All rights are reserved and copyright of this manuscript belongs to the authors. This manuscript has been published without reviewing and editing as received from the authors: posting the manuscript to SCIS 2010 does not prevent future submissions to any journals or conferences with proceedings.

SCIS 2010 The 2010 Symposium on Cryptography and Information Security
Takamatsu, Japan, Jan. 19-22, 2010
The Institute of Electronics, Information and Communication Engineers

Scalable Grouping-proof Protocol for RFID Tags

Dang Nguyen Duc * Jangseong Kim * Kwangjo Kim *

Abstract— In this paper, we propose a grouping-proof protocol for RFID tags based on (n, n)-secret sharing. Our proposed protocol addresses the scalability issue of the previous protocols by removing the need for an RFID reader to relay messages from one tag to another tag. We also present a security model for a secure grouping-proof protocol which properly addresses the so called *mafia fraud attack* (sometimes called distance fraud) which is a simple relay attack suggested by Desmedt. Any location-based protocol including RFID protocols is vulnerable to this attack even if cryptographic technique is deployed. One practical countermeasure against mafia fraud attack is to integrate a distance-bounding protocol into a location-based protocol. However, Chandran *et al.* pointed out that forging geographic location of devices cannot be theoretically prevented. Therefore, we need to take this fact into account in order to make sense about security notion for secure grouping-proof protocols.

Keywords: RFID Security, Yoking-Proof, Grouping-Proof, Scalability

1 Introduction

Grouping-proof protocols allow multiple RFID tags to be scanned at once such that their co-existence is guaranteed. One typical application of a grouping-proof protocol is to scan tags that are supposed to stay *together*. For example, RFID tags attached on different parts of a car should be located near each other. Juels [3] proposed the first protocol of this kind which is called yoking-proof. The protocol allows an RFID reader to produce a co-existence proof of two RFID tags. The proof can then be verified by a verifier which holds all the secret keys of tags. Unfortunately, Saito and Sakurai [4] showed that yoking-proof is vulnerable to replay attack and proposed a timestamp-based version of yoking-proof to withstand the attack. A true grouping-proof protocol which supports simultaneous scanning of more than two tags was also proposed in [4]. However, this protocol requires a pallet tag to act as a proxy between a reader and other regular tags. Other improved variants of yoking-proof were also proposed in [5, 6, 7].

In this paper, we point out that all of the previous grouping-proof protocols in [3, 4, 5, 6, 7] suffer from a scalability problem. More specifically, a reader has to relay messages from one tag to another tag which makes it difficult to scan a large number of tags at the same time. Our proposed protocol aims to solve this problem by removing the requirement to relay messages among tags.

An important part of designing a secure protocol is to define a security model in which the term *secure* correctly captures our intuition about real-world security of the protocol. We argue that this task has not been done adequately in previous works. In particular, no previous work addresses *mafia fraud attack* presented in [1]. Mafia fraud attack is simply a relay attack where an attacker relays messages exchanged between a reader and tags. As noted in [9], all of grouping-proof protocols for RFID are inherently insecure against this attack because the attacker can relay messages exchange between a reader and tags that reside out of the communication range of the reader. The result is an invalid proof that contains tags not in the communication range of the reader at the time of interrogation. Indeed, a security model that does not address this issue cannot be a proper security model for grouping-proof protocols because it would be impossible to prove the security. In practice, we can somehow mitigate this attack by using a distance bounding protocol [2] so that a relay attacker does not have sufficient time to relay messages out of the communication range of the reader. Indeed, some of previous protocols [4, 6] make use of timestamp which is actually used to defeat replay attack. However, this prevention method works only if an interrogation session lasts as short as possible. Since a reader has to relay messages among tags in previous protocols, a protocol session can be prolonged which makes mafia fraud attack more feasible. Therefore, it is also important to solve the scalability problem in order to defeat mafia fraud attack. Note that, the use of timestamp does not mean that we do not need to take mafia attack into account when defining a security notion for secure grouping-proof protocols. After all, mafia fraud attack is always feasible from a theoretical point of view. In fact, in [8], Chandran *et al.* showed that it is impossible to securely verify the geographic location of a device. Another issue when defining a security model for grouping-proof protocol is that the verifier has no knowledge of what or how many tags are actually in the communication range of

* KAIST, 119 Munjiro, Yuseong-gu, Daejeon, Republic of Korea
{nguyenduc, jskim.withkals, kkj}@kaist.ac.kr

a reader. Therefore, we cannot achieve security at all if a reader is allowed to behave maliciously in an arbitrary way. For example, a reader can deliberately avoid scanning some tags resulting in an invalid co-existence proof. In this paper, we present a secure model for secure grouping-proof protocols which takes the above issues into account. In particular, we put the following assumptions in our security model:

- Relaying messages out of the communication range of a reader is not allowed. We address this assumption by restricting the adversary's access to the tag oracle during the last phase of an experiment in which the adversary interacts with a set of oracles, receives a challenge and attempts to solve the challenge. We shall discuss this in more details in Section 4.
- The reader is trusted to execute the protocol fruitfully but it may report an invalid co-existence proof to the verifier. In particular, before reporting a valid proof to the verifier, a dishonest reader may try to remove a tag from the proof, replace a tag in the proof with another tag or add another tag to the proof. In practice, the protocol can be implemented in a tamper-proof chip whereas a proof is assembled and sent to the verifier by the reader in software (and therefore is subject to malicious behaviors of a reader).

It is important to note that, none of previous protocol appears to be secure in a weaker assumption. We then propose a grouping-proof protocol for RFID by using a (n, n)-secret sharing scheme (also referred to as *unanimous consent control* in [10]). The goal of using a (n, n)-secret sharing scheme in our protocol is to let n tags sign n different challenges. The n challenges are n shared secrets of a number which is randomly chosen by the verifier. The threshold property of a (n, n)-secret sharing scheme guarantees that n signed challenges are *tied together*. We then prove the security of our protocol.

2 Related Work

In this section, we briefly review existing grouping-proof protocols. We will use the notations summarized in Table 2.

Table 1: Notations

Notation	Description
K_i	Secret key of tag $\mathcal{T}_i$
$\text{MAC}_K[.]$	Message authentication code with secret key K
P	A co-existence proof of multiple tags
$\mathcal{R}$	Reader
TS	Timestamp
$\mathcal{T}_i$	An RFID tag
$\mathcal{V}$	Verifier (Back-end Database)

2.1 Yoking-Proof for RFID Tags

Yoking-proof [3] enables an RFID reader to produce a proof that two RFID tags are present within the communication range of the reader. The proof can then be verified by the verifier which knows secret keys of the two tags. In the yoking-proof protocol, a tag proves its presence by signing a random number generated by another tag. A message authentication code (MAC for short) algorithm can be used as a signing mechanism. The reader is in charge of forwarding the random numbers and collecting the MACs to form a proof of co-existence. The resulting co-existence proof can be verified by checking the validity of the MACs. The protocol proceeds as follows:

1. $\mathcal{R} \rightarrow \mathcal{T}_1$: request.
2. $\mathcal{T}_1 \rightarrow \mathcal{R}$: $\mathcal{T}_1, r_1$ where r_1 is chosen at random.
3. $\mathcal{R} \rightarrow \mathcal{T}_2$: r_1.
4. $\mathcal{T}_2 \rightarrow \mathcal{R}$: $\mathcal{T}_2, r_2, m_2 = \text{MAC}_{K_2}[r_1]$ where r_2 is chosen at random.
5. $\mathcal{R} \rightarrow \mathcal{T}_1$: r_2.
6. $\mathcal{T}_1 \rightarrow \mathcal{R}$: $m_1 = \text{MAC}_{K_1}[r_2]$.
7. $\mathcal{R} \rightarrow \mathcal{V}$: $P = (\mathcal{T}_1, r_1, m_1, \mathcal{T}_2, r_2, m_2)$.

2.2 Saitoh-Sakurai's Grouping-Proof for RFID Tags

Saitoh and Sakurai [4] showed that yoking-proof is vulnerable to replay attack. The reason is that the two messages m_1 and m_2 are not guaranteed to be generated in the same session. As a result, an attacker can reuse m_2 in another session which results in a forged proof. To prevent the attack, Saitoh and Sakurai proposed a timestamp-based yoking-proof which requires an online verifier to issue a timestamp TS for each session. TS is included in each co-existence proof and must be signed by both $\mathcal{T}_1$ and $\mathcal{T}_2$. The online verifier accepts a proof only if it is received within the expected lifespan of one interrogation session. Other variants of yoking-proof also appeared in [5, 6, 7].

In [4], the authors also proposed another protocol which allows the simultaneous scanning of more than two tags. The protocol is called grouping-proof and requires an additional entity called pallet tag. The pallet tag has more computational resource than an RFID tag and acts as a representative of all RFID tags that are in the same package with the pallet tag.

1. $\mathcal{V} \rightarrow \mathcal{R}$: TS.
2. $\mathcal{R} \rightarrow \mathcal{T}_1, \mathcal{T}_1, \cdots \mathcal{T}_n$: TS.
3. $\mathcal{T}_i \rightarrow \mathcal{R}$: $m_i = \text{MAC}_{K_i}[\text{TS}]$, for $i = 1, 2, \cdots, n$.
4. $\mathcal{R} \rightarrow$ Pallet Tag: TS, $m_1, m_2, \cdots, m_n$.
5. Pallet Tag $\rightarrow \mathcal{R}$: $\text{C}_P = \text{SK}_K[\text{TS}, m_1, m_2, \cdots, m_n]$.
6. $\mathcal{R} \rightarrow \mathcal{V}$: $P = (\text{TS}, \text{C}_P, \mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n)$.

The proof P is subject to timestamp verification by the online verifier in order to prevent replay attacks. Then, the co-existence proof is verified by checking the validity of each m_i.

3 Scalability Issue of Previous Grouping-Proof Protocols

The design of yoking-proof and timestamp-based yoking proof suffers from a serious scalability issue. The reason is that a reader needs to relay messages from one tag to another so that a tag can sign the random numbers that were generated by the other tags. As a result, if the reader wants to produce a co-existence proof of n tags, it is required to relay $n(n-1)$ messages among n tags. The number of relaying messages can be reduced to $(n-1)$ if a proof is constructed in a chaining fashion. That is, the first tag signs the second tag's random number. The second tag signs the third tag's random number and so on. However, this approach might be subject to replay attack if a protocol is not designed carefully. Let's assume that a tag $\mathcal{T}_i$ appears in two chaining proofs. Using $\mathcal{T}_i$ as a connector, an attacker might try to connect the first half of the first proof with the second half of the second proof to produce a forged proof. Nevertheless, this is a significant communication overhead compared to the traditional method of scanning one tag at a time which requires no message to be relayed by the reader. This same problem also appears in other variations of yoking-proof including the previous works [5, 6, 7].

The grouping-proof protocol by Saitoh and Sakurai does not use the same design of yoking-proof. However, it requires a pallet tag which is capable of performing symmetric encryption. This increases the cost of multiple scanning of tags and might not be flexible in practice. For example, in a retail store, items that are scanned at a point-of-sale usually do not have an accompanying pallet tag. In addition, the reader still needs to relay messages from all tags to the pallet tag. In order to scan n tags at once, the reader needs to relay n messages to the pallet tag.

As we pointed out earlier, the lifespan of one protocol session may affect the resilience of a grouping-proof protocol against mafia fraud attack. We believe that it is important to solve the scalability problem of previous grouping-proof protocols, for the sake of not only performance but also security.

4 Security Model for A Secure Grouping-Proof Protocol

Before designing a protocol to secure certain cryptographic tasks, It is important that one should clearly define the meaning of the term *secure*. In this paper, we present a security model for a secure grouping-proof protocol for RFID tags which addresses mafia fraud attack and the level of trust on an RFID reader. We then define what a secure grouping-proof protocol for RFID tags is. Our security model is a conventional security model in a sense that the adversary is given access to a set of oracles and the term *secure* is defined via a game between a challenger and an adversary. In [7], the authors proposed another security model for secure grouping-proof protocol in the *Universal Composable Framework* (UC framework for short). A protocol which is secure in UC framework is guaranteed to remain secure even when running as a component of a large system. The most important part of a security model in the UC framework is the *ideal functionality* which is a trusted party implementing the required cryptographic task. The ideal functionality defined in [7] is called $\mathcal{F}_{group}$ which interacts with different involving parties via 5 interfaces: activate, initiate, link, complete and verify (whereas involving parties do not interact with each other directly). Interested readers are referred to [7] for the description of each of $\mathcal{F}_{group}$'s interface. The problem with $\mathcal{F}_{group}$ is that there is no condition for a tag to call $\mathcal{F}_{group}$'s initiate. Indeed, only tags within the communication range of a reader are qualified to make the initiate calls to $\mathcal{F}_{group}$. Unfortunately, the communication range of a reader is not modeled in $\mathcal{F}_{group}$. That is probably why the full security proofs for two protocols in [7] are not yet available. Note that, this does not mean security proofs for lightweight authentication protocols for RFID are invalid. In case of an authentication protocol, the goal of the adversary is to impersonate a tag. Simply relaying message between a legitimate tag and a reader does not imply impersonation. It is also worth mentioning that most of previous grouping-proof protocols employ timestamp which makes it difficult to rigorously analyze their security. We believe that it is better to avoid using a physical object in the description of a protocol but embed it into the security model or assumption.

We now describe our security model for a secure grouping-proof protocol. First of all, we realize that for a grouping-proof for RFID tags protocol, the primary goal of an adversary is to inject some tags (possibly genuine) into a valid co-existence proof while the tags are not actually in the communication range of the reader. In addition, the adversary might also want remove some tags from a valid co-existence proof. It is also assumed that the reader can behave maliciously but does execute the protocol correctly. When reporting a co-existence proof to the verifier, a malicious reader may try to replace some tags in the proof with different tags, add a tag to the proof or remove a tag from the proof. One can obtain a stronger security notion by allowing a malicious reader to deviate from the protocol in any fashion. However, it is impossible to achieve security because the verifier has no knowledge of what and how many tags are actually in the communication range of the reader. The malicious reader can violate the security by deliberately not scanning some tags. This issue also appears in all of the previous protocols in [3, 4, 5, 6, 7]. Indeed, the timestamp-chaining protocol by Lin *et al.* is vulnerable to malicious behaviors of a reader even if the reader is trusted to execute the protocol correctly. The reason is that before reporting a co-existence proof of n tags to the verifier, the malicious reader can remove some tags at the end of the timestamp chain from the proof without invalidating the proof. We now define a set of oracles that

provide information to the adversary:

- The reader(.) oracle: This oracle simulates a reader during a protocol session. That it, it returns the reader's challenge to a tag.
- The corrupt-reader(.) oracle: This oracle corrupts a reader and returns the current state of the reader. The adversary is also allowed to control the reader after this oracle is called.
- The tag(.) oracle: This oracle simulates a tag during a protocol session. That is, it returns the tag's response given a challenge from a reader.
- The verify(.) oracle: This oracle takes a co-existence proof P as input and returns 1 if P is valid and 0 otherwise.

We now define the security notion for a secure grouping-proof protocol via the following game between a challenger and an adversary.

1. The challenger first sets the verifier and a reader and tags up to prepare for the game.
2. In the first phase of the game, the adversary collects information via 4 oracles: reader(.), tag(.), corrupt-reader(.) and verify(.). These oracles are simulated by the challenger.
3. In the second phase of the game, the challenger gives the adversary a valid proof P of n tags as a challenge. The adversary's goal is to either remove a tag from P or add a new tag to P or replace a tag in P with a different one. In this phase, the adversary is also given access to the corrupt-reader(.) oracle after the challenge proof P is constructed. However, the tag(.) oracle is not provided to the adversary after the adversary has seen P. This is to reflect our assumption that relay attack is not possible. The adversary should output a new proof P' which satisfies one of its goals.
4. The adversary wins the game if verify(P') returns 1. That is, P' is a valid co-existence proof.

Definition 1. *A grouping-proof protocol is said to be secure if the winning probability of the adversary in the above game is negligible. That is, for any polynomial-bounded adversary $\mathcal{A}$ and a sufficiently large security parameter k.*

$$\mathbf{Prob}[\mathcal{A}\ wins] < \frac{1}{poly(k)}$$

5 Our Proposed Grouping-Proof Protocol

We now propose our grouping-proof protocol for multiple RFID tags which does not suffer from the scalability problem. We use a (n, n)-secret sharing scheme to stop messages being relayed among tags. A (n, n)-secret sharing scheme allows one to *split one secret x* into n of so called shared secrets such that x can only be reconstructed from the shared secrets if and only if all of n shared secrets are provided. This property is used in our proposed protocol so that each tag can sign its own random number to prove its existence. The random numbers are shared secrets generated by a secret sharing scheme. If the original secret generated by a verifier can be recovered from signed shared secrets that were backscattered by tags, then the proof of co-existence of tags is verified. A (n, n)-secret sharing scheme can be implemented as follows:

- Given a secret x, a dealer chooses $(n-1)$ random numbers $y_1, y_2, \cdots, y_{n-1}$ as the first $(n-1)$ shared secrets.
- The last shared secret y_n is computed by $y_n = x \oplus y_1 \oplus y_2 \oplus \cdots \oplus y_{n-1}$.

It is easy to see that the above protocol achieves perfect security since it is impossible to recover x without any of $y_1, y_2, \cdots$ or y_n. In addition, for each randomly chosen x, a shared secret of x is also random. This property is important to prevent replay attack as a shared secret is used as a challenge in our proposed protocol. We now describe our grouping-proof protocol below.

1. $\mathcal{V} \rightarrow \mathcal{R}$: x chosen at random. The verifier also sets a time-to-live on x such that a co-existence proof associated with x must be received within the lifespan of x (which is approximately the time taken by one interrogation session of a reader).
2. $\mathcal{R} \rightarrow \mathcal{T}_i$: x, y_i for $i = 1, 2, \cdots, n$ where $y_1, y_2, \cdots,$ and y_n are n shared secrets of x.
3. $\mathcal{T}_i \rightarrow \mathcal{R}$: $\mathcal{T}_i, m_i = \text{MAC}_{K_i}[y_i, x]$, for $i = 1, 2, \cdots, n$.
4. $\mathcal{R} \rightarrow \mathcal{V}$: $P = (\mathcal{T}_1, y_1, m_1, \mathcal{T}_2, y_2, m_2, \cdots, \mathcal{T}_n, y_n, m_n)$.
5. $\mathcal{V}$: The verifier verifies a proof P by checking if P is received within the lifespan of $x = y_1 \oplus y_2 \oplus \cdots \oplus y_n$ and each m_i is valid MAC of the tag $\mathcal{T}_i$ on (x, y_i).

Remark 1. *Note that, it is important to stress that we do use timestamp in our protocol to prevent a malicious reader from abusing x (i.e., the malicious reader can take x and use shared secrets of x on different tags at different locations and times). However, the way which timestamp is used in our protocol is very different from in previous protocols. More specifically, we*

do not use timestamp as a challenge to a tag. Instead, only the verifier maintains timestamp for each interrogation session. This allows us to leave "time-to-live of x" to the security model. Indeed, the fact that a co-existence proof must be received within the lifespan of x fits in the assumption that a reader always executes the protocol correctly until reporting a proof to the verifier. Therefore, we can ignore the use of timestamp in the security proof of our protocol.

We now analyze the success probability of an adversary attacking our protocol. The probability is measured in terms of the success probabilities of adversaries attacking the underlying MAC and secret sharing schemes in the following theorem.

Theorem 1. *Let α be success probability of an adversary attacking the underlying MAC scheme. Let ϵ be the success probability of an adversary that attacks our proposed grouping-proof protocol, we have:*

$$\epsilon = O\left(\alpha + 2^{-\frac{l}{2}}\right)$$

where l is the bit length x and d is the number of tags in the tag database.

Proof. Let $\mathcal{A}$ be the adversary that attacks our proposed grouping-proof protocol. Given a challenge $P = (\mathcal{T}_1, y_1, m_1, \mathcal{T}_2, y_2, m_2, \cdots, \mathcal{T}_n, y_n, m_n)$ and let $x = y_1 \oplus y_2 \oplus \cdots \oplus y_n$, $\mathcal{A}$ wants to achieve one of the following goals:

- Construct a co-existence proof $P' = (\mathcal{T}_1^*, y_1^*, m_1^*, \mathcal{T}_2^*, y_2^*, m_2^*, \cdots, \mathcal{T}_n^*, y_n^*, m_n^*)$ such that $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\} \neq \{\mathcal{T}_1^*, \mathcal{T}_2^*, \cdots, \mathcal{T}_n^*\}$; $y_1^* \oplus y_2^* \oplus \cdots \oplus y_n^* = x$; and $m_1^*, m_2^*, \cdots$ and m_n^* are valid MACs of $\mathcal{T}_1^*, \mathcal{T}_2^*, \cdots$ and $\mathcal{T}_n^*$ on $(y_1^*, x), (y_2^*, x), \cdots$ and (y_n^*, x), respectively. In other words, $\mathcal{A}$ succeeds when it can replace at least one tag that is actually in the communication range of the reader by another tag. We call this type of adversary Type-I adversary.

- Construct a co-existence proof $P' = (\mathcal{T}_1^*, y_1^*, m_1^*, \mathcal{T}_2^*, y_2^*, m_2^*, \cdots, \mathcal{T}_{n-1}^*, y_{n-1}^*, m_{n-1}^*)$ such that the cardinality of $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\} \setminus \{\mathcal{T}_1^*, \mathcal{T}_2^*, \cdots, \mathcal{T}_{n-1}^*\}$ is 1; $y_1^* \oplus y_2^* \oplus \cdots \oplus y_{n-1}^* = x$; and $m_1^*, m_2^*, \cdots$ and m_{n-1}^* are valid MACs of $\mathcal{T}_1^*, \mathcal{T}_2^*, \cdots$ and $\mathcal{T}_{n-1}^*$ on $(y_1^*, x), (y_2^*, x), \cdots$ and (y_{n-1}^*, x), respectively. In other words, the adversary can remove a tag from P. We call this type of adversary Type-II adversary.

- Construct a co-existence proof $P' = (\mathcal{T}_1, y_1^*, m_1^*, \mathcal{T}_2, y_2^*, m_2^*, \cdots, \mathcal{T}_n, y_n^*, m_n^*, \mathcal{T}_{n+1}^*, y_{n+1}^*, m_{n+1}^*)$ such $y_1^* \oplus y_2^* \oplus \cdots \oplus y_n^* \oplus y_{n+1}^* = x$; and $m_1^*, m_2^*, \cdots, m_n^*$ and m_{n+1}^* are valid MACs of $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n$ and $\mathcal{T}_{n+1}^*$ on $(y_1^*, x), (y_2^*, x), \cdots, (y_n^*, x)$ and (y_{n+1}^*, x), respectively. In other words, the adversary can add the tag $\mathcal{T}_{n+1}^*$ to P. We call this type of adversary Type-III adversary.

In the first phase of the attack, $\mathcal{A}$ are given access to three oracles: the tag(.) oracle, the reader(.) oracle and the verify(.) oracle. The corrupt-reader(.) oracle is not required as $\mathcal{A}$ can eavesdrop x itself from challenges sent to tags (except that $\mathcal{A}$ can control the reader after seeing the challenge P, however this does not affect the analysis here). The tag(.) oracle is essentially a MAC oracle as it outputs MAC on an input value together with a tag ID. In the second phase of the attack, the adversary can only control the reader after seeing the challenge P. No oracle access is given in this phase. As usual, we limit the number of calls to oracles and running time of the adversary to be polynomial in security parameters. We analyze the success probability of each type of the adversary below.

Type-I Adversary: We distinguish two cases of Type-I adversary as follows:

- **Case 1**: none of (y_i^*, x) for $i = 1, 2, \cdots, n$ has not been asked to the tag(.) oracle. In this case, $\mathcal{A}$ is essentially a MAC forger with m_i^* is a forged MAC. Indeed, if $\mathcal{A}$ can forge a MAC, then it is obvious to attack the proposed grouping-proof protocol by constructing a forged MAC on one of (y_i, x) for $i = 1, 2, \cdots, n$ such that the forged MAC is a valid MAC of a tag not in $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}$. Therefore, the success probability of $\mathcal{A}$ is bounded by the success probability of the MAC adversary.

- **Case 2**: at least one of (y_i^*, x) for $i = 1, 2, \cdots, n$ has been asked to the tag oracle. We only consider the case that the adversary try to replace one tag in P with another tag. But it can be easily generalized to the case of replacing more than one tag. Since $\mathcal{A}$ is not supposed to forge a MAC (otherwise, it is easier to attack by executing the scenario of the adversary in the first case) and the tag(.) oracle is not provided in the second phase of the attack, $\mathcal{A}$ can only hope that its query to the tag oracle with (y_i^*, x) results in $(\mathcal{T}_i^*, m_i^*)$ such that $\mathcal{T}_i^*$ is not among $(\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n)$ and y_i^* constitutes a valid shared secret. However, because the underlying (n, n)-secret sharing scheme is perfectly secure and x is randomly chosen for each session, y_i^* has to be one of $y_1, y_2, \cdots, y_n$. In other words, $\mathcal{A}$ succeeds only if one of the pairs (y_i, x) for $i = 1, 2, \cdots, n$ has been queried to the tag(.) oracle in the querying phase such that the returned tuple $(\mathcal{T}_i^*, m_i^*)$ satisfies the adversary's goal. As shared secrets are randomly distributed and there are $(d - n)$ candidate tags for $\mathcal{T}^*$, the success probability of $\mathcal{A}$ is $\frac{d-n}{d} 2^{-\frac{l}{2}}$.

Type-II Adversary: Using the same analysis for Type-I adversary, we can see that the best option that adversary can succeed is to forge a MAC. For example, if the adversary wants to remove $\mathcal{T}_n$ from P, it can forge

a MAC of $\mathcal{T}_{n-1}$ on $(y_{n-1} \oplus y_n, x)$. The resulting proof P' is $(\mathcal{T}_1, y_1, m_1, \mathcal{T}_2, y_2, m_2, \cdots, \mathcal{T}_{n-1}, y^*_{n-1}, m^*_{n-1})$ where $y^*_{n-1} = y_{n-1} \oplus y_n$ and m^*_{n-1} is the forged MAC. Otherwise, the adversary would have to hope that $(y_{n-1} \oplus y_n, x)$ was queried to the $\mathsf{tag}(.)$ oracle during the querying phase. To conclude, the success probability of Type-II adversary is bounded by $\alpha + \frac{n}{d}2^{-\frac{l}{2}}$.

Type-III Adversary: The success probability of Type-III adversary can also be analyzed similarly. In particular, if the adversary can forge a MAC, he can add a tag $\mathcal{T}^*_{n+1}$ to P by forging two MACs of $\mathcal{T}_n$ and $\mathcal{T}^*_{n+1}$ on (y^*_n, x) and (y^*_{n+1}, x), respectively, such that $y^*_n \oplus y^*_{n+1} = y_n$. The forged proof P' is $(\mathcal{T}_1, y_1, m_1, \mathcal{T}_2, y_2, m_2, \cdots, \mathcal{T}_n, y^*_n, m^*_n, \mathcal{T}^*_{n+1}, y^*_{n+1}, m^*_{n+1})$ which should be correctly verified by the verifier. Therefore, we can obtain the success probability of Type-III adversary as $\frac{1}{2}\alpha + \frac{d-n}{d}2^{-\frac{l}{2}}$.

Combining the success probabilities of three types of the adversary, we complete the proof. □

Theorem 1 suggests that if the underlying MAC scheme is secure, *i.e.,* α is negligible, and l is long enough, then the success probability of an adversary attacking our proposed grouping-proof for RFID tags protocol is negligible. We conclude that our scheme is secure.

Assuming that we want to produce a co-existence proof of n tags, We compare our proposed scheme with previous protocols in terms of performance and security in Table 5.

Table 2: Comparison

Protocol	Number of Relaying Messages	Cost of Generating Proof
Yoking-Proof	$n(n-1)$	$2n(n-1)$ MACs
2nd Protocol [4]	n	n MACs 1 Encryption
Protocol in [5]	$n(n-1)$	$2n(n-1)$ MACs
1st Protocol [6]	$n(n-1)$	$2n(n-1)$ MACs 1 Encryption
1st Protocol in [7]	$n(n-1)$	4n(n-1) $f(.)$ Evaluations
Proposed Protocol	0	n MACs

6 Conclusion

In this paper, we present a grouping-proof for RFID tags protocol based on secret sharing. Our protocol solves the scalability issue of previous protocols by avoiding relaying messages among tags by a reader. We also define a security model for a secure grouping-proof protocol. Our security model deals with the case of un-trusted readers in a proper way. In particular, we cannot assume that a reader is totally un-trusted. Instead, we assume that a reader is trusted to execute a grouping-proof protocol correctly but may behave maliciously when reporting a co-existence proof of tags to the verifier. We also address the impact of mafia fraud attack on the security of a grouping-proof protocol. Finally, we show that our proposed grouping-proof protocol satisfies the security notion defined within the proposed security model.

References

[1] Yvo Desmedt, *Major security problems with the Unforgeable (Feige)-Fiat-Shamir proofs of identiy and how to overcome them*, In SecureCom'88, pp. 15-17, 1988.

[2] Stefan Brands and David Chaum, *Distance-Bounding Protocols*, In Advances in Cryptology EUROCRYPT'93, Volume 765 of Lecture Notes in Computer Science, Springer-Verlag, pp. 344-359, 1994.

[3] Ari Juels, *Yoking-Proofs for RFID Tags*, In the Proceedings of First International Workshop on Pervasive Computing and Communication Security, IEEE Press, pp.138-143, 2004.

[4] Junichiro Saitoh and Kouichi Sakurai, *Grouping-Proofs for RFID Tags*, In the Proceedings of AINA International Conference, IEEE Computer Society, pp. 621-624, 2005.

[5] Selwyn Piramuthu, *On Existence Proofs for Multiple RFID Tags*, In the Proceedings of ACS/IEEE International Conference on Pervasive Services, IEEE Computer Society, pp. 317-320, 2006.

[6] Chih-Chung Lin, Yuan-Cheng Lai, J. D. Tygar, Chuan-Kai Yang and Chi-Lung Chiang, *Co-existence Proof using Chain of Timestamps for Multiple RFID Tags*, In the Proceedings of AP-Web/WAIM International Workshop, Springer-Verlag LNCS 5189, pp. 634-643, 2007.

[7] Mike Burmester, Breno de Medeiros, and Rossana Motta, *Provably Secure Grouping-Proofs for RFID Tags*, In the Proceedings of CARDIS International Conference, Springer-Verlag LNCS 5189, pp. 176-190, 2008.

[8] Nishanth Chandran, Vipul Goyal, Ryan Moriarty and RafailOstrovsky, *Position Based Cryptography*, In the Proceedings of CRYPTO'09, Springer-Verlag LNCS 5677, pp. 391-407, 2009.

[9] Dang Nguyen Duc and Kwangjo Kim, *On the Security of RFID Group Scanning Protocols*, IEICE Transactions on Information and Communication Systems, Vol. E93-D, No. 3, Mar. 2010.

[10] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of Applied Cryptography*, Available at `http://www.cacr.math.uwaterloo.ca/hac/`, pp. 524-525.