaches since they are physical solving approaches.

2.2.1 Kill tag

The Auto-ID Center explicitly designed the EPC(Electronic Product Code) (EPCGLOBAL INC.) kill command as a pro-privacy technology. The designers realized that EPC tags might be irretrievably embedded in consumer devices and consumers might not want to be tracked. They viewed killing EPC tags at the point-of-sale as an easy way out of the apparent privacy dilemma. The underlying principle is that "dead tags don't talk". As an alternative to killing EPC tags, tags can also be attached to a product's price tag and discarded at the point-of-sale.

2.2.2 Renaming approach

Even if the identifier emitted by an RFID tag has no intrinsic meaning, it can still enable tracking. For this reason, merely encrypting a tag identifier does not solve the problem of security. An encrypted identifier is itself just a meta-identifier. It is static and, therefore, like any other serial number, is subject to tracking. To prevent RFID-tag tracking, it is necessary that tag identifiers be suppressed, or that they change over time.

2.2.3 Tag password

Basic EPC(EPCGLOBAL INC.) RFID tags have sufficient resources to verify PINs or passwords. At first glance, this appears to be a possible vehicle for privacy protection: A tag could emit important information only if it receives the right password. The paradox here is that a reader doesn't know which password to transmit to a tag unless it knows the tag's identity. Passwords might still prove useful in certain environments. For example, retail stores could program tags at checkouts to respond to a particular password permitted by the RFID network in a consumer's home. This would protect consumersą́ privacy between a store and their homes. If consumers want to use RFID tags in multiple environments, however, they would face a challenging password management problem.

2.2.4 The hash lock scheme

A reader defines a "Lock" value by computing lock = hash(key)(Sarma et al.a) where the key is a random value. This lock value is sent to a tag and the tag will store this value into its reserved memory location (i.e. a metaID value), then the tag will automatically enter into the locked state. To unlock the tag, the reader needs to send the original key value to the tag, and the tag will perform a hash function on that key to obtain the metaID value. The tag then has to compare the metaID with its current metaID value. If both of them match, the tag unlocks itself. Once the tag is in unlocked state, it can respond with its identification number such as the EPC(EPCGLOBAL INC.) to readers' queries in the forthcoming cycles. This approach makes it simple and straightforward to achieve data protection, i.e. the EPC code stored in the tag is being protected. Only an authorized reader is able to unlock the tag and read it, then lock the tag again after reading the code. This scheme will be analyzed in this chapter in detail in Section 4.1.

2.2.5 The randomized hash lock scheme

This is an extension(Weis et al.) of the hash lock(Sarma et al.b3) scheme based on Pseudo Random Functions (PRFs). An additional pseudo-random number generator is required to embed into tags for this approach. Presently, tags respond to reader queries by a pair of values (r, hash(IDk ‖ r)) where r is the random number generated by a tag, IDk is the ID of the k-th tag among a number of tags in ID1, ID2, . . ., IDk, . . ., IDn. For reader queries, the tag returns two values. One is the random number. The other is a computed hash value based on the concatenation(‖) on its own IDk and r. Once the reader gets two values, it retrieves the current N number of ID (i.e. ID1, ID2, . . ., IDn) from the backend database. The reader performs the above hash function on each ID from 1 to n with r until it finds a match. When the reader finds a match, the reader is able to identify that tag k is on its tag's ID list (i.e. tag authentication). The reader will then send the IDk value to the tag for unlocking it. This scheme also will be analyzed in detail in Section 4.2.

2.2.6 The hash based ID variation scheme

Henrici-M*ü*ller(Henrici & Muller) propose a 4-round protocol for low-cost RFID systems based on a one way hash function and a random number generator. The protocol begins as the reader queries the tag, and the tag responds with its hashed identification and current transaction number. The response is forwarded by the reader to the back-end server for validation. To identify the tag, the server checks the validity of the identifying information of the tag. The server then concludes by sending a random number to the tag so that the tag's identification is refreshed and synchronized. This scheme also will be analyzed in detail in Section 4.3.

3. Modeling RFID security protocols in CSP

In this section we describe how the security protocol of the RFID system is modeled using CSP(Communication Sequential Processes)(Hoare) and how this model allows us to reason about it. We denote R as Reader, T as Tag and m as Message, for figuring out the RFID system.

3.1 The message datatype

The datatype Message represents the messages exchanged between the different agents. It is based on a set of atoms called Atom where the set of Key (contains session keys used in RFID), Nonce and Text(for Authenticating between Reader and Tag) are defined as subsets of the atom set (Key $\subseteq$ Atom, Text $\subseteq$ Atom and Nonce$\subseteq$Atom). In addition, we define HashFn to be the set that contains all the available cryptographic hash functions. The datatype Message is composed of encrypting, hashing, sequencing and the atomic value in the Atom and is defined by:

Message ::= Encrypt.Key.Message | Hash.HashFn.Message | Sq.Message* | Atom.Atom

In this chapter we use the Casper notation of writing {m}k for Encrypt.k.m. We use H(|m|) for Hash.H.m and abbreviate Sq.<m_1,. . . ,m_n> to <m_1,. . . ,m_n>. For example, we denote the construct Encrypt.k.(Sq.<a,n_a>) by $\{a,n_a\}_k$.

3.2 Trustworthy agents

Every agent taking part in the protocol is modeled as a CSP process. (An agent can also be internalized in the intruder deduction set (Broadfoot & Roscoe), but for now we assume that all honest agents are implemented as CSP processes). We define the process P_R, denoting agent R, using the following events:

- *send*.R.T.M - symbolizes agent R sending message M to agent T.
- *receive*.T.R.M - symbolizes agent T receiving message M apparently from R.

In addition, we define the following events for delineating specifications for the protocol we want to analyze. (See (Lowe) for more details, on how these events are used to express properties of security protocols)

- *claimSecret*.R.T.M symbolizes that R thinks that message M is a secret shared only with agent T.
- *running*.R.T.M_1,. . . ,M_n symbolizes that R thinks he started a new run of the protocol with T where M_1,. . . ,M_n represent some details of this run.
- □*nish*.R.T.M $_1$,. . . ,M_n symbolizes that R thinks he has just finished a run of the protocol with T where M_1,. . . ,M_n represent some details of this run.

For more information regarding the translation of a protocol description to a CSP representation see (Ryan & Schneider).

3.3 Modeling the intruder and putting the network together

Based on the Dolev-Yao model (Dolev & Yao), we allow the intruder to have the following abilities when attacking a set S of trusted agents: (i) overhearing all messages flowing through the network, (ii) intercepting messages, (iii) faking messages based on what he knows limited only by cryptography, and (iv) behaving as would any agent outside of S. We first define the rules that allow the intruder to construct new messages. The definition is based on the relation $\vdash$, which characterizes deduction rules by which the intruder can deduce new messages. We say that $B \vdash M$ if message M can be deduced from the set of messages B.

member $B \in M \Rightarrow B \vdash M$
sequencing $B \vdash \{M_1, \ldots, M_n\} \Rightarrow \{M_1, \ldots, M_n\}$
splitting $B \vdash <\ldots, M, \ldots> \Rightarrow B \vdash M$
encrypting $B \vdash M \wedge B \vdash \text{Atom } K \wedge K \in \text{Key} \Rightarrow B \vdash \{M\}K$
decrypting $B \vdash \{M\}K \wedge B \vdash \text{Atom } K^{-1} \Rightarrow B \vdash M$
hashing $B \vdash M \wedge H \subseteq \text{HashFn} \Rightarrow B \vdash H(|M|)$

Informally, the intruder can conduct encryption when he knows the message and the key. He can decipher an encryption for which he knows the inverse of the key, create a reference to a key that he knows, hash every message he knows and can both break up and form sequences. Since by using the Dolev-Yao model, the intruder should be able to overhear, intercept and block each message, the intruder process also models the communication medium. The process representing the intruder is parameterized by **X**, which ranges over subsets of Message, and represents all the facts the intruder has learned. In this model the intruder gets every message sent by the honest agents or by the server via the *send* channel. He then can pass it to the agents via the appropriate *receive* channel unless he decides to block it or fake a new message instead.

$\text{Intruder}(X) \mathrel{\widehat{=}} \Box_{M \in Message} \text{send?R?T!M!} \rightarrow \text{Intruder}(X \cup \{M\})$
$\Box_{M \in Message, X \vdash M} \text{receive?R?T!M!} \rightarrow \text{Intruder}(X)$
$\Box_{M \in Message, X \vdash M} \text{leak.M} \rightarrow \text{Intruder}(X)$

The initial state of the intruder is IIK(Intruder Initial Knowledge). The complete system is then:[1]
SYSTEM $\widehat{=}$ $(|||_{A \in Honest} P_A)$ || INTRUDER(IIK)

3.4 Specifying protocol requirements

The requirements of the protocols are encapsulated by *trace speci□cations*.
Secrecy As mentioned in Section 3.2, secrecy is when agent R performs the event *claimSecret.R.T.Secs*, and that we believe, at this point in the protocol run, that the values in the set *Secs* are secret and shared only with agent T. It expresses the expectation that the intruder cannot be in possession of values from *Secs*, i.e. the intruder should not be able to perform *leak.M* where M $\in$ *Secs*.
Authentication We first introduce the *□nish* and *running* events (See (Ryan & Schneider) for more details)[2]. The *□nish* event is performed by the honest agents when they complete a protocol run and the running event should be performed before the last send event. We will use the following definition which is one of the more common forms of authentication:

> If Reader thinks he has completed a run of the protocol with Tag, then Tag has previously been running the protocol, with Reader, both agents agreed on the roles they took, both agreed on the values of the variables $v_1, \ldots, v_n$, and there is a one-to-one relationship between the runs of Tag and the runs of Reader.

The following specification corresponds to this definition[3]:

> $Agreement_{AgreementSet}(tr) \widehat{=}$
> $\forall R \in Agent; T \in Honest; Ms \in AgreementSet \bullet tr \downarrow \square nished.R.T.Ms \leq tr \downarrow running.T.R.Ms$

4. Analyzing hash based protocols

In this section we specify hash based protocols introduced briefly in Section 2 using Casper and describe the result of verification. In particular, we obtained each CSP model of hash based protocol through the process of code generation from Casper.
Message Datatype, Trustworthy Agents and Intruder Model specified in Section 3 are applied in hash based protocols as identical models in this section and Protocol Requirements are applied in analyzing the verification of each of the protocols.

- **The message datatype in hash based protocols** The *metaID*(in Fig.1) datatype represents the metaID messages that are sent and received by the agents. Similar to the Message datatype, *metaID* will be based on the Atom set, where Key $\subseteq$ Atom, and on HashFn, the set that contains all cryptographic hash functions. In addition, we define % notation as *metaID*. The % notation is used so that the metaID can be forwarded to other participants. This is why a reader can not construct the metaID, since the other reader does not know the value of the hash function, where m is a message and v is a variable, denoting that the recipient of the message should not attempt to decrypt the message m, but should instead store it in the variable v. Similarly, $v \% m$ is written to indicate that the sender should send the message stored in the variable v, and the recipient should expect a message of the form

[1] For clarity this model is abstracted. In the model generated by Casper each fact is modeled as a process paralleled with the entire fact space. This technique reduced dramatically the state space that FDR needs to explore (see (Ryan & Schneider) for more details).
[2] notice that (Ryan & Schneider) refers to these events as *Running and Commit*
[3] The binary operator $\downarrow(tr \downarrow e)$ represents the number of occurrences of event e in a trace tr.

given by m. Therefore, *metaID* could not be known the result value of the hash function for tag by first receiver.

- **Trustworthy Agents** In hash based protocols, every agent taking a part in the protocol sends and receives in the communication channel as described in Section 3.2
- **The Intruder** The Intruder's definitions in the hash based protocol models are the same as the one in the RFID model in Section 3.3 and is still based on the basic six deduction rules presented in Section 3.3.

T	RF tag's identity
R	RF reader's identity
DB	Back-end server's identity that has a database
Xkey	Session Key generated randomly from X
metaID	Key generated from reader using hash function
ID	Information value of tag
Xn	A random nonce generated by X
H	Hash function

Table 1. The Hash Lock Scheme Notation

4.1 Hash unlocking protocol

Message 1. R $->$ T : Query
Message 2. T $->$ R : (H(Rkey)) % metaID
Message 3. R $->$ DB : metaID % (H(Rkey))
Message 4. DB $->$ R : RKey, ID
Message 5. R $->$ T : RKey
Message 6. T $->$ R : ID

Fig. 1. The hash unclocking protocol

The general overview of the above protocol(Fig.1) was already described in Section 2.2.4(Sarma et al.a).
To unlock the tag, in the first line, the reader needs to send a query to the tag and the tag sends the metaID to authenticate with the reader (Message 1,2). The reader forwards this metaID to DataBase to confirm his identity (Message 3). The DataBase compares the metaID with its current metaID value and ,if both of them match, it lets the reader know the key and ID of the tag (Message 4). The reader authenticates his identity with the tag sending key received from the database (Message 5). As a result, if both of them match, the tag unlocks itself. Once the tag is in an unlocked state, it can respond with its identification number(*ID*) to queries of readers in the forthcoming cycles (Message 6).

4.1.1 Protocol requirements and verification results

We describe the properties, i.e *Secret* property associated with data privacy and *Agreement* property associated with authentication between tag and reader, then show verification results of the safety specifications of the hash unlocking protocol in the hash lock scheme, using traces refinement provided in the FDR tool. In particular, we focus on the Session key(Rkey), ID for tag and communicating messages to verify the requirements.
After running the FDR model checking tool, this protocol does not satisfy the *Secret* and *Agreement* requirements and the testing result of the protocol can be described in CSP as below.

1. $Secret_{R,T}$(tr) = $\forall$ m • signal.*Claim_Secret.R.T*.m in *tr* $\wedge$ R $\in$ Honest $\wedge$ T $\in$ Honest $\Rightarrow$ (*leak.Rkey* in tr)

   For all message m, through trace specification(tr), the message Rkey was leaked by an intruder. Therefore, T cannot ensure the *key(Rkey)*. That is, in message 2, the confidentiality of the Rkey cannot be ensured due to the sniffed metaID(H(Rkey)) value. This makes a replay and man-in-the-middle attack possible.

2. $Secret_{R,T}$(tr) = $\forall$ m • signal.*Claim_Secret.R.T*.m in *tr* $\wedge$ R $\in$ Honest $\wedge$ T $\in$ Honest $\Rightarrow$ (*leak.ID* in tr)

   For all message m, through trace specification(tr), T's data(ID) was leaked by an intruder. It is possible to be sniffed a identity easily by an intruder in message 4. The privacy problem of the user will be brought out.

3. Agreement$_{AgreementSet}$(tr) $\widehat{=}$ R $\in$ Honest $\Rightarrow$ *signal.Running_Initiator.T.R.ID.Rkey* precedes *signal.Commit_Responder.R.T.ID.Rkey*

   If agreement is required on some or all of this information, then the signal event at the end of the responder's run should be *signal.Commit_Responder.R.T.ID.Rkey* and it should follow an event *signal.Running_Initiator.T.R.ID.Rkey* in the initiator's run. It means that the responder is not even in possession of all information until receipt of the last message, so the only possible for the commit message is right at the end of the protocol. Similarly, if the initiator is not in possession of all the information until just before its final message, the *Running* signal should either precede or follow that message. However, in this protocol, we cannot guarantee that the corresponding Running signal has occurred, provided we assume that the responder is honest: that $R \in Honest$. The intruder can capture messages and modify them. This may result in a failed key agreement between two agents.

Through debugging the counter-example trace events, we confirm that the hash unlocking protocol may be susceptible to a sniff and spoof attacks by an intruder due to the unsecured communication channel between reader and tag. A general attack scenario that could be found in this protocol is described below; *I _ Agent* means an intruder who can sniff messages and spoof his identity.

```
1.    Tag      ->  I_Reader   : H(RKey)
2. I_Mallory   ->  DataBase   : H(RKey)
3.  DataBase   -> I_Mallory   : RKey, ID
```

An intruder may obtain the current metaID value(H(RKey)) by querying a tag. The intruder replays the obtained metaID value and broadcasts it to any readers nearby, to get the specific random key for this metaID value if any reader responds to his replay. Therefore, the intruder may have a chance to get the key to unlock the tag and obtain its data.

4.2 The randomized hash unlocking protocol

This is an extension of the hash unlocking protocol. In the randomized hash unlocking protocol (Sarma et al.a), the tag makes a random number, and then sends it to the reader as a response on every session. For reader queries, the tag returns two values. One is the random number (Rn). The other is a computed hash value based on the concatenation($\|$) of its own IDk and Rn. Once the reader gets two values, it retrieves the current N number of ID (i.e. ID1, ID2, . . ., IDn) from the backend database. The reader will perform the above hash function on each ID from 1 to n with Rn until it finds a match. When the reader finds a match, the reader is able to identify that tag k is on its tag's ID list (i.e. tag authentication). The reader will then

send the IDk value to the tag for unlocking it. Fig. 2 shows the process of the randomized hash unlocking protocol.

Message 1. R $->$ T : Query
Message 2. DB $->$ R : ID1, ID2, ..., IDk , ..., IDn
Message 3. T $->$ R : Rn, H(IDk $\|$ Rn)
Message 4. R $->$ T : IDk

Fig. 2. The randomized hash unlocking protocol

4.2.1 Protocol requirements and verification results

After running the FDR model checking tool, this protocol satisfies the first *Secret* requirement regarding the *RKey* in Section 4.1.1 because this protocol does not use the session key in the communication channel. However this protocol does not satisfy the other two requirements, as follows;

1. $Secret_{R,T}$(tr) = $\forall$ m $\bullet$ signal.*Claim_Secret.R.T*.m in *tr* $\wedge$ R $\in$ Honest $\wedge$ T $\in$ Honest $\Rightarrow$ (*leak.IDk* in tr)

   For all message m, through trace specification(tr), the message IDk was leaked by an intruder. That is, IDk is sent to the tag through the insecure channel. Therefore, the tag can be tracked. In addition, the protocol is vulnerable to a replay attack since the attacker can masquerade as the right tag when the attacker overhears the tag's response(Rn, H(IDk $\|$ Rn)) then sends it to the reader.

2. Agreement$_{AgreementSet}$(tr) $\widehat{=}$ R $\in$ Honest $\Rightarrow$ *signal.Running_Initiator.T.R.Rn.H(IDk $\|$ Rn)* precedes *signal.Commit_Responder.R.T.Rn.H(IDk $\|$ Rn)*

   However, in this protocol, we cannot guarantee that the corresponding Running signal has occurred with Rn and H(IDk $\|$ Rn), provided we assume that the responder is honest: that *R* $\in$ *Honest.* The intruder can capture messages and modify them. This may result in a failed key agreement between two agents.

4.3 The hash based ID variation protocol

The hash-based ID variation protocol (Henrici & Muller) exchanges the ID as a tag's identification information on every session like the hash-chain protocol(Ohkubo et al.). This protocol is secure against replay attacker since the tag's ID is renewed by random number R and LT and LST are updated. LT means the last transaction number and LST means the last successful transaction number. Fig. 3 shows the process of the hash-based ID variation protocol. In message 2 and 3, these messages are the same as the hash unlocking protocol and R means the public identity of the Reader in the channel. Message 4 and 5 can be described in the same ways as message 2 and 3.

Message 1. R $->$ T : Query
Message 2. T $->$ R : (H(ID), H(LT (+) Id))%metaID, $\triangle$ LT
Message 3. R $->$ DB : metaID%(H(ID), H(LT (+) ID)), $\triangle$ LT
Message 4. DB $->$ R : R, (H(R (+) LT (+) ID))%metaID2
Message 5. R $->$ T : R, metaID2%(H(R (+) LT (+) ID))

Fig. 3. The Hash based ID Variation protocol

4.3.1 Protocol requirements and verification results

After running the FDR model checking tool, this protocol satisfies the first and second *Secret* requirement in Section 4.1.1 because it also does not use any key in the communication channel and the ID can be updated using △LT (LT = LST - LT). However this protocol does not satisfy the third requirements as below;

1. Agreement$_{AgreementSet}$(tr) $\widehat{=}$ R $\in$ Honest $\Rightarrow$ *signal.Running_Initiator.T.R.LT* precedes *signal.Commit_Responder.R.T.LT*
   The attackers can be authenticated when the attacker disguises the reader and receives H(ID),H(LT (+) ID),△ LT from the tag then sends them to the reader as a response before the tag performs the next authentication session. In this time, if the attackers don't transmit the information described in message 5 in Fig. 3, the tag classifies that the information as lost and the tag doesn't update its ID. Therefore the attackers can track the location of the tag since H(ID) is fixed, before the tag performs the next authentication session and updates its H(ID). The IDs of the back-end database and tag are updated on every session. Therefore this protocol isn't suitable for a ubiquitous computing environment with distributed databases.

4.4 Summary of possible attacks

We can summarize the verification results of the above protocols using model checking. We find that the previous protocols are vulnerable to the spoofing attack and replay attack and can be tracked by an attacker. The attacker performs the following attack.

1. Security against the spoofing attack : The attacker masquerades as the reader, then sends Query to the tag. The attacker gets the tag's response value due to not ensuring the response value of the hash function from this attack.
2. Security against the replay attack : After the reader transmits Query to the tag, the attacker eavesdrops on the response value from the tag.
3. Security against traffic analysis and tracking: To receive responses, the attacker masquerades as the reader then transmits a fixed Query and reads the tag or eavesdrops on the information sent between the reader and the tag. The attacker can therefore analyze the response from the tag.

5. The proposed security protocols for RFID system

The most threatening attacks are spoofing, replay attack, tracking and eavesdropping attacks, as these attacks affect all participants. To protect from these attacks, this section proposes two effective countermeasures.

5.1 Modified hash based protocol

Firstly, modified hash based protocol is the extended version of hash unlocking protocol using hash and exclusive-or algorithm, and can be used in the environment that requires lower burden of communication load with hand-held device reader.
We propose a modified protocol(Kim et al.a) (Fig.4) to ensure secure channel between DB and Reader, and Reader and Tag, using agent's nonce and exclusive-or technique in Casper, as follows;
In this protocol, we assume that the communication channel between Reader and DataBase is secure and the description of the protocol is as follows;

Message 1. T $->$ R : Tn, H(Rkey(+)Tn) % metaID
Message 2. R $->$ DB : Tn, metaID % H(Rkey(+)Tn)
Message 3. DB $->$ R : DBn, H(Rkey(+)DBn) % auth
Message 4. R $->$ T : DBn, auth % H(Rkey(+)DBn)
Message 5. T $->$ R : ID, H(ID)

Fig. 4. The modified hash based protocol for secure RFID systems

In message 1, we add *Tn* to metaID and use the exclusive-or(+)technique with Rkey, where it is originally stored in the hash-lock protocol and would be sent with nonce(Tn) to the Reader. In message 2, the Reader will forward metaID to DB with Tn to let the DB know who sent this message and to compare it with Tn in metaID. In message 3, DB sends another value *auth* to Reader including Rkey and DBn, using exclusive-or and hash function with DBn. This message is forwarded from Reader to Tag in message 4. Finally, the Tag unlocks itself after checking the Rkey when the Tag receives message 4 and sends the ID to the Reader.

5.1.1 Protocol analysis and verification result

In this chapter, the main ideas of our modified protocol to correct the problems in previous protocols are as follows;

1. To ensure the data privacy and freshness of tag's behavior over a number of requests from the reader and authentication between Tag and Reader, we introduce the *Tag's nonce(Tn) and DataBase's nonce(DBn)*.For this, a tag needs to have a Random Nonce Generator(PRN). Although there is literature indicating that a PRN needs greater computation capability, it is mandatory that there exists at most one PRN, to protect against replay and tracking attacks.
2. To ensure the confidentiality of data between agents, we add the *exclusive-or(+)technique* into this protocol.
3. To establish a secure channel between the reader and the tag, we introduce an *Auth* value, which consists of Rkey and DBn similar to metaID. This makes it possible for the protocol to be protected against spoofing attack.

After running the FDR tool, we confirm that our modified protocol overcomes the security weaknesses in hash-lock protocols and this protocol satisfies the Secret and Agreement requirements in whole Casper script and the testing result of the protocol can be described in FDR as below(Fig.5). The "$\surd$" marks in ahead of each statements show the satisfaction of the properties(i.e. two *Secrets* and *Auth1* : each statement means secrecy property and authentication property in Casper script) if the properties do not satisfy then the "X" mark would be shown.

5.2 Challenge response based protocol

Secondly, this protocol is based on the security algorithm employed in Yahalom protocol (Gong et al.) and can be used in the environment that user uses a more sophisticated secret to calculate the response to a challenge issued by the network requiring higher complex capability like fixed reader for warehousing and out of a ware-house of inventory systems.
The proposed protocol(Kim et al.b)(Fig.6) must guarantee the secrecy of the session key: in message 4 and 5, the value of the session key (Skey) must be known only by the participants playing the roles of Tag and Reader. Reader and Tag must be also properly authenticated to the DB. In this protocol, we use the *Server Key* and *Tag's Nonce(Tn)* to minimize the burden of

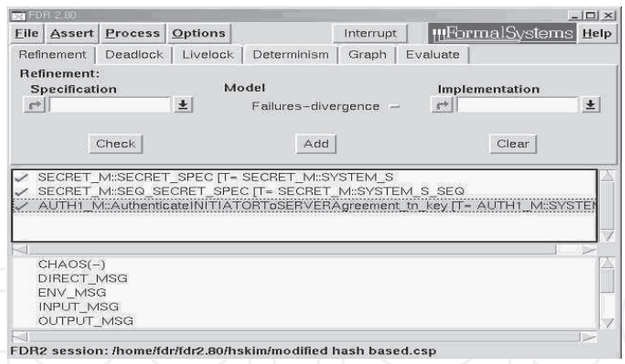


Fig. 5. The verification result of modified hash based protocol using FDR

Message 1. T $->$ R : Tn
Message 2. R $->$ DB : {T, Tn, Rn}{ServerKey(R)}
Message 3. DB $->$ R : $(\{R, Tn, Rn, ID, Skey\}\{ServerKey(T)\})$%metaID
Message 4. R $->$ T : metaID%({R, Tn, Rn, ID, Skey}{ServerKey(T)})
Message 5. DB $->$ R : {Skey}{ServerKey(R)}
Message 6. T $->$ R : {ID}{Skey}

Fig. 6. The challenge response based protocol for secure RFID systems

the Tag and to ensure authentication between Tag and Reader. The functions can be defined to take in an input parameter and return an output. It resembles a functional programming language in this aspect. The definition of a function called *ServerKey*, which takes in the name of an *Agent* and returns a *ServerKey*, could be given as shared : $Agent \rightarrow ServerKey$. In message 1, we design that Tag makes random nonce Tn and sends it to the Reader. This makes simple challenge-response easy. Therefore, in message 2, through *T, Tn, Reader's Nonce(Rn), and Server Key*, Reader can ensured authentication from DataBase. In message 3 and 4, DB encrypts all of the *R (Reader's identity), Tn, Rn, ID and Skey(Session key)* received from Reader and sends these to Reader. Then Reader forwards this metaID to the Tag for letting the Reader authenticate securely using a session key in message 6. In message 5, the DB also sends *Skey* to Reader for him to decrypt Tag's ID in message 6. In message 6, Tag can send his *ID* securely using Skey received in Message 4.

5.2.1 Protocol analysis and verification result

We describe the main ideas of our challenge-response protocol to correct the problems in previous protocols as follows;

1. Shifting all data except the ID to the backend : This is also recommended for data management. (i.e. the ID for the Tag existing at the backend database will be shifted to protect spoofing and eavesdropping attacks securely on the Tag through the database when the Reader sends a request. This means that the Tag originally does not have the an ID value).
2. Encoding data transfer : We support encryption of the data transmission to ensure authorized access to the data of concern and to protect against replay attack and tracking.
3. When a tag receives a "get challenge(query)" command from a Reader, it generates a random number Tn and sends it to the Reader. The Reader in turn generates a random number Rn and generates an encrypted data block that includes Tag's identity and Tn on

the basis of an encryption algorithm with *Serverkey(R)*. The data block is then returned to the database to authenticate the reader. Both Reader and Tag use the same encryption algorithm and since the server key is stored on the Tag, the Tag is capable of decrypting the *Serverkey(T)*. If the original random number Tn and the random number Tn in message 4, which has now been decrypted, are identical, then the authenticity of the Tag is ensured. Vis-a-vis the Reader has also been proved.

In addition, we verify the proposed authentication protocol based on a challenge-response authentication mechanism, which establishes a secure channel between Tag and Reader and confirms that our protocol satisfies the Secret and Agreement requirements in whole Casper script and the testing result of the protocol can be described in FDR as below(Fig.7).

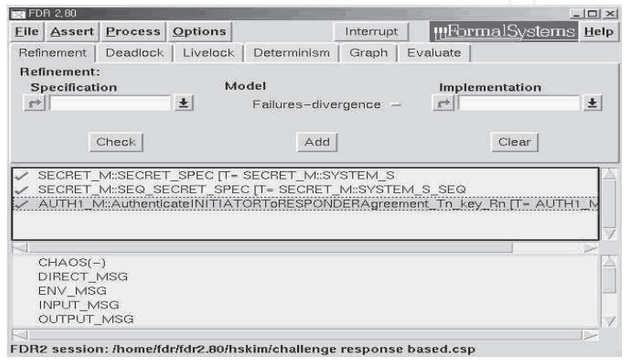


Fig. 7. The verification result of challenge-response protocol using FDR

6. Conclusions

Mobile and ubiquitous computing based on the RFID tag is defined as environments where users can receive network services for anytime and anywhere access through any device, with the tag connected via a wired and wireless network, to information appliances, including the PC. In this environment, there are many security threats that violate user privacy and interfere with services. It would be ideal if we could overcome RFID system's privacy and authentication threats by making minor modifications to the technology itself. Technical solutions have great appeal, implementation and testing costs are fixed and up-front, and once developed, the solutions can be directly integrated into the product. Further these solutions generally require little user education or regulatory enforcement.

Therefore, as an approach to solve the security problems using security protocol at the design level before implementation, we focus on safety analysis of the protocols and propose security protocols that can be widely researched in RFID systems using Casper and FDR. In verifying our protocols with the FDR tool, we were able to confirm that our protocols protect against some of the known security vulnerabilities that are likely to occur in RFID systems.

7. References

[Sarma et al.a] Sarma, S.; Weis, S. & Engels, D. (2003). RFID systems and security and privacy implications.*Proceedings of Workshop on Cryptographic Hardware and Embedded Systems(CHES 2002)*, pp. 454-469, LNCS No. 2523.

[EPCGLOBAL INC.] http://www.epcglobalinc.org.

[Gong et al.] Gong, L.; Needham, R. & Yahalom, R. (1990). Reasoning about Belief in Cryptographic Protocols. *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, pp. 18-36.

[Sarma et al.b3] Sarma, S. E.; Weis, S. A. & Engels, D. W.(2003). Radio-frequency-identification security risks and challenges. *Security Bytes*, Vol. 6(1).

[Henrici & Muller] Henrici, D. & Muller, P. (2004). Hash based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers. *Proceedings of PerSecaf04 at IEEE PerCom*, pp. 149-153.

[Juels] Juels, A. (2004). Minimalist cryptography for low-cost RFID tags, *Proceedings of the Fourth International Conf. on Security in Communication Networks*, LNCS, Springer-Verlag, September.

[Gildas] Gildas, A.(2005). Adversarial model for radio frequency identification.

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

[Hoare] Hoare, C.A.R.(1985). *Communicating Sequential Processes.* Prentice-Hall, Englewood Cliffs. NJ.

[Broadfoot & Roscoe] Broadfoot, P.J & Roscoe, A.W. (2002). Internalising Agents in CSP Protocol Models, *Proceedings of Workshop in Issues in the Theory of Security(WITS '02)*, Protland Oregon, USA.

[Dolev & Yao] Dolev, D and Yao, A.C. (1983). On the Security of Public-key Protocols. *Communications of the ACM*, 29(8), pp. 198-208

[Lowe] Lowe, G. (1997). Casper: A compiler for the analysis of security protocols. *Proceeding of the 1997 IEEE Computer Security Foundations Workshop X*, IEEE Computer Society. Silver Spring. MD, pp. 18-30.

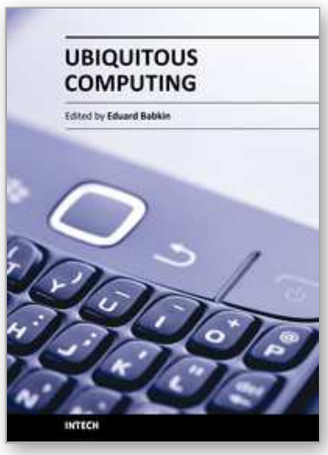
[FDR] *Formal Systems Ltd.* FDR2 User Manual. Aug, 1993.

[Ryan & Schneider] Ryan, P. Y. A.; Schneider, S. A. (2001). *Modelling and Analysis of Security Protocols: the CSP Approach.* Addison-Wesley.

[Ohkubo et al.] Ohkubo, M.; Suzuki, K. & Kinoshita, S. (2004). Hash-Chain Based Forward-Secure Privacy Protection Scheme for Low-Cost RFID. *Proceedings of the SCIS 2004.* pp. 719-724.

[Kim et al.a] Kim, H.S.; Oh, J.H. & Choi, J.Y. (2006). Analysis of the RFID Security Protocol for Secure Smart Home Network, *Proceedings of the International Conference on Hybrid Information Technology*, pp. 356-363.

[Kim et al.b] Kim, H.S.; Oh, J.H. & Choi, J.Y. (2006). Security Analysis of RFID Authentication for Pervasive Systems using Model Checking, *Proceedings of the thirtieth Annual International COMPSAC*, pp. 195-202.



Ubiquitous Computing

Edited by Prof. Eduard Babkin

ISBN 978-953-307-409-2
Hard cover, 248 pages
Publisher InTech
Published online 10, February, 2011
Published in print edition February, 2011

The aim of this book is to give a treatment of the actively developed domain of Ubiquitous computing. Originally proposed by Mark D. Weiser, the concept of Ubiquitous computing enables a real-time global sensing, context-aware informational retrieval, multi-modal interaction with the user and enhanced visualization capabilities. In effect, Ubiquitous computing environments give extremely new and futuristic abilities to look at and interact with our habitat at any time and from anywhere. In that domain, researchers are confronted with many foundational, technological and engineering issues which were not known before. Detailed cross-disciplinary coverage of these issues is really needed today for further progress and widening of application range. This book collects twelve original works of researchers from eleven countries, which are clustered into four sections: Foundations, Security and Privacy, Integration and Middleware, Practical Applications.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen