Privacy-Preserving Authorized RFID Authentication Protocols

Nan Li[1(✉)], Yi Mu[1], Willy Susilo[1], Fuchun Guo[1], and Vijay Varadharajan[2]

[1] Centre for Computer and Information Security Research,
School of Computer Science and Software Engineering,
University of Wollongong, Wollongong, Australia
{nl864,ymu,wsusilo,fuchun}@uow.edu.au

[2] Faculty of Science, Information and Networked Systems Security Research,
Department of Computing, Macquarie University, Sydney, Australia
vijay.varadharajan@mq.edu.au

Abstract. Radio Frequency Identification (RFID) has been widely ad-opted for object identification. An RFID system comprises three essential components, namely RFID tags, readers and a backend server. Conventionally, the system is considered to be controlled by a single party who maintains all the secret information. However, in some practical scenarios, RFID tags, readers and servers could be operated by different parties. Although the private information should not be shared, the system should allow a valid tag to be authenticated by a legal reader. The challenge in designing the system is preserving the tag and reader's privacy. In this paper, we propose a novel concept of *authorized RFID authentication*. The proposed protocols allow the tag to be merely identifiable by an authorized reader and the server cannot reveal the tag during the reader-server interaction. We provide a formal definition of privacy and security models of authorized authentication protocols under the strong and weak notions and propose three provably secure protocols.

1 Introduction

A Radio Frequency Identification (RFID) system comprises three components: RFID tags, RFID readers and a backend server. An RFID tag is associated with a unique identifier which is allocated by the backend server. The typical RFID system is established by a single party who initiates the secret keys. To identify a tag, a reader communicates with the tag and sends the tag's response to the backend server. The server checks the tag's identity by using the shared keys and informs the reader whether the tag is valid.

Many RFID authentication protocols [13,14,23,24] have been proposed to preserve the tag privacy in conventional systems. These protocols assume that a reader and a server are held by a single entity. However, in some practical scenarios, we found that tag, reader and server are relatively independent, and

This work is supported by the Australian Research Council Discovery Project DP110101951.

N. Saxena and A.-R. Sadeghi (Eds.): RFIDSec 2014, LNCS 8651, pp. 108–122, 2014.
DOI: 10.1007/978-3-319-13066-8_7

hence, the existing solutions of RFID authentication protocols are deemed to be impractical. Consider the following scenario.

In an priviledged membership club, there are sole facilities provided for their members exclusively, such as restaurant, massage and sauna. Each of these facilities is operated by different business owners, who are paid by the owner of the club, who is also taking membership fees from its members. Hence, these facilities will allow exclusive club members only to access them and enjoy the service provided. In order to provide this benefit to the members, the club issues a membership card that is used to identify each member's identity. Nevertheless, to ensure the privacy of each member, the member would like to ensure that his/her identity will remain private whenever he/she is enjoying those services. Otherwise, these facilities will not be attractive to the members, if they have to sacrifice their privacy to trade for the facilities offered. In addition, the facilities are also expected to prevent the sensitive customer information from being exposed to the club, even though the members are indeed paying the membership fee to the club. The current solution may sound feasible to be implemented with an RFID system. Nevertheless, the requirement to maintain both privacy and accountability at the same time is seemingly contradictory.

The challenge in designing authentication protocols for the above scenario is the tag and reader's privacy. A strong tag privacy prevents a tag being linked in two different sessions even if the tag is completely corrupted. Most previous protocols consider the tag untraceability under the assumption that the server is honest and the reader can authenticate all the tags. However, it is suitable to our scenario where the server and the reader are relatively independent. The adversary who plays as an authorized reader can attempt to disclose a tag which is not intended to be identifiable. The reader's privacy is considered as whether the backend server can reveal the tag's identity during the protocol run. Specifically, the tag is merely identifiable by the authorized reader rather than the server; otherwise the server can obtain the merchant's (reader) client information and trace the tag. Unfortunately, to the best of our knowledge, existing protocols ignore this requirement and there is no protocol that cater the reader's privacy. Therefore, we need new models to evaluate a protocol's privacy and a novel protocol is desired.

Tag impersonation is one of crucial security problems of authentication protocols. Normally, it is hard to resist this attack if the tag is compromised. However, in our system, the untrusted server can cheat the reader without corrupting the tag by using the tag's shared secret. Hence, the protocol needs to prevent abuse of the shared information by the server.

Our Contributions. We introduce a novel notion of authorized RFID authentication (ARA, for short) protocols. In an ARA protocol, a reader is required to be authorized prior to identifying a tag and the server is blind regarding the tag which is being identified. The exiting protocols allow the server to disclose the tag's identity, which is not desirable for systems which require strong privacy protection. In this paper, provide three constructions. First, we propose a concrete construction based on the symmetric key cryptography. The protocol provides a weak privacy

while the tag only needs to perform hash computations. To improve the privacy, we provide two protocols which achieve the strong privacy. The second protocol provides constant authentication time on the reader, while the communication cost is dependent on the number of authorized tags. The third protocol supports the constant communication cost, while it requires exhaustive key search.

We discuss the privacy and security requirements of ARA protocols. Firstly, a reader is only allowed to authenticate a specified group of tags which are currently authorized. It indicates the tag's forward privacy and backward privacy. These two notions are different from the traditional definitions. We give the definition in Sect. 4.2. Then, the security of a tag is considered such that the server cannot forge a tag unless it corrupts the tag. According to the proposed threat model, the privacy and security models are classified in strong and weak. We prove that our proposed ARA protocols are secure.

Related Work. Vaudenay [26] proposed a strong privacy model which is considered as the most complete one. The privacy of an RFID tag authentication protocol is classified in several levels which are strong, destructive, forward and weak. Each level is with respect to a different adversary with a set of oracle calls. A strong adversary is allowed to corrupt a tag and continues future interactions with the compromised tag.

Another strong privacy model was introduced by Juels and Weis [15]. The model is based on the IND-CCA experiment and the adversary of the experiment aims to distinguish two different tags. Later, Hermans, Pashalidis, Vercautern and Preneel [12] proposed a new practical RFID privacy model. They defined the "left" and "right" world that an adversary needs to decide which world is simulated in the experiment. Many other RFID privacy models (e.g., [5–7,20]) are also presented in the literature.

Nithyanand, Tsudik and Uzun [21,22] considered the reader revocation problem in the public key infrastructure based RFID system. This problem is prominent as the (passive) tag could not check the time information during the protocol execution. The proposed solution requires a tag to equip a date display and a user checks during the certificate verification.

The elliptic curve cryptography (ECC) based RFID authentication protocols are acceptable by low-cost RFID tags [11,19]. Many ECC based RFID authentication protocols [2,16–18,25] were proposed. The main purpose of the ECC based protocol is to provide the strong privacy. However, most of existing schemes have been unfortunately broken later in [4,8–10,16].

2 System Model

In this section, we describe the entities of the ARA system and the formal definition of ARA protocols. The system defines the following entities: Tags, Readers and Servers.

– **Tag** T_i: Has a small storage and is not temper-resistant. It stores the keys in a non-volatile memory and requires capabilities to perform hash computations

and ECC computations depends on the protocols. It can be considered as a membership card held by the member who initiates the tag's secret key.
- **Reader** R_i: A powerful device which is authorized by a server to authenticate a group of tags with the given period key. R_i is controlled by a merchant who has an individual backend server.
- **Server** S_i: S_i provides the membership registration for customers and aids the reader to authenticate a tag. The server can authorize the reader to authenticate a group of tags and revoke the reader when it is no longer qualified.

The ARA protocol is executed by tag, reader and server. In the system, a server creates a tag and publishes a set of public information, such as the public key of the server. The member initiates the tag with the server's information and the keys which are chosen by himself. The public key of the tag is given to the server when the card is activated, while the private key is unknown to the server. To authorize a reader, the server generates a period key for the reader. During the tag authentication, the reader needs to cooperate with the server. However, the server cannot discover the identity of the tag which is involved in the session. To revoke a reader, the server can let the reader's period key be expired.

Our protocol consists of four algorithms: server key generation (ServerKeyGen), tag key generation (TagKeyGen), reader authorization (ReaderAuth) and tag authentication (Auth). The definition of algorithms are depicted as follows.

- $\mathsf{ServerKeyGen}(k) \rightarrow (PK, SK)$: Taking as input a security parameter k, it generates the server's public/private key pair (PK, SK).
- $\mathsf{TagKeyGen}(T, k) \rightarrow (pk, sk)$: Taking as input a security parameter k for the tag T, it outputs T's public key pk and private key sk.
- $\mathsf{ReaderAuth}(\{pk_i\}, \mathbb{T}_R, sk, R) \rightarrow (rsk, rpk)$: Taking as input a set of public keys $\{pk_i\}$ of tags $\mathbb{T}_R$, the server's private key SK and a reader R, it outputs a secret rsk and the reader's period key rpk. rpk is given to the reader and rsk is given to the server. For each run of this algorithm, the reader's current keys are revoked.
- $\mathsf{Auth}(sk, PK, rsk, rpk) \rightarrow \{T, \perp\}$: The tag takes as input a private key sk and a server's public key PK, a reader takes as input a period key rpk and the server takes as input a secret rsk, it outputs T if the tag is authenticated, $\perp$ otherwise.

3 Proposed Protocols

The concrete constructions of proposed ARA protocols are presented in this section. The protocol in Sect. 3.1 is based on symmetric key cryptography and it achieves basic requirements of ARA protocols. Section 3.2 shows the drawbacks of protocol 1 and describe an ECC-based solution which the server handles most computations of the protocol execution. Optionally, Sect. 3.3 introduces a protocol which only requires constant communication cost during the authentication and provides the false output detection. As an overview, Table 1 summarizes

the security and privacy properties of three protocols along with communication cost, computational efficiency and tag capabilities.

We define three cryptographic hash functions H_1, H_2, H_3, where $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}$ and employ the pairing group $(g, h, p, \hat{e}, \mathbb{G}, \mathbb{G}_T)$. $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups of the same prime order p. g, h are two generators of group $\mathbb{G}$. The map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a symmetric bilinear mapping.

Table 1. Comparison of proposed protocols and tag capabilities. $\surd$: the protocol achieves this property; $\times$: the protocol cannot provide this property; ⧫: the protocol achieves this property without tag corruption operations; H: requires hash computations; PK: requires ECC computations. Note that tag unforgeability is against a malicious server who cannot corrupt tags.

	Forward Privacy	Backward Privacy	Reader Privacy	Tag Unforgeability	Constant Com. Cost	Constant Reader Auth.	Tag Cap.
P1	⧫	⧫	$\surd$	$\times$	$\times$	$\surd$	H
P2	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	H, PK
P3	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	H, PK

3.1 Protocol 1

Our proposed protocol 1 is based on the symmetric key cryptography. It only requires a tag to compute hash values. The protocol achieves basic privacy requirements of ARA protocols with a relaxed condition. We analyze the privacy in Appendix A. The protocol is presented in Fig. 1.

- ServerKeyGen: The server generates a key space $\mathcal{K}$.
- TagKeyGen: The member randomly chooses $x \in \mathcal{K}$ and sets $(pk, sk) = (\cdot, x)$. The secret key sk is stored in the tag and given to the server.
- ReaderAuth: To authorize the reader R to identify a specified set of tags $\mathbb{T}_R$, the server randomly chooses $\gamma \in \{0,1\}^l$, and sets $(rpk, rsk) = (\gamma, \gamma)$.
- Auth:To authenticate a tag, the tag, reader and server interact as follows
  1. The reader randomly chooses $s \in \{0,1\}^l$ and send (s, γ) to the tag.
  2. Upon receiving (s, γ), the tag selects $a \in \{0,1\}^l$ and sends the reader the response (a, C), where $C = H_1(x, a, s, \gamma)$.
  3. Upon receiving the tag's response C, the reader sends (a, s) to the server.
  4. Upon receiving (a, s), the server retrieves $(\mathbb{T}_R, \gamma)$. For each $T_i \in \mathbb{T}_R$, the server computes $C' = H_1(x_i, a, s, \gamma)$ then sends the reader a set $\{(T_i, C_i') : T_i \in \mathbb{T}_R\}$.
  5. Finally, the reader outputs T_i if $C \in \{(T_i, C_i') : T_i \in \mathbb{T}_R\}$.

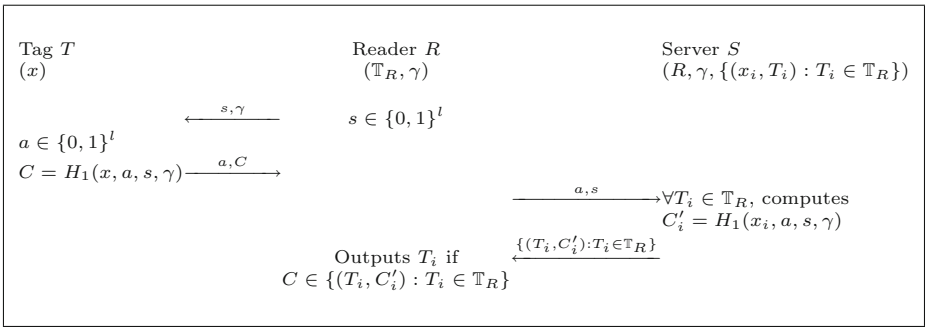

Fig. 1. Authorized RFID authentication protocol 1.

3.2 Protocol 2

In ARA system model, tag, reader and server are relatively independent. The key of a tag is expected to be unknown by the server since the server could abuse the key to forge the tag. It is difficult to prevent forging a tag by the server from using symmetric key based protocols. A trivial solution may be that the tag sends a signed nonce to the reader. However, in this case, the tag's response is publicly verifiable that an adversary can identify the tag by exhaustive public key search. Thus, tag's identity needs to be concealed and only an authorized reader is entitled to reveal. We then present Protocol 2 with ECC to tackle this issue. The protocol is presented in Fig. 2.

- ServerKeyGen: The server randomly picks $\alpha \in \mathbb{Z}_p^*$, and sets the public/private key pair $(PK, SK) = (g^\alpha, \alpha)$.
- TagKeyGen: The member randomly chooses $x \in \mathbb{Z}_p^*$, and computes the tag's public and private keys $(pk, sk) = (g^x, x)$. (g, sk, pk, PK) are stored in the tag and pk is given to the server.
- ReaderAuth: To authorize the reader R to identify a specified set of tags $\mathbb{T}_R$, the server randomly chooses $\gamma \in \mathbb{Z}_p^*$, and sets the reader's period $rpk = \{\gamma, (T_i, g^{x_i}) : T_i \in \mathbb{T}_R\}$ and secret key $rsk = (\gamma, \alpha)$. The server stores $(R, rsk, \mathbb{T}_R)$ and sends rpk to the reader R.
- Auth: To authenticate a tag, the tag, reader and server communicate as follows.
  1. The reader randomly selects $B \in \mathbb{G}$ and sends (B, γ) to the tag.
  2. Upon receiving (B, γ) from the reader, the tag randomly chooses $r \in \mathbb{Z}_p^*$, and computes (w, s, C_1, C_2, C_3). It sends (C_1, C_2, C_3) as the response to the reader. Note that C_1 is to assist reader identify a tag and hide the value s, otherwise the tag's response is publicly verifiable.
  3. Upon receiving (C_1, C_2, C_3), the reader forwards (B, C_3) to the server.
  4. Upon receiving the message (B, C_3) from the reader R, for each $T_i \in \mathbb{T}_R$, the server computes (w', V_i, U_i). Then the server replies $\{(T_i, V_i, U_i) : T_i \in \mathbb{T}_R\}$.
  5. Finally, the reader outputs T_i if $C_1 = U_i$ and $\hat{e}(C_2, g^{x_i + V_i}) = \hat{e}(g, g)$, otherwise rejects.

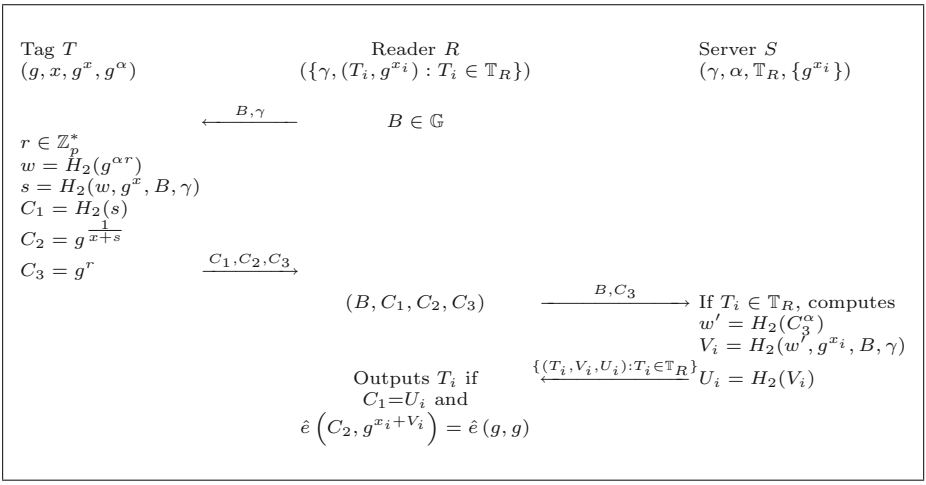

Fig. 2. Authorized RFID authentication protocol 2.

3.3 Protocol 3

ARA protocol 2 engages the reader to perform constant computations during the tag authentication. Instead, the server needs to send the reader a set of possible values for tag identification. In some scenarios where the communication bandwith is limited, it is desired to reduce the size of the set. Hence, we introduce third protocol which only transfers one group element from the server to the reader. Additionally, we consider a new attack that the server may cheat a reader by replying a random value which is called *false output*. Then, the reader could not successfully authenticate a tag even the tag is valid. This attack cannot be detected in neither protocol 1 nor protocol 2. Fortunately, our protocol 3 below shows that it is able to determine whether the received value is a false output. The protocol is depicted as in Fig. 3.

- ServerKeyGen: The server picks α, where $\alpha \in \mathbb{Z}_p^*$, and sets the public key $PK = h^\alpha$ and the private key $SK = \alpha$.
- TagKeyGen: The member randomly chooses $x \in \mathbb{Z}_p^*$ and computes the tag's public and private keys $(pk, sk) = (h^x, g^x)$. (g, sk, PK) are stored in the tag and pk is given to the server.
- ReaderAuth: To authorize the reader R to identify a specified set of tags $\mathbb{T}_R$, the server randomly chooses a secret $\gamma \in \mathbb{Z}_p^*$, and computes g^γ. For each tag $T_i \in \mathbb{T}_R$, the server computes $(h^{x_i})^{\alpha\gamma}$ and sets the reader's period key $rpk = \{g^\gamma, (T_i, h^{\alpha x_i \gamma}) : T_i \in \mathbb{T}_R\}$ and the secret $rsk = (\gamma, \alpha)$. The server stores $(R, rsk, \mathbb{T}_R)$ and sends rpk to the reader R.
- Auth: To authenticate a tag, the following steps are implemented.
  1. The reader randomly selects $B \in \mathbb{G}$ and sends (B, g^γ) to the tag.
  2. Upon receiving (B, g^γ) from the reader, the tag chooses two random numbers $r, s \in \mathbb{Z}_p^*$, and computes a tuple $(C_1, C_2, C_3, C_4, C_5)$. It sends the tuple to the reader as a response.

3. Upon receiving the response, the reader checks $\hat{e}(C_2, C_4) \stackrel{?}{=} \hat{e}(g, A)$. If it holds, the reader forwards (B, C_3, C_4, C_5) to the server.
4. Upon receiving the message (B, C_3, C_4, C_5) from the reader R, the server retrieves (γ, g^γ) and check $\hat{e}(C_3, A') \stackrel{?}{=} \hat{e}(h^\alpha, C_5)$. If it holds, the server calculates and sends $V = C_3^\gamma$ to the reader.
5. Finally, the reader authenticate the tag according to the server's response. Firstly, the checks the equation $\hat{e}(V, g) \stackrel{?}{=} \hat{e}(C_3, g^\gamma)$. If the equation does not hold, the reader outputs false. After that, the reader computes $\hat{e}(V, C_1)$ and checks whether there exists a pair $(T_i, h^{\alpha x_i \gamma}) \in rpk$, such that $\hat{e}(V, C_1) = \hat{e}(h^{\alpha x_i \gamma}, C_2)$. The reader outputs T_i if the above equation holds, otherwise rejects.

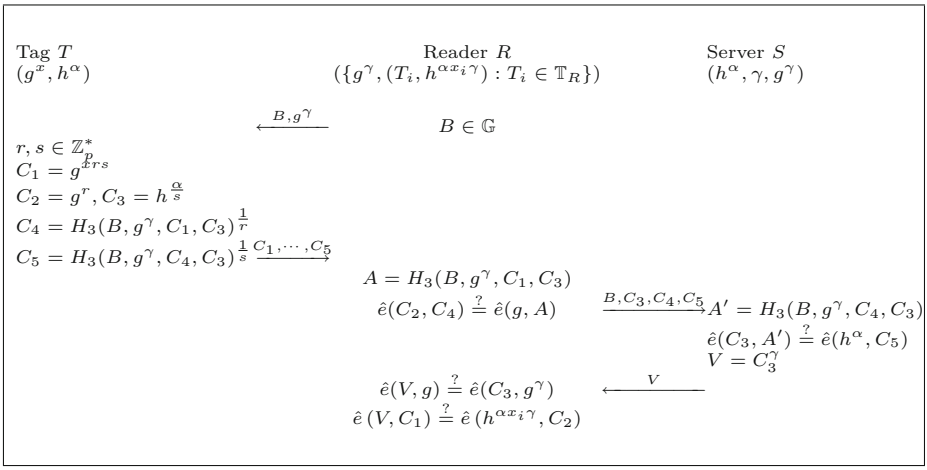

Fig. 3. Authorized RFID authentication Protocol 3.

3.4 Efficiency

We compare the efficiency in Table 2. In Protocol 2, C_1 is used for the reader to quickly identify the tag, and C_2 is for identity verification. The main computational cost of reader is dominated by computing $g^{H_2(V_i, B)}$ and $\hat{e}(C_2, g^{x_i} \cdot g^{H_2(V_i, B)})$ for the verification, where $\hat{e}(g, g)$ can be pre-computed. This protocol requires the server to compute all potential hash values for the reader. The communication cost therefore is all V_i for each tag in $\mathbb{T}_R$. The server can send all hash values in sequences to eliminate sending T_i. In protocol 3, the communication cost and the computational cost of server is constant-size and independent of the size of $\mathbb{T}_R$. The price to pay of this protocol is a liner computation cost on the reader. The reader needs to identify the potential tag one by one until the correct one is found. The computation time therefore is linear in n for $|\mathbb{T}_R| = n$.

Note that Protocol 2 is suitable for computationally weak readers without bandwidth limitation, while Protocol 3 fits for scenarios of limited bandwidth.

Table 2. Efficiency of two Protocols. Here, we assume the reader can identify n tags (i.e., $|\mathbb{T}_R| = n$).

Protocols	Reader Computation Cost	Communication Cost	Server Computation Cost
Protocol 2	$\mathbb{G}+\hat{e}$	$2n\|\mathbb{Z}_p\|$	$\mathbb{G} + 2nH_2$
Protocol 3	$(n+1)\hat{e}$	$\|\mathbb{G}\|$	$\mathbb{G} + \hat{e}$

4 Privacy and Security Models

In this section, we consider the privacy and security models of authorized authentication protocols. We assume that the communication channel between the reader and the server is secure.

4.1 Adversaries and Oracles

We define a set of oracles and four attacks which respectively aim at different goals. In the particular attack, the ability of an adversary is regarded as the actions executed by oracle calls.

Definition 1 (Oracles). *The adversary plays with the challenger by given public information of the system and the following oracles.*

- $\mathsf{TagCorrupt}(T) \rightarrow sk$*: On input a tag* T*, it outputs the tag's private key* sk*.*
- $\mathsf{ReaderAuth}(\mathbb{T}_R) \rightarrow rpk$*: On input a set of tags* $\mathbb{T}_R$*, it outputs the reader's period key* rpk*.*
- $\mathsf{SendTag}(T, m, \pi) \rightarrow m'$*: On input a tag* T*, a message* m *and a session* π*, it sends the message* m *to the tag and receives the tag's response* m'*.*
- $\mathsf{SendServer}(m) \rightarrow m'$*: On input a message* m*, it sends the message* m *to the server and receives the response* m'*.*
- $\mathsf{Challenge}(m^*, T^*) \rightarrow C^*$*: On input a message* m^* *and a target tag* T^* *which is not issued to* ReaderAuth *oracle, it flips a coin* b *and outputs a response* C^* *regarding to the tag* T^* *(if* $b = 1$*) or a random tag* $T^{*\prime}$ *(if* $b = 0$*). This oracle can be called at most once of a game.*

Definition 2 (Strong and weak adversaries). *We define four types of attacks as follows.*

- Forward *attack: The adversary plays as a malicious reader who attempts to trace the tags' previous communications after it has been authorized by the server.*

- Backward *attack: The adversary plays as a malicious reader who attempts to trace the tags' future communications after it has been revoked by the server.*
- Outside *attack: The adversary plays as a dishonest server who attempts to discover the tag which is authenticating by the reader.*
- Impersonation *attack: The adversary plays as an impersonator who is not the tag holder attempts to impersonate the tag which is not compromised without being detected.*

A strong adversary can access all above oracles and launch all above attacks while a weak adversary cannot access the TagCorrypt$(\cdot)$ *oracle.*

4.2 Privacy and Security Models

Forward Privacy. The forward privacy game allows the adversary $\mathcal{A}$ to launch the forward attack. In the ARA system, a reader R may be authorized to authenticate a tag T in a certain period P of time. However, R shall not be able to interpret T's sessions prior to P since R is unauthorized to authenticate T outside the time P. In the forward privacy game, $\mathcal{A}$ is given the reader's current period key and attempts to decide whether the tag which can be authenticated currently was involved in the *previous* interactions.

The forward privacy game is defines in two phases, which are Forward Phase and Backward Phase. $\mathcal{A}$ plays with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen and TagKeyGen to generate the server and tags' public/private keys (PK, SK) and $\{(pk_i, sk_i)\}$, respectively. The challenger gives public keys to $\mathcal{A}$.
- Forward Phase: The challenger sets the reader's period key and $\mathcal{A}$ can query Challenge$(\cdot)$ for the challenge. $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Backward Phase: The challenger refreshes the reader's period key and $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Definition 3. *An authorized authentication scheme provides forward privacy if there is no $\mathcal{A}$ who wins the above game with the probability* $\Pr[b' = b] \geq \frac{1}{2} + \epsilon$, *where ϵ is negligible.*

Backward Privacy. The backward privacy game allows the adversary $\mathcal{A}$ to launch the backward attack. It is different from the forward attack that a reader R attempts to trace the tag after R has been revoked. In the backward privacy game, $\mathcal{A}$ is given the reader's current period key to authenticate the tags, while $\mathcal{A}$ needs to decide whether a tag involves in the *future* interactions after the reader was revoked.

The backward privacy game is defined in two phases, which are Forward Phase and Backward Phase. $\mathcal{A}$ plays with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen and TagKeyGen to generate the server and tags' public/private keys (PK, SK) and $\{(pk_i, sk_i)\}$, respectively. The challenger gives public keys to $\mathcal{A}$.
- Forward Phase: The challenger sets the reader's period key and $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Backward Phase: The challenger refreshes the reader's period key and $\mathcal{A}$ can query Challenge$(\cdot)$ for the challenge. $\mathcal{A}$ interacts with the challenger through the oracles which can be accessed by the classified type of $\mathcal{A}$.
- Guess: $\mathcal{A}_{\mathcal{F}}$ outputs a bit b' and wins the game if $b' = b$.

Definition 4. *An authorized authentication scheme provides backward unlinkability if there is no $\mathcal{A}$ who wins the above game with the probability* $\Pr[b' = b] \geq \frac{1}{2} + \epsilon$*, where ϵ is negligible.*

Reader Privacy. The reader privacy game allows the adversary $\mathcal{A}$ to launch the outside attack. Conventionally, the reader and the server are mutually trusted in RFID systems. However, the reader's privacy is needed to be considered in ARA protocols. For instance, the server may intend to learn the identity of the tag which is authenticating by the reader. Since the reader and the server are operated by different parties, the reader/tag interaction should be invisible to the server. In the reader privacy game, $\mathcal{A}$ is given the secret of the server and attempts to distinguish the tags during the server/reader interactions. $\mathcal{A}$ interacts with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen, TagKeyGen and ReaderAuth to respectively generate the server's public and private keys (PK, SK), tag's public and private keys (pk, sk) and reader's keys (rpk, rsk). The challenger gives the server and tags' public/private keys and the reader's period key rpk to $\mathcal{A}$.
- Query: The adversary is allowed to make queries to the oracle SendServer$(\cdot)$.
- Challenge: The adversary outputs two tags T_0 and T_1 to the challenger. The challenger randomly chooses a bit $b \in \{0, 1\}$. Let M_t be the output of SendTag$(\cdot)$ with respect to the tag T_b and M_s be the corresponding query to SendServer$(\cdot)$. The challenger sends M_s to the adversary.
- Guess: $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

Definition 5. *An authorized authentication scheme provides reader privacy if there is no $\mathcal{A}$ who wins the above game with the probability* $\Pr[b' = b] \geq \frac{1}{2} + \epsilon$*, where ϵ is negligible. We say that it unconditionally preserves the reader privacy if $\epsilon = 0$.*

Tag Unforgeability. The tag unforgeability is with respect to the security of the protocol and the attacker is referred to a malicious sever. This game allows an adversary $\mathcal{A}$ to launch the impersonation attack. Clearly, it is hard to prevent the impersonation attack if the tag is corrupted. Symmetry-key based protocols

are not secure against this attack as a server obtains secret keys of tags during the system setup. Hence, TagCorrupt$(\cdot)$ oracle cannot be queried during the game. $\mathcal{A}$ attempts to forge a tag's response to pass the authentication. It allows $\mathcal{A}$ to access the secret of the server and the reader. $\mathcal{A}$ interacts with the challenger as follows.

- Setup: The challenger runs the algorithms ServerKeyGen, TagKeyGen and ReaderAuth to respectively generate the server's public and private keys (PK, SK), tag's public and private keys (pk, sk) and reader's keys (rpk, rsk). The challenger gives the server's private key, tags' public key and reader's keys to $\mathcal{A}$.
- Query: The adversary can query the oracle SendTag$(\cdot)$ to the challenger.
- Forgery: $\mathcal{A}$ outputs a valid session π which is not queried to the SendTag$(\cdot)$ oracle.

Definition 6. *An authorized authentication scheme provides tag unforgeability if there is no $\mathcal{A}$ who can outputs a valid forgery of the tag with the non-negligible advantage ϵ.*

5 Privacy and Security Analysis

To analyze the privacy and security of protocols, we define two new complexity assumptions which are given in Appendix A. Due to the page limitation, we refer the readers to the full version of this paper for the proof of theorems[1].

Theorem 1. *Our ARA protocol 1 provides forward privacy and backward privacy against the weak adversary if H_1 is pre-image resistant and provides unconditional reader privacy.*

Theorem 2. *Our ARA protocol 2 provides forward privacy and backward privacy against the strong adversary if the ODH assumption holds, tag unforgeability if BB signature [3] is secure and unconditional reader privacy.*

Theorem 3. *Our ARA protocol 3 provides forward privacy and backward privacy against the strong adversary if the V-l-wDBDHI assumption holds, reader privacy if the EDBDH assumption holds and tag unforgeability if k+1-Exponent assumption holds.*

6 Conclusion

In this paper, we introduced a novel concept of authorized RFID authentication protocols. The reader's privacy is considered as a new issue that it prevents the server disclosing the identity of the tag which is authenticated by the reader. Three protocols were proposed based on the different efficiency requirements. We provided the formal definition of privacy and security models of authorized authentication protocols and proved that our protocols are secure against the various adversaries.

[1] The full version of the paper can be requested from the authors.

A Complexity Assumptions

Definition 7 (Oracle Diffie-Hellman Assumption [1]). *Given g^a, g^b, a function $H : \{0,1\}^* \to \{0,1\}^l$ and an oracle $\mathcal{O} = H(X^b)$, where $X \neq g^a$, the advantage of an adversary $\mathcal{A}$ in violating the ODH assumption is*

$$Adv^{odh}_{\mathcal{A},H} = \left|\Pr\left[a, b : \mathcal{A}^{\mathcal{O}}(g^a, g^b, H(g^{ab})) = 1\right] - \Pr\left[a, b : \mathcal{A}^{\mathcal{O}}(g^a, g^b, t) = 1\right]\right|,$$

where $t \in \{0,1\}^l$ We say that the ODH assumption holds, if $Adv^{odh}_{\mathcal{A},H}$ is negligible.

Definition 8 (EDBDH Assumption). *Let $(g, p, \mathbb{G}, \mathbb{G}_T)$ be a pairing group. Given (g, g^a, g^b, g^c, g^t), the Extended Decisional Bilinear Diffie-Hellman problem is to determine whether $g^t = g^{abc}$. We say that the EDBDH assumption holds, if no PPT algorithm $\mathcal{A}$ can solve the problem with non-negligible advantage.*

Definition 9 (V-l-wDBDHI Assumption). *Let $(g, h, p, \mathbb{G}, \mathbb{G}_T)$ be a pairing group. Given $(g, h, g^a, g^{a^2}, \cdots, g^{a^l}, h^a, h^{a^2}, \cdots, h^{a^l}, g^t)$, the Variant l-weak Decisional Bilinear Diffie-Hellman Inversion problem is to determine whether $g^t = g^{a^{2l+1}}$. We say that the V-l-wDBDHI assumption holds, if no PPT algorithm $\mathcal{A}$ can solve the problem with non-negligible advantage.*

Definition 10 (k+1-Exponent Assumption). *Given $(g, g^a, g^{a^2}, \cdots, g^{a^k})$, the k+1-Exponent problem is to compute $g^{a^{k+1}}$. We say that the k+1-Exponent assumption holds, if no PPT algorithm $\mathcal{A}$ can solve the problem with non-negligible advantage.*

We show that the security of EDBDH assumption is related to the security of Decisional Bilinear Diffie-Hellman (DBDH) assumption.

Lemma 1. *The EDBDH assumption holds if the DBDH assumption holds.*

Proof. Suppose that there is a PPT algorithm $\mathcal{A}$ who can break the EDBDH assumption. Given an instance (g, g^a, g^b, g^c, g^t), $\mathcal{A}$ can output whether $g^t = g^{abc}$ in polynomial time with non-negligible advantage. It implies that $\mathcal{A}$ decides whether $\hat{e}(g, g^t) = \hat{e}(g, g^{abc})$ which is a solution of DBDH problem. Therefore, if DBDH problem is intractable then the EDBDH assumption holds. □

In terms of V-l-wDBDHI, a solution of V-l-wDBDHI problem also implies that the algorithm $\mathcal{A}$ can decide whether

$$\hat{e}(g, g^t) = \hat{e}(g, g^{a^{2l+1}}).$$

Since that V-l-wDBDHI problem is modified from l-wDBDHI problem, its security can be bounded by using the similar strategy in the generic group model.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, p. 143. Springer, Heidelberg (2001)
2. Batina, L., Seys, S., Singelée, D., Verbauwhede, I.: Hierarchical ECC-based RFID authentication protocol. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 183–201. Springer, Heidelberg (2012)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
4. Bringer, J., Chabanne, H., Icart, T.: Cryptanalysis of EC-RAC, a RFID identification protocol. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 149–161. Springer, Heidelberg (2008)
5. Burmester, M., Le, T.V., de Medeiros, B., Tsudik, G.: Universally composable RFID identification and authentication protocols. ACM Trans. Inf. Syst. Secur. **12**(4), 1–33 (2009)
6. Canard, S., Coisel, I., Etrog, J., Girault, M.: Privacy-preserving RFID systems: model and constructions. IACR Cryptology ePrint Archive **2010**, 405 (2010)
7. Deng, R.H., Li, Y., Yung, M., Zhao, Y.: A new framework for RFID privacy. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 1–18. Springer, Heidelberg (2010)
8. van Deursen, T., Radomirović, S.: Untraceable RFID protocols are not trivially composable: attacks on the rvision of ec-rac. IACR Cryptol. ePrint Archive **2009**, 332 (2009)
9. van Deursen, T., Radomirović, S.: EC-RAC: enriching a capacious RFID attack collection. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 75–90. Springer, Heidelberg (2010)
10. Fan, J., Hermans, J., Vercauteren, F.: On the claimed privacy of EC-RAC III. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 66–74. Springer, Heidelberg (2010)
11. Hein, D., Wolkerstorfer, J., Felber, N.: ECC is ready for RFID – a proof in silicon. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 401–413. Springer, Heidelberg (2009)
12. Hermans, J., Pashalidis, A., Vercauteren, F., Preneel, B.: A new RFID privacy model. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 568–587. Springer, Heidelberg (2011)
13. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 52. Springer, Heidelberg (2001)
14. Juels, A., Weis, S.A.: Authenticating pervasive devices with human protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
15. Juels, A., Weis, S.A.: Defining strong privacy for RFID. In: PerCom Workshops, pp. 342–347. IEEE Computer Society (2007)
16. Lee, Y.K., Batina, L., Verbauwhede, I.: Ec-rac (ecdlp based randomized access control): Provably secure RFID authentication protocol. In: 2008 IEEE International Conference on RFID, pp. 97–104 (2008)
17. Lee, Y.K., Batina, L., Verbauwhede, I.: Untraceable RFID authentication protocols: Revision of EC-RAC. In: 2009 IEEE International Conference on RFID, pp. 178–185 (2009)

18. Lee, Y.K., Batina, L., Singelée, D., Verbauwhede, I.: Wide–weak privacy–preserving RFID authentication protocols. In: Chatzimisios, P., Verikoukis, C., Santamaría, I., Laddomada, M., Hoffmann, O. (eds.) MOBILIGHT 2010. LNICST, vol. 45, pp. 254–267. Springer, Heidelberg (2010)
19. Lee, Y.K., Sakiyama, K., Batina, L., Verbauwhede, I.: Elliptic-curve-based security processor for RFID. IEEE Trans. Computers **57**(11), 1514–1527 (2008)
20. Ng, C.Y., Susilo, W., Mu, Y., Safavi-Naini, R.: RFID privacy models revisited. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 251–266. Springer, Heidelberg (2008)
21. Nithyanand, R., Tsudik, G., Uzun, E.: Readers behaving badly. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 19–36. Springer, Heidelberg (2010)
22. Nithyanand, R., Tsudik, G., Uzun, E.: User-aided reader revocation in PKI-based RFID systems. J. Comput. Secur. **19**(6), 1147–1172 (2011)
23. Song, B., Mitchell, C.J.: RFID authentication protocol for low-cost tags. In: Gligor, V.D., Hubaux, J.P., Poovendran, R. (eds.) WISEC, pp. 140–147. ACM (2008)
24. Tsudik, G.: Ya-trap: Yet another trivial RFID authentication protocol. In: PerCom Workshops, pp. 640–643. IEEE Computer Society (2006)
25. Tuyls, P., Batina, L.: RFID-tags for anti-counterfeiting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 115–131. Springer, Heidelberg (2006)
26. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)