2017

An efficient and provably secure RFID grouping proof protocol

Shu Cheng
Macquarie University

Vijay Varadharajan
Macquarie University

Yi Mu
University of Wollongong, ymu@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Publication Details

Cheng, S., Varadharajan, V., Mu, Y. & Susilo, W. (2017). An efficient and provably secure RFID grouping proof protocol. Australasian Computer Science Week Multiconference (ACSW '17) (pp. 1-7). New York: ACM.

An efficient and provably secure RFID grouping proof protocol

Abstract
RFID Grouping proof convinces an offline verifier that multiple tags are simultaneously scanned. Various solutions have been proposed but most of them have security and privacy vulnerabilities. In this paper, we propose an elliptic-curve-based RFID grouping proof protocol. Our protocol is proven secure and narrow-strong private. We also demonstrate that our grouping proof can be batch verified to improve the efficiency for large-scale RFID systems and it is suitable for low-cost RFID tags.

Keywords
proof, grouping, protocol, rfid, efficient, provably, secure

Publication Details
Cheng, S., Varadharajan, V., Mu, Y. & Susilo, W. (2017). An efficient and provably secure RFID grouping proof protocol. Australasian Computer Science Week Multiconference (ACSW '17) (pp. 1-7). New York: ACM.

An Efficient and Provably Secure RFID Grouping Proof Protocol*

Shu Cheng
Advanced Cyber Security Research Centre
Macquarie University
Sydney, Australia
shu.cheng@hdr.mq.edu.au

Vijay Varadharajan
Advanced Cyber Security Research Centre
Macquarie University
Sydney, Australia
vijay.varadharajan@mq.edu.au

Yi Mu
Centre for Computer and Information Security Research
University of Wollongong
Wollongong, Australia
ymu@uow.edu.au

Willy Susilo
Centre for Computer and Information Security Research
University of Wollongong
Wollongong, Australia
wsusilo@uow.edu.au

ABSTRACT

RFID Grouping proof convinces an offline verifier that multiple tags are simultaneously scanned. Various solutions have been proposed but most of them have security and privacy vulnerabilities. In this paper, we propose an elliptic-curve-based RFID grouping proof protocol. Our protocol is proven secure and narrow-strong private. We also demonstrate that our grouping proof can be batch verified to improve the efficiency for large-scale RFID systems and it is suitable for low-cost RFID tags.

1. INTRODUCTION

Radio Frequency Identification (RFID) technologies are widely deployed in the industry such as logistics and supply chain management nowadays. Goods are attached with RFID tags and their information is stored in the tags. Consider the application scenario where several items are required to be delivered together. The logistics service provider needs a proof to convince the customer that this has been achieved. In 2004, Juels [12] introduced the concept called yoking proof that proves two tags has been scanned simultaneously. In his proposal, an untrusted reader interacts with two tags and generates a proof to guarantee the combined presence for the trusted offline verifier. Later, Saito and Sakurai [19] extended the concept to grouping proof which allows multiple tags to generate the proof of presence.

Numerous protocols have been proposed in the literature since the introduction of yoking/grouping proof. Juels presented two yoking-proofs in [12] based on message authentication code (MAC). However, both of them are vulnerable to replay attacks and compromised tag attacks as illustrated in [19, 3, 4]. Subsequently, some other grouping proof schemes based on MAC were proposed in [19, 18] but they are still vulnerable to replay attacks. In 2008, Burmester, Medeiros and Motta [4] presented a security model for grouping proofs based on Universal Composability framework. They proposed three grouping proof schemes that provide anonymity and forward security properties. The protocols are proved secure in their model. Later, Peris-Lopez, Orfila, Hernandez-Castro and Lubbe [17] showed that the protocol is vulnerable to impersonation attacks. Some other protocols based on symmetric-key cryptography were proposed in [15, 7, 6, 11, 5]. However they all suffer impersonation attacks and/or anonymity attacks as illustrated in [17].

Recent studies [1, 8, 14] show that public-key cryptography is feasible for lightweight RFID tags. Batina et al. [2] proposed a grouping proof based on elliptic curve cryptography to prevent colluding tag attacks. But the scheme is vulnerable to impersonation attacks and man-in-the-middle (MITM) attacks as shown in [10]. Hermans and Peeters [10] introduced two grouping proof protocols. The protocols are proved to be impersonation attack resistant and narrow-strong private. As [16] pointed out, however, whether the protocols are secure against MITM attacks is not clear. Vaudenay [20] proved that it is essential to employ public-key cryptography in RFID to fulfil the highest privacy requirement. Symmetric-key based RFID protocols always leak messages with recognisable information and is not scalable in large-scale RFID systems. Moreover, corrupted tags are identifiable even in past and future protocol runs. In this paper we focus on public-key based grouping proof protocols to guarantee security against powerful adversaries and provide strong tag privacy.

1.1 Our Contribution

We propose a novel and efficient RFID grouping proof protocol based on elliptic curve cryptography. The scheme is provably secure against MITM attacks and impersonation attacks. It offers tag privacy against powerful narrow-strong

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

ACSW '17, January 31-February 03, 2017, Geelong, Australia

DOI: http://dx.doi.org/10.1145/3014812.3014885

adversaries in the model described in [9]. Additionally we show that our protocol is especially suitable for verifying proofs generated by a large number of tags.

The rest of this paper is organised as follows: In Section 2, we describe the preliminaries for our scheme and define the security and privacy model. Our elliptic-curve-based RFID grouping proof protocol is proposed in Section 3. In Section 4, we analyse our scheme and prove its security and privacy. In Section 5, we show how to develop our scheme into an efficient batch proof verification scheme and we evaluate the performance of the scheme. Section 6 concludes this paper.

2. PRELIMINARIES

We now give the definitions of related complexity assumptions and present the security and privacy model.

2.1 Complexity Assumptions

Let $\mathbb{G}$ be a cyclic additive group with order q, where q is a k-bit prime.

DEFINITION 2.1. **Computational Diffie-Hellman (CDH) Problem.** *Given a randomly chosen generator $P \in \mathbb{G}$, as well as aP, bP for unknown randomly chosen a, $b \in \mathbb{Z}_q^*$, compute abP.*

DEFINITION 2.2. **Decisional Diffie-Hellman (DDH) Problem.** *Given a randomly chosen generator $P \in \mathbb{G}$, as well as aP, bP and cP for unknown randomly chosen a, b, $c \in \mathbb{Z}_q^*$, decide whether $cP = abP$.*

DEFINITION 2.3. **Decisional Diffie-Hellman (DDH) Assumption.** *The DDH problem is (t, ϵ)-hard, if there is no probabilistic polynomial-time adversary $\mathcal{A}$ that can solve the DDH problem with a probability $Succ_{DDH} > \epsilon$ in time t.*

DEFINITION 2.4. **One More Computational Diffie-Hellman (OMCDH) Problem.** *Given a randomly chosen generator $P \in \mathbb{G}$, an element $aP \in \mathbb{G}$, an oracle that can solve the CDH problem for the given aP and arbitrary $bP \in \mathbb{G}$, and a challenge oracle that returns random point $b_iP \in \mathbb{G}$. After $n+1$ queries to the challenge oracle and at most n queries to the CDH oracle, an efficient polynomial-time algorithm must compute the solutions ab_iP of all CDH instances with input aP, b_iP $(i = 0, 1, ..., n)$.*

DEFINITION 2.5. **One More Computational Diffie-Hellman (OMCDH) Assumption.** *The OMCDH problem is (t, ϵ)-hard, if there is no probabilistic polynomial-time adversary $\mathcal{A}$ that can solve the CDH problem with a probability $Succ_{OMCDH} > \epsilon$ in time t.*

2.2 Security and Privacy Model

There are three different parties engaging in the grouping proof protocol: a group of tags, a reader and a verifier.

- Tags T_i are low-cost devices and are able to perform lightweight cryptographic operations such as elliptic curve cryptography. A unique key is assigned to each of the tags while some secrets may also be shared among all the tags.
- Reader R is potentially untrusted since it can be controlled by a malicious third party.
- Verifier V is an offline trusted third party. Its public key is known by all parties involving in the protocol.

The reader coordinates the execution of the protocol and relays the messages between the tags. At the end of the protocol run, the grouping proof is constructed by the tags and the reader for the verifier to verify at a certain time later. Both the tags and the reader have a timeout mechanism. They measure the time between sending a message and receiving the corresponding response. Once the time exceeds the preset threshold, the protocol execution will be terminated. This mechanism guarantees the grouping proof is generated by simultaneously scanning the tags.

Tags are vulnerable to compromise due to their limited computational capability. An adversary can read the internal state of a tag. We consider the attack scenario where at most $n-1$ tags in a group of n tags are compromised. It is a trivial case that the adversary compromises all the tags and obtains their secrets because it can then forge a valid grouping proof easily while none of the tags is present.

We use the privacy model introduced by Hermans et al. [9] in this paper. The oracles defined in the model are as follows.

- $\mathsf{CreateTag}(\mathrm{ID}) \rightarrow T_i$: the oracle takes the identifier ID of a tag as input and registers the tag to the server. T_i is returned as the reference of the tag.
- $\mathsf{Launch}() \rightarrow \pi, m$: the oracle launches a new protocol instance π as well as the first message m sent by the reader.
- $\mathsf{DrawTag}(T_i, T_j) \rightarrow vtag$: the oracle takes two tag references T_i and T_j as input and generates the virtual reference $vtag$. Depending on the value of a random bit b chosen by the challenger, $vtag$ refers to either T_i (if $b = 0$) or T_j (if $b = 1$). The oracle outputs $\perp$ if either of the tags has been drawn without being freed. Otherwise it outputs $vtag$.
- $\mathsf{Free}(vtag)_b$: the oracle takes $vtag$ as input and retrieves the tuple $(vtag, T_i, T_j)$. Depending on the same value of b chosen by the challenger, the oracle resets the volatile memory of tag T_i (if $b = 0$) or T_j (if $b = 1$). Then the oracle moves T_i and T_j from the set of drawn tags to the set of free tags.
- $\mathsf{SendTag}(vtag, m)_b \rightarrow m'$: the oracle takes a message m and a tag reference $vtag$ as input. It retrieves the tuple $(vtag, T_i, T_j)$ and sends m to T_i (if $b = 0$) or T_j (if $b = 1$). Then the oracle outputs the response message m' from the tag.
- $\mathsf{SendReader}(\pi, m) \rightarrow m'$: the oracle takes a protocol instance π and a message m as an input. It returns $\perp$ if π is not an active instance. Otherwise it outputs the response m' from the reader.
- $\mathsf{Corrupt}(T_i) \rightarrow s$: the oracle takes a tag reference T_i as input, and outputs the internal state s of the tag.

Note that we omit the detail of the oracle $\mathsf{Result}(\pi)$. The oracle is not used in grouping proof protocol because the verifier verifies the grouping proof offline at a certain stage later.

The model also defines eight different notions for privacy and adversaries. A *wide* adversary has access to Result oracle while a *narrow* adversary does not. Result is not used in the grouping proof protocol as we mentioned above. Thus we only consider *narrow* adversaries in this paper. Orthogonal to these two classes, there are *weak*, *forward*, *destructive* and *strong* adversaries. They are classified with the capabilities of different oracle access. A *strong* adversary can access all of the seven oracles defined above multiple times in any order.

2.3 Grouping Proof

A sound grouping proof protocol should be correct, secure and private. Correctness means a legitimately generated grouping proof will always be accepted and all the involved tags should be identified by the verifier correctly.

A scheme is secure if an adversary cannot impersonate a legitimate tag and forge a valid proof without corrupting this tag. We consider two types of adversary based on the ability.

Type I adversary ($\mathcal{A}_I$) can perform MITM attacks to grouping proof protocols. $\mathcal{A}_I$ can interact with the system by calling Launch, SendTag, SendReader oracle. The tags are assumed to be non-compromised. After the oracle calls, $\mathcal{A}_I$ outputs a proof that is not constructed when all the tags are in a matching session (otherwise the proof is valid). $\mathcal{A}_I$ wins if the proof is accepted by the verifier. The details of Type I security experiment are described in Figure 1.

Experiment $\mathbf{Exp}^{secure}_{\mathcal{S},\mathcal{A}_I}(k)$:

- Setup: The challenger $\mathcal{S}$
  - initialises the system with the security parameter k.
  - sets up the verifier and creates a set of tags.
- Learning: $\mathcal{A}_I$ interacts with Launch, SendTag, SendReader oracles.
- Challenge: $\mathcal{A}_I$ returns a candidate grouping proof σ for a set of tags $\{T_i : i \in [0,n]\} \subseteq \mathcal{T}$. If σ is accepted by the verifier and σ is not constructed when all the tags T_i are in the same protocol instance, $\mathcal{S}$ outputs 1. Otherwise outputs 0.

Figure 1: Type I Security experiment of grouping proof protocols.

Definition 2.6. *An RFID tag grouping proof scheme is secure against MITM attacks, if for any polynomially bounded adversary $\mathcal{A}_I$, the probability of success of winning the experiment $\mathbf{Exp}^{secure}_{\mathcal{S},\mathcal{A}_I}(k)$ is negligible. In other words,*

$$\mathbf{Adv}^{secure}_{\mathcal{A}_I} = |Pr[\mathbf{Exp}^{secure}_{\mathcal{S},\mathcal{A}_I}(k) = 1]| \leq \epsilon.$$

Type II adversary ($\mathcal{A}_{II}$) is able to compromise tags and perform impersonation attacks. $\mathcal{A}_{II}$ monitors the communications between the reader and all the participating tags, and controls all the tags in the group except for one tag T_0. Note that we do not consider the attack scenario where all the tags are compromised because it would allow the adversary to generate a valid proof without the presence of the tags. $\mathcal{A}_{II}$ is able to interacts with the system by using SendTag, SendReader, Corrupt oracles. $\mathcal{A}_{II}$ outputs a proof in the end without the presence of T_0. $\mathcal{A}_{II}$ wins if the proof is accepted by the verifier. The details of Type II security experiment are described in Figure 2.

Experiment $\mathbf{Exp}^{secure}_{\mathcal{S},\mathcal{A}_{II}}(k)$:

- Setup: The challenger $\mathcal{S}$
  - initialises the system with the security parameter k.
  - sets up the verifier and creates a set of tags.
  - picks a tag T_0 as the target tag.
- Learning: $\mathcal{A}_{II}$ interacts with SendTag and SendReader oracles for all the tags. $\mathcal{A}_{II}$ can also interacts with Corrupt(T_i) oracle for $T_i \neq T_0$.
- Challenge: $\mathcal{A}_{II}$ returns a candidate grouping proof σ. If σ is accepted by the verifier, $\mathcal{S}$ outputs 1. Otherwise outputs 0.

Figure 2: Type II Security experiment of grouping proof protocols.

Definition 2.7. *An RFID tag grouping proof scheme is secure against Type II attacks, if for any polynomially bounded adversary $\mathcal{A}_{II}$, the probability of success of winning the experiment $\mathbf{Exp}^{secure}_{\mathcal{S},\mathcal{A}_{II}}(k)$ is negligible. In other words,*

$$\mathbf{Adv}^{secure}_{\mathcal{A}_{II}} = |Pr[\mathbf{Exp}^{secure}_{\mathcal{S},\mathcal{A}_{II}}(k) = 1]| \leq \epsilon.$$

The privacy of grouping proof protocols is based on an indistinguishing experiment. The challenger sets up a system and pick a random bit b. The adversary is able to interact with the system by using CreateTag, Launch, DrawTag, Free, SendTag, SendReader, and Corrupt oracles. After interacting with the oracles, $\mathcal{A}$ outputs a guess bit g. $\mathcal{A}$ wins if $g = b$. The details of the privacy experiment are described in Figure 3.

Experiment $\mathbf{Exp}^{NS-private,b}_{\mathcal{S},\mathcal{A}}(k)$:

- Setup: The challenger $\mathcal{S}$
  - initialises the system with the security parameter k.
  - sets up the verifier.
  - chooses $b \in_R \{0,1\}$.
- Learning: $\mathcal{A}$ interacts with all the seven oracles defined above.
- Challenge: $\mathcal{A}$ returns a guess bit g. If $g = b$, $\mathcal{S}$ outputs 1. Otherwise outputs 0.

Figure 3: Narrow-strong privacy experiment of grouping proof protocols.

DEFINITION 2.8. *An RFID tag grouping proof scheme is narrow-strong private, if for any polynomially bounded adversary $\mathcal{A}$ of the class narrow-strong, the probability of success of winning the experiment* $\mathbf{Exp}_{\mathcal{S},\mathcal{A}}^{NS-private,b}(k)$ *is negligible. In other words,*

$$\mathbf{Adv}_{\mathcal{A}}^{NS-private} = |Pr[\mathbf{Exp}_{\mathcal{S},\mathcal{A}}^{NS-private}(k) = 1] - \frac{1}{2}| \leq \epsilon.$$

3. OUR PROPOSED PROTOCOL

Firtly, we propose an RFID grouping proof protocol based on elliptic curve cryptography, which is shown in Figure 5. The steps of our proposed protocol is illustrated below.

Initialisation phase

Let E be an elliptic curve defined over a finite field $\mathbb{Z}_q^*$, where q is an k-bit prime number. Assume P is a generator of $\mathbb{G}$, which is an additive cyclic group of points on the elliptic curve E. Let (x_i, $X_i = x_iP$) denote the private/public key pairs of the tag T_i in a group of n tags, and (y,$Y = yP$) denote the private/public key pair of the verifier V, where $x_i, y \in \mathbb{Z}_q^*$. $H : \{0,1\}^d \times \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^k$ is a random function that takes as input a d-bit and two k-bit strings, and outputs a k-bit string. $k_g \in \{0,1\}^d$ is a d-bit randomly selected group secret key and shared between tags. The notations are depicted in Table 1.

Table 1: Notion of symbols

P	:	a generator of the elliptic curve
x_i	:	the private key of the tag T_i
$X_i(= x_iP)$	:	the public key of the tag T_i
y	:	the private key of the verifier V
$Y(= yP)$	:	the public key of the verifier V
k_g	:	the group secret key shared between the tags
H	:	a random function

Construction phase

The grouping proof is constructed by the tags in a chain structure as shown in Figure 4.

1. The reader R randomly chooses a number $r_s \in \mathbb{Z}_q^*$ and broadcasts ("*init*", r_s) to a group of tags to launch a new protocol run.

2. Each tag T_i randomly chooses $r_i \in \mathbb{Z}_q^*$ and sends r_i to R. Upon receiving the responses from all the tags, R assigns an index to each of the tags, which indicates the order for R to interact the tags.

3. For the tag T_0, R sends r_{n-1} as input. T_0 randomly chooses $s_0 \in \mathbb{Z}_q^*$ and sends $A_0 = s_0H(k_g, r_{n-1}, r_0)Y$, $B_0 = s_0P + x_0r_sP$ to the reader R. For each tag T_i, $1 < i \leq n-1$, R sends r_{i-1} as input and gets (r_i, A_i, B_i) as output, where $A_i = s_iH(k_g, r_{i-1}, r_i)Y$, $B_i = s_iP + x_ir_sP$.

4. R outputs $\sigma = \{r_s, (r_i, A_i, B_i)_{i\in[0,n-1]}\}$ as the grouping proof and send σ to the verifier for later verification.

The details of our protocol is illustrated in Figure 5.

4. PROTOCOL ANALYSIS

We analyse our scheme in three steps. Firstly we show the correctness of our scheme; secondly we investigate the security of our scheme; finally we prove the privacy of our scheme.

4.1 Correctness

To verify the grouping proof, the verifier V first checks that the proof was not used before to prevent replay attacks. Then V performs the computations for $i \in [0, n-1]$

$$\begin{aligned} h_i &= \begin{cases} H(k_g, r_{n-1}, r_0), & \text{if } i = 0. \\ H(k_g, r_{i-1}, r_i), & \text{otherwise.} \end{cases} \\ X_i &= r_s^{-1}(B_i - y^{-1}h_i^{-1}A_i). \end{aligned}$$

The reader can directly retrieve the public key of each tag from the proof instead of an exhaustive search through a database. If all the public keys are in the database and in the same group, the proof is accepted.

4.2 Security

THEOREM 4.1. *The proposed grouping proof protocol is secure against Type I attacks in the random oracle model if* $\frac{(2^k-n)!}{2^{k!}}$ *is negligible.*

Proof. Let $\mathcal{A}_I$ be a Type I adversary that can perform MITM attacks to our proposed protocol. A simulator $\mathcal{S}$ sets up the system $(k, q, d, P, \mathbb{G}, E, H)$. Assume there are n tags in the group. Let k_g denote the shared group key. Let (x_i, x_iP) denote the private/public key pair of the tag T_i. Let $(y, Y = yP)$ denote the key pair of the verifier. $\mathcal{S}$ maintains a list $L_H = \{(k_g, r_j, r_i, h)\}$ and a list $L_T = \{(T_i, r_i)\}$. Both lists are initially empty. W.l.o.g. we assume there is only one tag T_0 that is not in the same protocol instance with the other tags. $\mathcal{A}_I$ calls Launch oracle twice and gets $(\pi_0, r_{s0}P)$ and $(\pi_1, r_{s1}P)$. Let π_0 denote the protocol instance where $\mathcal{A}_I$ interacts with T_0 and π_1 denote the instance for the rest tags. $\mathcal{S}$ simulates SendTag oracle as follows:

- First SendTag(T_i, r_sP): If $(T_i = T_0 \wedge r_sP = r_{s0}P) \vee (T_i \neq T_0 \wedge r_sP = r_{s1}P)$, $\mathcal{S}$ outputs $r_i \in_R \mathbb{Z}_q^*$; otherwise outputs $\perp$. If there is an entry $(T_i, \cdot)$ in L_T, $\mathcal{S}$ removes the entry. $\mathcal{S}$ adds (T_i, r_i) to L_T.

- Second SendTag(T_i, r_j): If there is no entry $(T_i, \cdot)$ in L_T, $\mathcal{S}$ outputs $\perp$. Otherwise $\mathcal{S}$ retrieves r_i from the entry (T_i, r_i). $\mathcal{S}$ looks up the list L_H. If there is an entry $(k_g, r_j, r_i, \cdot)$ in L_H, $\mathcal{S}$ obtains the value h; otherwise, $\mathcal{S}$ selects $h \in_R \{0,1\}^k$ such that there is no existing entry $(k_g, \cdot, \cdot, h)$ in L_H, and adds (k_g, r_j, r_i, h) to L_H. If $T_i = T_0$, $\mathcal{S}$ outputs $(A = shY, B = sP + x_0r_{s0}P)$; otherwise $\mathcal{S}$ outputs $(A = shY, B = sP + x_ir_{s1}P)$, where $s \in_R \mathbb{Z}_q^*$.

At the end of the game, $\mathcal{A}_I$ outputs

$$\sigma = \{r_s^*, (r_i^*, A_i^*, B_i^*) | i \in [0, n-1]\}$$

as the candidate proof. Upon receiving σ, $\mathcal{S}$ checks L_H. If there is an entry $(k_g, r_{i-1}^*, r_i^*, h_i^*)$, $\mathcal{S}$ retrieves the value h_i^*; otherwise, $\mathcal{S}$ randomly chooses $h_i^* \in \{0,1\}^k$ such that there is no entry $(k_g, \cdot, \cdot, h_i^*)$. $\mathcal{S}$ adds $(k_g, r_{i-1}^*, r_i^*, h_i^*)$ to L_H. Note that for $i = 0, r_{i-1}^* = r_{n-1}^*$. $\mathcal{S}$ then verifies

$$X_i = r_s^{*-1}(B_i^* - y^{-1}h_i^{*-1}A_i^*).$$

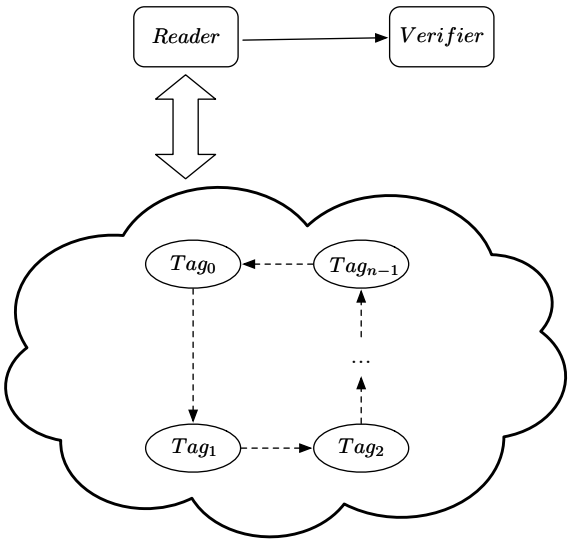

Figure 4: Grouping proof systems

If the equation holds, the grouping proof is valid and $\mathbf{Exp}^{secure}_{\mathcal{S},\mathcal{A}_I}(k)$ outputs 1. In order for this to occur, $\mathcal{A}_I$ must use the same h_i^* to compute A_i. By the definition of Type I adversary $\mathcal{A}_I$ cannot corrupt the tags to get k_g, so $\mathcal{A}_I$ has no access to the list L_H. Since L_H is uniformly distributed, $\mathcal{A}_I$ can only choose the same $\{h_i^*|i \in [0, n-1]\}$ with the probability no more than $\frac{1}{P(2^k,n)}$. Thus we have $\mathbf{Adv}^{secure}_{\mathcal{A}_I} \leq \frac{(2^k-n)!}{2^k!}$. □

THEOREM 4.2. *The proposed grouping proof protocol is secure against Type II attacks if the OMCDH problem is (t, ϵ)-hard in a cyclic additive group $\mathbb{G}$.*

Proof. Assume $\mathcal{A}_{II}$ is a Type II adversary against the proposed protocol. We now show how to construct an algorithm $\mathcal{B}$ to solve the OMCDH problem that executes $\mathcal{A}_{II}$. Let P be a generator of $\mathbb{G}$ of order k. Let $\mathcal{O}_{CDH}(P, aP, bP)$ be an oracle that outputs the CDH solution abP. Let $\mathcal{O}_1$ be an oracle that outputs a random element $bP \in \mathbb{G}$. Given an OMCDH instance $(P, aP, b_0P, ..., b_nP)$, $\mathcal{B}$ now sets up the system. $\mathcal{B}$ creates a group of m tags. W.l.o.g. we assume T_0 is the target tag that $\mathcal{A}_{II}$ wants to impersonate. $\mathcal{B}$ sets aP as the public key and a as the private key of T_0. Note that a is unknown to $\mathcal{B}$. $\mathcal{B}$ randomly selects private/public key pairs (x_i, x_iP) for the rest $m-1$ tags. $\mathcal{B}$ chooses k_g as the shared group key and (y, yP) as the private/public key pair of the verifier. $\mathcal{B}$ also maintains a list L_H=$\{(k_g, r_j, r_i, h)\}$, which is initially empty.

$\mathcal{B}$ broadcasts the message ("*init*", $b_lP \leftarrow \mathcal{O}_1()$) to initiate $(l+1)$th protocol run. $\mathcal{B}$ interacts with $\mathcal{A}_{II}$ as follows.

- Assume $\mathcal{A}_{II}$ issues H query on input (k_g, r_j, r_i) at most q_H times. $\mathcal{B}$ outputs h if there is an entry (k_g, r_j, r_i, h) in the list L_H; otherwise $\mathcal{B}$ randomly chooses h such that there is no entry $(k_g, \cdot, \cdot, h)$ in L_H and adds the entry (k_g, r_j, r_i, h) to L_H.
- For $\mathsf{Corrupt}$ queries, $\mathcal{B}$ outputs the private key x_i of T_i, as well as the shared group key k_g.
- $\mathsf{SendTag}$ query is trivial to all the tags except for T_0 because $\mathcal{A}_{II}$ can calculate the output using the keys of the compromised tags. When $\mathcal{A}_{II}$ issues a $\mathsf{SendTag}$ query to T_0 with the input r_j, $\mathcal{B}$ checks the list L_H. If (k_g, r_j, r_0, h) is in L_H, $\mathcal{B}$ obtains h; otherwise, $\mathcal{B}$ randomly chooses h such that there is no entry $(k_g, \cdot, \cdot, h)$ in L_H and adds the entry (k_g, r_j, r_0, h) to L_H. $\mathcal{B}$ then calls $\mathcal{O}_{OMCDH}(P, aP, b_lP)$ and gets ab_lP. $\mathcal{B}$ outputs (A, B) to $\mathcal{A}_{II}$ where $A = shY$, $B = sP + ab_lP$, $s \in_R \mathbb{Z}_q^*$.

In the challenge phase, $\mathcal{B}$ broadcasts $b_nP \leftarrow \mathcal{O}_1()$ to initiate $(n+1)$th protocol run. $\mathcal{B}$ sends r_j^* to $\mathcal{A}_{II}$ and $\mathcal{A}_{II}$ outputs (r_0^*, A^*, B^*). $\mathcal{B}$ finds the entry (k_g, r_j^*, r_0^*, h^*) in L_H. Then $\mathcal{B}$ computes $ab_nP = B^* - y^{-1}h^{*-1}A^*$. $\mathcal{B}$ outputs $\{ab_lP\}_{l\in[0,n]}$ to the challenger, thereby solving the OMCDH problem.

The simulation fails only when (r_0^*, A^*, B^*) is a valid grouping proof while there is no entry of (k_g, r_j^*, r_0^*, h^*) in the list L_H. Since L_H is uniformly distributed, this only occurs with the probability no more than $\frac{1}{2^k - q_H}$. Thus we have $\mathbf{Adv}^{secure}_{\mathcal{A}_{II}} \leq \epsilon + \frac{1}{2^k-q_H}$. □

4.3 Narrow-Strong Privacy

THEOREM 4.3. *The proposed grouping proof protocol is narrow-strong private if the DDH problem is (t, ϵ)-hard in a cyclic additive group $\mathbb{G}$.*

Proof. Assume there is an adversary $\mathcal{A}$ that can win the narrow-strong experiment $\mathbf{Exp}^{NS-private,b}_{\mathcal{S},\mathcal{A}}(k)$. We now construct an algorithm $\mathcal{B}$ run by the challenger that can solve the DDH problem using $\mathcal{A}$. Given a DDH instance (P, aP, bP, cP), $\mathcal{B}$ sets $Y = bP$ as the public key of the verifier and k_g as the shared group key. $\mathcal{B}$ chooses a random bit

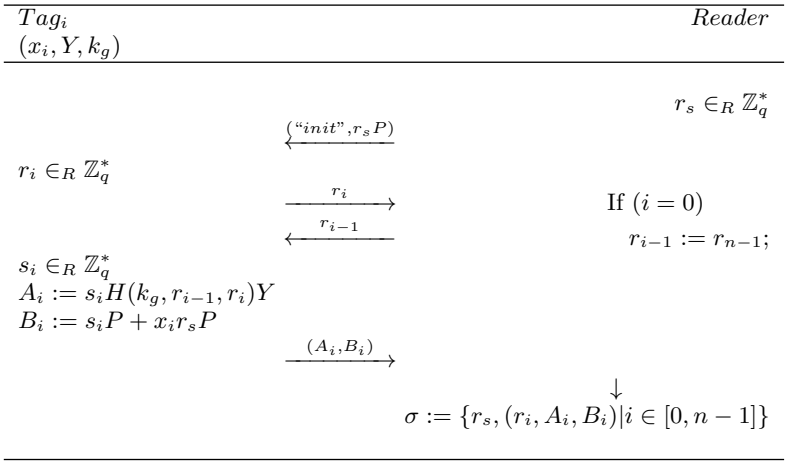

Figure 5: The proposed gouping proof protocol

$b \in 0, 1$. $\mathcal{B}$ maintains a list $L_H = \{(k_g, r_j, r_i, h)\}$, which is initially empty. $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

- CreateTag(ID): $\mathcal{B}$ creates a tag reference T_i and sets (x, xP) as the key pair of T_i.
- Launch(): $\mathcal{B}$ randomly chooses r_sP and outputs $(\text{“}init\text{”}, r_sP)$.
- DrawTag(T_i, T_j): $\mathcal{B}$ sets T_i and T_j as drawn tags. $\mathcal{B}$ creates a virtual tag $vtag$. $vtag$ refers to T_i if $b = 0$; otherwise $vtag$ refers to T_j.
- Free($vtag$): $\mathcal{B}$ resets the volatile memory of $vtag$ and sets T_i and T_j as free tags.
- Corrupt(T_i): $\mathcal{B}$ outputs the private key x_i of T_i, as well as the shared group key k_g.
- SendReader: $\mathcal{B}$ takes r_i as input and outputs a randomly chosen r_j.
- First SendTag: $\mathcal{B}$ takes $(vtag, \text{“}init\text{”}, r_sP)$ as input and outputs a randomly chosen r_i.
- Second SendTag: Upon receiving the input $(vtag, r_j)$, $\mathcal{B}$ selects $s \in_R \mathbb{Z}_q^*$. If there is no record $(k_g, r_j, \cdot, h)$ in the list L_H, $\mathcal{B}$ randomly chooses h such that $(k_g, \cdot, \cdot, h)$ is not in L_H and adds (k_g, r_j, r_i, h) to the list; otherwise $\mathcal{B}$ gets h. B outputs $A = hcP$, $B = aP + x_br_sP$, where x_b is the private key of T_i or T_j depending on b.
- H(k_g, r_j, r_i): Assume $\mathcal{A}_{II}$ issues H query at most q_H times. $\mathcal{B}$ outputs h if there is an entry (k_g, r_j, r_i, h) in the list L_H; otherwise $\mathcal{B}$ randomly chooses h such that there is no entry $(k_g, \cdot, \cdot, h)$ in L_H and adds (k_g, r_j, r_i, h) to L_H.

Eventually $\mathcal{A}$ outputs a guess bit $g \in \{0, 1\}$ and $\mathcal{B}$ can use g to solve the DDH problem. $cP = abP$ if $g = b$; otherwise $cP \neq abP$. The simulation fails only when $\mathcal{A}$ outputs g while (k_g, r_j, r_i, h) is not in L_H. The case occurs when $\mathcal{A}$ gets output (A, B) without calling the second SendTag query. It implies that $\mathcal{A}$ guess the correct h. Since L_H is uniformly distributed, this only occurs with the probability no more than 2^{-k}, i.e. $\mathbf{Adv}_{\mathcal{A}}^{NS-private} \leq \epsilon + \frac{1}{2^k - q_H}$. □

5. BATCH VERIFICATION AND PERFORMANCE

We now show how to batch verify the grouping proof for the verifier. Consider a grouping proof

$$\sigma = \{r_s, (r_i, A_i, B_i) | i \in [0, n-1]\}$$

where

$$\begin{aligned} A_i &= s_iH(k_g, r_{i-1}, r_i)Y, \\ B_i &= s_iP + x_ir_sP. \end{aligned}$$

The verifier, instead of verifying the messages separately, verifies them as follows:

1. Randomly picks $v_0, v_1, ..., v_{n-1} \in \mathbb{Z}_q^*$;
2. Computes

$$\begin{aligned} h_i &= \begin{cases} H(k_g, r_{n-1}, r_0), & \text{if } i = 0. \\ H(k_g, r_{i-1}, r_i), & \text{otherwise.} \end{cases} \\ A &= \sum_{i=0}^{n-1} v_i h_i^{-1} A_i, \\ B &= \sum_{i=0}^{n-1} v_i B_i. \end{aligned}$$

3. Accepts all the tags and outputs *accept* if the equation holds

$$r_s^{-1}(B - y^{-1}A) = \sum_{i=0}^{n-1} v_i X_i;$$

else *reject*.

The batch verification scheme is correct as

$$\begin{aligned} & r_s^{-1}(B - y^{-1}A) \\ = \; & r_s^{-1}(\sum_{i=0}^{n-1} v_i B_i - y^{-1} \sum_{i=0}^{n-1} v_i h_i^{-1} A_i) \\ = \; & r_s^{-1}(\sum_{i=0}^{n-1} v_i s_i P + \sum_{i=0}^{n-1} v_i r_s X - \sum_{i=0}^{n-1} v_i s_i P) \\ = \; & \sum_{i=0}^{n-1} v_i X_i. \end{aligned}$$

In the verification phase, instead of verifying each proof generated by each tag, the reader simply adds up all the proofs of the tags and proceeds *one* verification in the scheme, which improves the efficiency of RFID applications with abundant tags as the verifier does not need to compare each tag's public key with the database.

In terms of tag performance of our scheme, the most complicated operation is scalar multiplication on an elliptic curve. Our protocol can be easily implemented in the low-cost RFID processor presented in [13]. The RFID chip is designed in a 130 nm CMOS technology. It operates in a frequency of 700 KHz. The power consumption of the processor is 13.8 μW. The number of cycles is $59,790$ per elliptic curve scalar multiplication. In our scheme a tag performs three scalar multiplications for each protocol run which means the total number of cycles is $179,370$ and the cost of time is 256 ms, which is low enough for an RFID tag.

6. CONCLUSION

We proposed a novel and efficient RFID grouping proof protocol. With elliptic curve cryptography, the proposed scheme is provably secure against active attacks such as man-in-the-middle attacks and impersonation attacks. The scheme provides narrow-strong privacy in the model described in [9]. Our scheme also allows the batch verification to reduce the workload of the verifier and it is feasible for low-cost RFID tags in terms of power consumption and processing time.

7. REFERENCES

[1] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-key cryptography for RFID-tags. In *PerCom*, pages 217–222, 2007.

[2] L. Batina, Y. K. Lee, S. Seys, D. Singelée, and I. Verbauwhede. Extending ECC-based RFID authentication protocols to privacy-preserving multi-party grouping proofs. *Personal and Ubiquitous Computing*, 16(3):323–335, 2012.

[3] L. Bolotnyy and G. Robins. Generalized "yoking-proofs" for a group of RFID tags. In *MobiQuitous*, pages 1–4, 2006.

[4] M. Burmester, B. de Medeiros, and R. Motta. Provably secure grouping-proofs for RFID tags. In *CARDIS*, pages 176–190, 2008.

[5] H. Chien, C. Yang, T. Wu, and C. Lee. Two rfid-based solutions to enhance inpatient medication safety. *Journal of Medical Systems*, 35(3):369–375, 2011.

[6] H.-Y. Chien and S.-B. Liu. Tree-based RFID yoking proof. In *NSWCTC*, pages 550–553, 2009.

[7] J. Cho, S. Yeo, S. Hwang, S. Rhee, and S. K. Kim. Enhanced yoking proof protocols for RFID tags and tag groups. In *AINA*, pages 1591–1596, 2008.

[8] D. M. Hein, J. Wolkerstorfer, and N. Felber. ECC is ready for RFID - a proof in silicon. In *SAC*, pages 401–413, 2008.

[9] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A new RFID privacy model. In *ESORICS*, pages 568–587, 2011.

[10] J. Hermans and R. Peeters. Private yoking proofs: Attacks, models and new provable constructions. In *RFIDSec*, pages 96–108, 2012.

[11] H. Huang and C. Ku. A RFID grouping proof protocol for medication safety of inpatient. *Journal of Medical Systems*, 33(6):467–474, 2009.

[12] A. Juels. "Yoking-proofs" for RFID tags. In *PerCom*, pages 138–143, 2004.

[13] Y. K. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-cost untraceable authentication protocols for RFID. In *WISEC*, pages 55–64, 2010.

[14] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede. Elliptic-curve-based security processor for RFID. *IEEE Transactions on Computers*, 57(11):1514–1527, 2008.

[15] C. Lin, Y. Lai, J. D. Tygar, C. Yang, and C. Chiang. Coexistence proof using chain of timestamps for multiple RFID tags. In *DBMAN*, pages 634–643, 2007.

[16] D. Moriyama. A provably secure offline RFID yoking-proof protocol with anonymity. In *LightSec*, pages 155–167, 2014.

[17] P. Peris-Lopez, A. Orfila, J. C. Hernandez-Castro, and J. C. A. van der Lubbe. Flaws on RFID grouping-proofs. guidelines for future sound protocols. *Journal of Network and Computer Applications*, 34(3):833–845, 2011.

[18] S. Piramuthu. On existence proofs for multiple RFID tags. In *ICPS*, pages 317–320, 2006.

[19] J. Saito and K. Sakurai. Grouping proof for RFID tags. In *AINA*, pages 621–624, 2005.

[20] S. Vaudenay. On privacy models for RFID. In *ASIACRYPT*, pages 68–87, 2007.