The ISC Int'l Journal of Information Security
ISeCure

November 2023, Volume 15, Number 3 (pp. 117–128)
http://www.isecure-journal.org

Presented at the ISCISC'2023 in Tehran, Iran.

A Lightweight RFID Grouping Proof Protocol With Forward Secrecy and Resistant to Reader Compromised Attack [**]

Fatemeh Borjal Bayatiani [1], and Hamid Mala [1,*]
[1] *Department of Information Technology Engineering, University of Isfahan, Isfahan, Iran*

ARTICLE INFO.

Keywords:
Grouping Proof, RFID Tags, Security, Forward Secrecy, Reader Compromised Attack

Type:
Research Article

doi: `10.22042/isecure.2023.418765.1030`

ABSTRACT

Today, passive RFID tags have many applications in various fields such as healthcare, transportation, asset management, and supply chain management. In some of these applications, a group of tags need to prove they are present in the same place at the same time. To solve this problem, many protocols have been proposed so far, and each of them has been able to solve some security and performance problems, but unfortunately, many of these protocols have security vulnerabilities or do not have the necessary performance to run on passive RFID tags. In this study, a secure and lightweight protocol for RFID tags grouping proof called LSGPP is proposed. In this protocol, the reader is an untrusted entity, in other words, the protocol is secure even if the reader is hijacked by an attacker. This study shows that the LSGPP protocol is secure against tracking, eavesdropping, replay, concurrency, impersonation, desynchronization, denial of service (DoS), proof forgery, message integrity, man-in-the-middle, secret disclosure, denial of proof (DoP), and unlinkability attacks, and supports anonymity and forward secrecy features. Also, in this study, the notion of RFID reader compromised attack is introduced, and it is shown that, unlike its predecessors, the LSGPP protocol is also secure against this attack. Also, using the Proverif tool, it is shown that the proposed protocol provides confidentiality and authentication features. The LSGPP protocol uses lightweight operations affordable for passive RFID tags and is shown to be compliant with the EPC C1G2 standard.

1 Introduction

Today, with the advancement of Internet of Things technology and the movement of various sectors towards smartening, automatic identification and collection of data without the need for human intervention has become very important and vital. To answer this need, several technologies have been designed

[*] Corresponding author.
[**]The ISCISC'2023 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: `f.b.bayatiani@eng.ui.ac.ir`, `h.mala@eng.ui.ac.ir`

ISSN: 2008-2045 © 2023 ISC. All rights reserved.

and implemented. One of these technologies is called RFID which has many applications in healthcare, transportation, asset management, and supply chain management. In this technology, RFID tags are attached to various objects and assets and their task is to assign a unique identity to each object.

RFID tags are divided into two categories: active and passive. Active RFID tags are battery-powered and transmit signals intermittently. But in passive RFID tags, unlike active tags, there is no battery and these tags wait to receive energy from the reader. Due to the absence of batteries in passive tags, the proposed protocol for these tags should, in addition to providing the required security, use lightweight operations so that the energy required to run the protocol is low. For passive tags, there is a global standard called EPC C1G2 [1], which specifies the capabilities and limitations of these tags.

One of the issues raised about RFID tags is the issue of grouping proof of RFID tags. In some applications that use RFID tags, two or more RFID tags are placed in a group, and proving that each group of tags is present at the same time and in the same place is a problem that researchers have tried to find suitable and secure solutions for it.

In grouping proof of RFID tags, there is usually a backup server called the verifier. This server knows all the information of groups and secret values inside tags and is a trusted entity. Another entity that exists is the reader. The reader acts as an interface between the verifier and the RFID tags. Unlike the verifier, the reader in such applications is assumed to be an untrusted entity that should not be able to obtain additional information during grouping proof sessions. Also, according to the applications of grouping proof protocols, passive RFID tags are usually used due to their lower price. Whenever a grouping proof needs to be generated, the verifier sends the required values to the reader, and then the reader generates the grouping proof in association with the tags.

The protocols designed in the field of grouping proof can be divided into two general groups, online and offline, based on the role of the verifier in the proof generation process. In online protocols, the verifier is active in the entire process of grouping proof generation. However, in offline protocols, the verifier only sends challenges to the reader and receives the grouping proof at the end, and there is no need for the verifier to be continuously present in the whole grouping proof generation process. Efficiency in offline mode is more than in online mode. Therefore, many current grouping proof protocols use offline mode [2].

In another division, grouping proof protocols can be divided into two general groups, parallel and series. In the series method, the message generated by the reader is sent to the first tag, and the calculation result is sent to the next tag, and this process continues circularly until the last tag, and the final result is returned to the reader, to generate a proof, and then send it to the verifier to check its correctness. In the parallel method, the message generated by the reader is broadcast to all tags, and the reader receives the response from all tags and generates the proof. In some parallel method protocols, a second round of communication between the reader and tags is required, after which the proof is generated. Protocols that use the serial method usually have longer run time, and parallel protocols have higher performance.

In grouping proof protocols, the generated proof must show the dependency between tags; That is, at least one of the values in the grouping proof must depend on all the tags of the group near the reader. This feature prevents a malicious reader or an attacker from eavesdropping on the communication channel between entities by removing several tags from the grouping proof. In serial protocols, each tag performs its calculations on the value generated by the previous tag, which makes the final value generated by the last tag have the attribute of dependency between tags. In parallel protocols that have one round of communication, this property is not satisfied. But in parallel protocols that have two rounds of communication to generate proof, in the second round, dependencies between tags can be established.

In this paper, a secure and lightweight protocol for grouping proof of RFID tags named LSGPP (Lightweight Secure Grouping Proof Protocol) is proposed in the presence of an untrusted reader. The protocol is resistant to tracking, eavesdropping, replay, concurrency, impersonation, desynchronization, denial of service (DoS), proof forgery, message integrity, man-in-the-middle, secret disclosure, denial of proof (DoP) and unlinkability attacks, and provides tag anonymity and forward secrecy feature. Also, in this study, reader compromised attack is proposed, and it is shown that the LSGPP protocol is also secure against this attack.

The rest of this paper is organized as follows. in Section 2, types of attacks in grouping proof protocols are described. In Section 3, we briefly review some grouping proof protocols and analyze their security. In Section 4, the LSGPP protocol is described. In Section 5, the proposed protocol is analyzed in terms of security. In Section 6, the proposed protocol is evaluated from the computational and communicational overhead, and finally, in Section 7, the conclusion of this research is stated.

2 Attacks In Grouping Proof Protocols

In the process of generating and verifying grouping proof, there is a possibility of many attacks, and the proposed protocol for grouping proof should resist these attacks, and according to the computational limitation of passive tags, use lightweight operations.

Security attacks in this field include tracking, eavesdropping, replay, concurrency, impersonation, desynchronization, denial of service (DoS), proof forgery, message integrity, man-in-the-middle, secret disclosure, denial of proof (DoP) and unlinkability attacks, and the protocols must support anonymity and forward secrecy features. Also, in this study, reader compromised attack is proposed, which will be explained below.

Forward secrecy means that if an attacker obtains the secret values of a tag, he cannot obtain the secret values used in previous sessions and track the tag in previous sessions [3].

The attack that is defined in this research is a reader compromised attack. In this attack the attacker's goal is to take control of the reader and find out that the reader is communicating with a specific group of tags in several different grouping proof sessions, and by doing this, he can determine and track the location of the group of tags at several different times. In another case of this attack, the attacker takes control of the reader, generates one or more grouping proofs with the help of tags, when all the tags of the group are together, and sends it to the verifier at the other times.

3 Related Work

In this section, we will introduce some similar protocols and some of their problems and vulnerabilities will be stated.

Sundaresan *et al.* [4] proposed a lightweight zero-knowledge serial protocol conforming to the EPC C1G2 standard, which also has the feature of forward secrecy. In this article, pseudo-random squares and quadratic residuosity are used to achieve a zero-knowledge protocol. However, this protocol has a high communication overhead because the number of messages transferred between tags and reader is high [5, 6]. Also, this protocol is vulnerable to desynchronization attack [6, 7]. In addition, in the initialization phase, the values required to generate several grouping proof sessions are placed inside the reader, and the attacker can successfully carry out the reader compromised attack by taking control of the reader, accessing its internal memory, and obtaining the value of the group ID.

Yang *et al.* [8] believe that ownership transfer and grouping proof of RFID tags can be combined to reduce computational and communication overhead. However, the protocol proposed by them is not compliant with passive tags, and the tags must have symmetric encryption and decryption, MAC, random number generation, and hash functions, which is not affordable due to the limitations of these passive tags.

Rafati *et al.* [9] proposed a parallel and offline protocol that has 2 communication rounds to generate grouping proof of RFID tags. This protocol conforms to the EPC C1G2 standard and is therefore suitable for passive tags. This protocol is vulnerable to reader compromised attack, because the reader knows the fixed ID of the group it is connected to, so the attacker can track the tag group at different times by taking control of the reader and can successfully carry out the reader compromised attack.

In the scheme of Hsi *et al.* [10], a parallel protocol is designed that has 1 round of communication to generate grouping proof. In the assumptions of this protocol, the reader is trusted to execute the protocol. In this paper, pseudonyms are used instead of tags' real IDs to prevent tracking attack using tag IDs. The article [11] has stated that this protocol is vulnerable to two attacks, denial of proof and message integrity. Another security problem of this protocol is that, contrary to the claim of the authors, this scheme does not have forward secrecy, because, if an attacker manages to obtain the hidden values inside the tag, and has eavesdropped on the previous messages transmitted between the tag and the reader, the attacker can then obtain and track the tag's previous locations by comparing the values it eavesdropped for each session and the values it can calculate by having the hidden values. In this protocol, it is assumed that each reader is specific for communication with a specific group of tags, so it is also vulnerable to the reader compromised attack.

In the scheme of Ya-li *et al.* [12], a parallel protocol called FSGP is designed, which has 1 round of communication to generate grouping proof and 1 round of communication to update the hidden values inside the tags. Contrary to the claim of its authors, this protocol is vulnerable to a tracking attack [10], because there is a correlation between the random values used for a tag in each session, with the help of which an attacker can track the tag. Also, by taking control of the reader and finding the connection between these random values, the attacker can perform a reader compromised attack. This protocol is also vulnerable to desynchronization and replay attack.

For more explanation about these attacks, refer to reference [13].

In the scheme of Shi, Zhang, and Liu [2], a parallel and offline protocol is designed, which has 2 communication rounds to generate grouping proof and 1 communication round to update the hidden values inside the tags. In this paper, to achieve the forward secrecy feature, the hidden values inside the tags are updated. Also, when updating, the last previous value of the hidden values is also kept in the verifier to prevent desynchronization attack. This is one of the advantages of this protocol because other investigated protocols in grouping proof of RFID tags that have forward secrecy are vulnerable to desynchronization attack. In this article, two protocols have been designed, in one of which it is assumed that the reader is a trusted entity, and in the other, the reader is an untrusted entity. In the protocol with the untrusted reader, in the case where the attacker has taken control of the reader, the attacker does not send the secret value update messages received from the verifier to the tags during the secret value updating phase. Rather, the attacker sends eavesdropped messages of previous sessions to track the group of tags. In this way, the attacker has a successful reader compromised attack.

In this protocol, if the attacker does not want to generate grouping proof and only wants to obtain more information, he can prevent the secret value updating messages from reaching the tags, and resend the values of the previous execution to the tags. in this case, the tags generate and send the same values as before, and an attacker can track the tags and perform a successful replay attack.

This protocol is also vulnerable to a message integrity attack, and if an attacker modifies any of the messages passed between the tags and the reader during grouping proof generation, the receiver of the message has no way of knowing whether the message is correct or not.

Also, this protocol is vulnerable to denial of proof attack. In the second round of communication, where the dependency between the tags is established in proof, the attacker can change the value sent by the reader to the tags. This will cause the value calculated by the reader at the end of the second round, which is supposed to show the dependency between the tags in the proof to have a wrong value, and the proof will not be done correctly.

In the protocol designed by Shi *et al.*, if the attacker has already eavesdropped on the message sent by the verifier at the beginning of the protocol and resend it, only the tags of the same group will respond to the received message, and thus the group can be tracked, and a group unlinkability attack occurs.

Also, in this scheme, the length of the secret values is considered to be 32 bits, which is very short for security purposes. For a more detailed study of attacks on this protocol, refer to reference [13].

In addition to the protocols described in this section, there are other protocols in this context [3, 4, 8, 10, 14–21]. In Section 5, these protocols are compared with the proposed protocol in terms of security.

4 Proposed Grouping Proof Protocol

According to the reviews on similar papers in the field of grouping proof of RFID tags, and the points mentioned in Section 3, the protocol of Shi, Zhang, and Liu [2] was considered as the basis for designing a secure protocol. Along with the disadvantages and vulnerabilities of Shi *et al.*'s protocol, this protocol also has strengths that made it used as a basis for the design of the desired protocol. One of the strengths of this protocol is that it has the forward secrecy feature, and despite having the forward secrecy feature, it is also resistant to the desynchronization attack. Additionally, this protocol can be modified to be resistant to all the vulnerabilities discussed in Section 2.

4.1 Threat Model

In the proposed protocol of this research, which is called LSGPP, like the protocol of Shi, Zhang, and Liu [2], we suppose that the channel between verifier and reader is secure, and the channel between reader and RFID tags is insecure. In this protocol, we assume that the reader is untrusted, so it should not obtain additional information from the group of tags during the protocol execution. Also, the reader can be compromised by an attacker in the reader compromised attack.

4.2 Description of the LSGPP Protocol

The LSGPP protocol is a parallel and offline grouping proof protocol that has two communication rounds to generate grouping proof and one communication round to update secret values inside RFID tags.

In the LSGPP protocol, each tag stores its current secret key value, the ID of the group it belongs to, and the timestamp of the previous session it participated in. Each reader stores its secret key. The verifier knows the reader's secret key, the previous and current secret keys of each tag, and the IDs of all groups of tags.

In this protocol, the verifier stores two old and new key values for each tag to prevent desynchronization attack. In this case, if a problem occurs in the process

Table 1. Symbols used in the LSGPP protocol

symbols	description
Gid	The ID of a group of tags
Rsk	The reader's secret key
Tsk^{new} Tsk^{old}	The secret key of a tag at the current and previous time of protocol execution
ts	Timestamp of the current protocol session
ts^{old}	Timestamp of the previous protocol session
rv	Random value generated by the verifier
vm	Values generated by the verifier
rm	Values generated by the reader
tm	Values generated by tags
pm	Values generated by the reader that are included in the proof
P	Grouping proof
$prng(m)$	A 128-bit pseudo-random number generator
$rand()$	A 128-bit random number generator
$\oplus$	Bitwise xor operation

of updating the secret key of the group of tags, and the tag fails to update its key, subsequent sessions of the protocol execution will not be affected.

Unlike the protocol of Shi *et al.* [2], which considers the length of at least 32 bits for secret values and functions, here, the values of secret keys and group identifiers are considered to be 128 bits. 128-bit XOR function, 128-bit pseudo-random number generator (PRNG), and 128-bit random number generator have also been used. Section 6 shows that the proposed protocol is compatible with the EPC C1G2 standard.

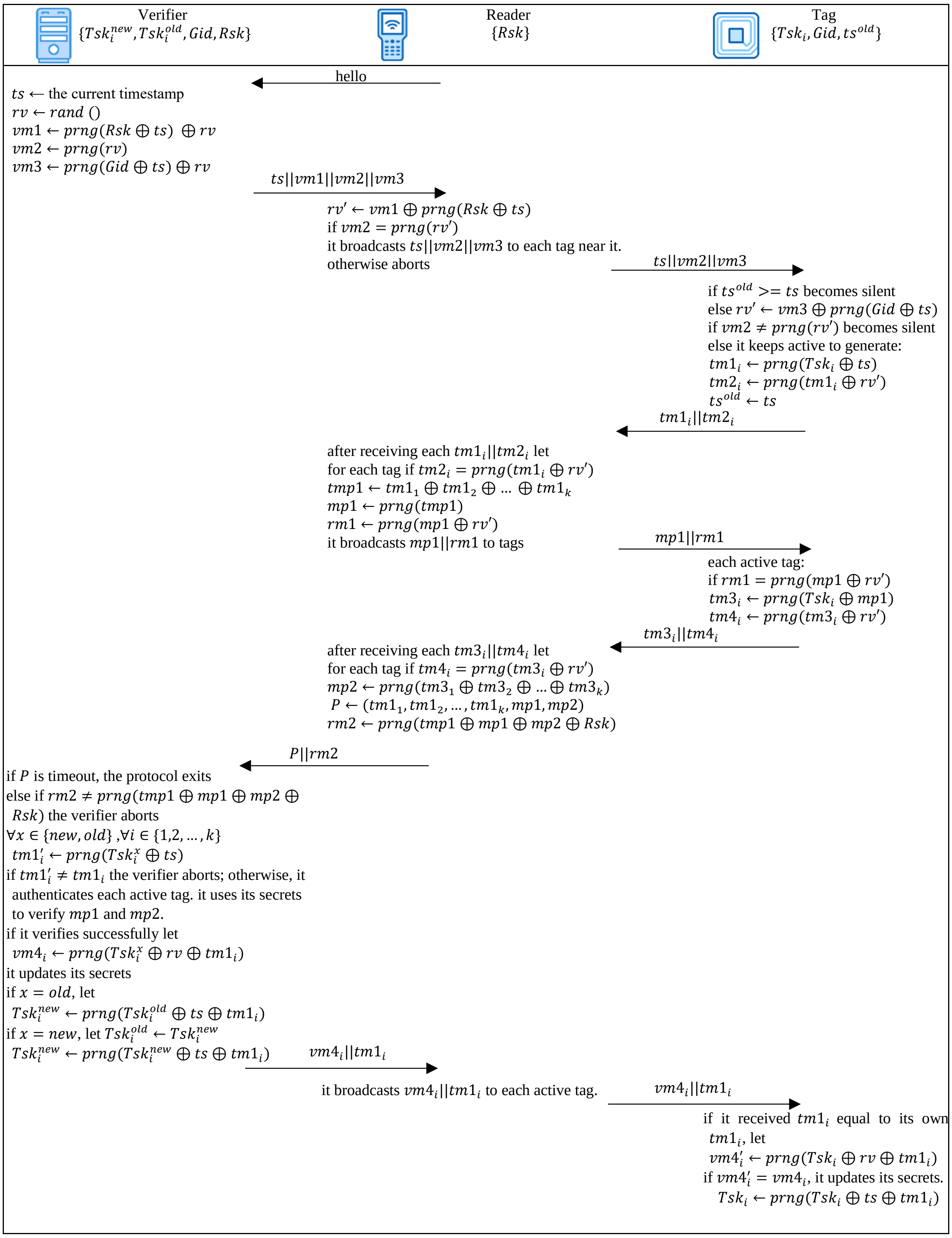
The LSGPP protocol is shown in Figure 1 and is described below. the symbols used in this protocol are shown in Table 1.

Step 1 - Initialization phase:

(1) In this step, the verifier selects and stores a tag secret key (Tsk) for each tag, a group ID (Gid) for each group of tags, and a reader secret key (Rsk) for each reader. The length of each of these values is 128 bits.
(2) Then the verifier stores the secret key of the tag and its group ID into the tag. Also, the verifier stores each reader's secret key into it.

Step 2 - Grouping proof generation phase:

(1) As shown in Figure 1, the reader sends a "hello" message to the verifier to initiate the grouping proof generation process.
(2) The verifier first obtains the current timestamp value (ts) and generates a random number rv. rv helps the entities involved in the protocol to check the integrity of the messages generated by other parties, which prevents message integrity attack, proof forgery attack, and denial of proof attack. (The timestamp in this protocol has the same function as the sequence number, and it does not mean that the tag or the reader needs to obtain the current global timestamp and compare it with the received value. Rather, it is enough for the tag to compare its internal value with the received timestamp.)
(3) Then, the verifier uses ts, rv, and the secret values of the tags and the reader, to generate the messages $vm1$, $vm2$, and $vm3$ as follows.
   - $vm1 \leftarrow prng(Rsk \oplus ts) \oplus rv$
   - $vm2 \leftarrow prng(rv)$
   - $vm3 \leftarrow prng(Gid \oplus ts) \oplus rv$
(4) In this step, the verifier starts a timer and sends $ts||vm1||vm2||vm3$ to the reader.
(5) After receiving the verifier message, the reader uses $vm1$ to compute the rv' and checks its integrity using the $vm2$ message, to make sure that this is the same rv that the verifier has generated for this session. In fact, by checking the integrity of this value, the reader ensures that the values of ts, Rsk, and rv, which were used in the construction of the $vm1$ message, are correct. If this value is true, the protocol execution continues. Otherwise, execution of this grouping proof session terminates.
(6) If the protocol continues, the reader broadcasts $ts||vm2||vm3$ to all tags near it.
(7) Each tag receiving this message compares the received time value with the timestamp stored from the previous session, and the received timestamp value must be greater than the timestamp stored in the tag. This comparison helps the tag not to respond to duplicate and old messages. If the timestamp value of the received message is not fresh, the tag aborts.
(8) If the timestamp value is fresh, the tag uses $vm3$ to calculate the rv' and checks the correctness of this value using the $vm2$ message. If this value is true, the tag remains active and continues to run. The non-equality of $vm2$ and $prng(rv')$ can be due to the generation of these values by the attacker, or this could be because the tag that received this value was not in the target group, and has a different Gid.
(9) In this step, the $i-th$ tag calculates $tm1_i$ and $tm2_i$ as in the following equations. It also updates the timestamp stored within itself.
   - $tm1_i \leftarrow prng(Tsk_i \oplus ts)$
   - $tm2_i \leftarrow prng(tm1_i \oplus rv')$
(10) Each tag sends $tm1_i||tm2_i$ to the reader.
(11) When the reader receives $tm1_i||tm2_i$ ($i \in \{1, 2, 3, \ldots, k\}$) from each active tag, it checks the validity of $tm2_i$ for each tag (k is the number of active tags in the group).
(12) Then, with the help of the correct values received from the tags, the reader calculates $mp1$ and $rm1$ as follows and broadcasts these values



Verifier $\{Tsk_i^{new}, Tsk_i^{old}, Gid, Rsk\}$

Reader $\{Rsk\}$

Tag $\{Tsk_i, Gid, ts^{old}\}$

hello

$ts \leftarrow$ the current timestamp
$rv \leftarrow rand\ ()$
$vm1 \leftarrow prng(Rsk \oplus ts) \oplus rv$
$vm2 \leftarrow prng(rv)$
$vm3 \leftarrow prng(Gid \oplus ts) \oplus rv$

$ts||vm1||vm2||vm3$

$rv' \leftarrow vm1 \oplus prng(Rsk \oplus ts)$
if $vm2 = prng(rv')$
it broadcasts $ts||vm2||vm3$ to each tag near it.
otherwise aborts

$ts||vm2||vm3$

if $ts^{old} >= ts$ becomes silent
else $rv' \leftarrow vm3 \oplus prng(Gid \oplus ts)$
if $vm2 \neq prng(rv')$ becomes silent
else it keeps active to generate:
$tm1_i \leftarrow prng(Tsk_i \oplus ts)$
$tm2_i \leftarrow prng(tm1_i \oplus rv')$
$ts^{old} \leftarrow ts$

$tm1_i||tm2_i$

after receiving each $tm1_i||tm2_i$ let
for each tag if $tm2_i = prng(tm1_i \oplus rv')$
$tmp1 \leftarrow tm1_1 \oplus tm1_2 \oplus \ldots \oplus tm1_k$
$mp1 \leftarrow prng(tmp1)$
$rm1 \leftarrow prng(mp1 \oplus rv')$
it broadcasts $mp1||rm1$ to tags

$mp1||rm1$

each active tag:
if $rm1 = prng(mp1 \oplus rv')$
$tm3_i \leftarrow prng(Tsk_i \oplus mp1)$
$tm4_i \leftarrow prng(tm3_i \oplus rv')$

$tm3_i||tm4_i$

after receiving each $tm3_i||tm4_i$ let
for each tag if $tm4_i = prng(tm3_i \oplus rv')$
$mp2 \leftarrow prng(tm3_1 \oplus tm3_2 \oplus \ldots \oplus tm3_k)$
$P \leftarrow (tm1_1, tm1_2, \ldots, tm1_k, mp1, mp2)$
$rm2 \leftarrow prng(tmp1 \oplus mp1 \oplus mp2 \oplus Rsk)$

$P||rm2$

if P is timeout, the protocol exits
else if $rm2 \neq prng(tmp1 \oplus mp1 \oplus mp2 \oplus Rsk)$ the verifier aborts
$\forall x \in \{new, old\}, \forall i \in \{1,2,\ldots,k\}$
$tm1_i' \leftarrow prng(Tsk_i^x \oplus ts)$
if $tm1_i' \neq tm1_i$ the verifier aborts; otherwise, it authenticates each active tag. it uses its secrets to verify $mp1$ and $mp2$.
if it verifies successfully let
$vm4_i \leftarrow prng(Tsk_i^x \oplus rv \oplus tm1_i)$
it updates its secrets
if $x = old$, let
$Tsk_i^{new} \leftarrow prng(Tsk_i^{old} \oplus ts \oplus tm1_i)$
if $x = new$, let $Tsk_i^{old} \leftarrow Tsk_i^{new}$
$Tsk_i^{new} \leftarrow prng(Tsk_i^{new} \oplus ts \oplus tm1_i)$

$vm4_i||tm1_i$

it broadcasts $vm4_i||tm1_i$ to each active tag.

$vm4_i||tm1_i$

if it received $tm1_i$ equal to its own $tm1_i$, let
$vm4_i' \leftarrow prng(Tsk_i \oplus rv \oplus tm1_i)$
if $vm4_i' = vm4_i$, it updates its secrets.
$Tsk_i \leftarrow prng(Tsk_i \oplus ts \oplus tm1_i)$

Figure 1. The LSGPP protocol

to each active tag of the group.

- $tmp1 \leftarrow tm1_1 \oplus tm1_2 \oplus \ldots \oplus tm1_k$
- $mp1 \leftarrow prng(tmp1)$
- $rm1 \leftarrow prng(mp1 \oplus rv')$

(13) Each active tag uses $rm1$ to check the validity of $mp1$, and if it is true, calculates $tm3_i$ and $tm4_i$ as follows. It then sends $tm3_i||tm4_i$ to the reader.

- $tm3_i \leftarrow prng(Tsk_i \oplus mp1)$
- $tm4_i \leftarrow prng(tm3_i \oplus rv')$

(14) For each message received from group tags, the reader checks $tm3_i$ using $tm4_i$.

(15) Then, the reader calculates $mp2$, the grouping proof of P and $rm2$ as follows:

- $mp2 \leftarrow prng(tm3_1 \oplus tm3_2 \oplus \ldots \oplus tm3_k)$
- $P \leftarrow (tm1_1, tm1_2, \ldots, tm1_k, mp1, mp2)$
- $rm2 \leftarrow prng(tmp1 \oplus mp1 \oplus mp2 \oplus Rsk)$

Step 3 - Verification phase:

(1) As shown in Figure 1, the verifier first checks that the protocol execution time has not expired. If this time is over, the protocol is terminated.

(2) Next, the verifier checks the correctness of $rm2$, to ensure that the received grouping proof has not been changed during transmission. If this value is correct, the verifier must then authenticate the tags.

(3) To authenticate the tags, the verifier checks the integrity of the message $tm1_i$ ($i \in \{1, 2, 3, \ldots, k\}$) using the old and new secret key values of the tags. If this value is correct for all the active tags of the group, the verifier can verify the identity of all the active tags of the group. Otherwise, the protocol encounters an error and terminates.

(4) Now, the verifier uses the secret values of tags and $tm1_i$s to check the correctness of $mp1$ and $mp2$. If these values are correct, the verifier accepts the correctness of the grouping proof and enters the secret values updating phase. Otherwise, the protocol encounters an error and terminates.

Step 4 - Secret values updating phase:

(1) As shown in Figure 1, the verifier calculates the value of $vm4_i$ for each tag using the current key stored in the tag, say Tsk_i^x, where $x \in \{new, old\}$, as shown in the following equation.

- $vm4_i \leftarrow prng(Tsk_i^x \oplus rv \oplus tm1_i)$

(2) The current key stored in the tag is stored in Tsk_i^{old}. Then, it updates the key for each tag in the group. To do this, it uses the current key in the tag to calculate Tsk_i^{new}. Also, it updates the timestamp and $tm1_i$, as follows, and stores it.

- $Tsk_i^x \leftarrow prng(Tsk_i^{new} \oplus ts \oplus tm1_i)$

(3) Then, the verifier generates the message $vm4_i||tm1_i$ for each active tag and sends it to the reader. The reader broadcasts these incoming messages. The sent message includes the value of $tm1_i$ such that each tag knows which message is related to it.

(4) After each active tag receives $vm4_i||tm1_i$ messages, it finds its corresponding message by comparing the received and stored $tm1_i$ value.

(5) It then checks that the value of $vm4_i$ is correct. If it is true, the tag updates its secret key value.

5 Security Analysis Of The LSGPP

In this section, the LSGPP protocol will be investigated in terms of security and efficiency. First, the proposed protocol will be examined in terms of security against the attacks specified in Section 2, and it is found that this protocol is resistant to all these attacks. In the following, the Proverif tool is used to prove the confidentiality and authentication features.

5.1 Informal Security Analysis

In the LSGPP protocol, two communication rounds between the reader and the group tags are used to generate grouping proof. if one communication round was used to generate a proof, the reader could remove some tags from the proof and delete the $tm1_i$ from $mp1$ and P. But when there are two rounds of communication, the probability that the reader can do this is very low, and it is equal to the fact that the reader can guess the $tm3_i$ of the tag that it wants to remove from the group among the received $tm3_i$s.

In this section, the security of LSGPP protocol against all attacks described in Section 2 is discussed. According to the threat model specified in Section 4.1, the reader is an untrusted entity.

- tracking attack: The timestamp value sent by the verifier for tags in the LSGPP protocol, and the use of PRNG, prevent this attack. An attacker cannot use messages from previous sessions to track a tag's location because each tag only responds to fresh messages with a newer timestamp than the previous session. Also, due to the use of the PRNG, the attacker cannot find the values of the group ID or the secret key of the tag, which would help them generate a valid message with a new timestamp.
- Anonymity: In the LSGPP protocol, the identity of a tag is the secret key of the tag, which cannot be accessed by analyzing the values generated by the tag. In all steps of the protocol, when the secret key needs to be sent, the value of the secret key is XORed with another value and then entered into the PRNG, which makes it

impossible for the attacker to access the secret key.

- Eavesdropping attack: In this protocol, as previously stated, all important values, including the secret key of the tags, the secret key of the reader, and the group ID, are not sent obviously in any part of the protocol execution. Rather, wherever these values need to be sent, with the help of XOR and PRNG, we hide them from the attacker's access.
- Replay attack: In the LSGPP protocol, tags do not respond to duplicate messages because they contain old timestamps, and they do not update their internal secrets. For this reason, if the attacker has intercepted several messages, he cannot reuse them.
- Concurrency attack: In the LSGPP protocol, at the beginning of the protocol execution, the verifier activates a timer, and does not generate a $ts||vm1||vm2||vm3$ message for the same group of tags to any other reader until the timer expires.
- Impersonation Attack: Without having the group ID and reader key and tags key, the attacker cannot generate valid $vm1$, $vm3$, or $vm4_i$ values, and spoof the identity of the verifier. The attacker needs the secret key of the tag to impersonate the tag, and the messages generated without the valid secret key of the tag are not accepted by the verifier in the verification phase. To impersonate the reader, the attacker needs its secret key, and if the attacker does not have this value, he cannot obtain the correct rv value. For this reason, tags notice a problem in the execution of the protocol in the second round of grouping proof and do not give a response to the reader.
- Desynchronization attack: In the article [2], a solution is presented to resist this attack. In this paper, the verifier stores both the tag's previous key value and the value to be updated. Because of this, if the attacker intercepts the update messages, there will be no problem with subsequent proofs. The same method is used in this protocol.
- Denial Of Service (DoS) attack: In grouping proof protocols where there is a phase to update the secret values of tags, the vulnerability against this attack seems to be high. In fact, the attacker causes a mismatch between the group tags and the verifier by blocking the update messages because the key in the tags is not the same as the key stored in the verifier. But [2] provides a solution for this problem. In this paper, the verifier stores both the tag's previous key value and the value to be updated. Because of this, if the attacker intercepts the update messages, there will be no problem with subsequent proofs. The same method is used in the LSGPP.
- Proof forgery attack: In the LSGPP protocol, due to the limited time for grouping proof generation, the attacker cannot generate valid proof without having the secret values of all group tags. Also, due to the existence of a timestamp in all the generated proofs, the attacker cannot use the previous proofs.
- Message integrity attack: In the LSGPP protocol, for each proof generation session, the verifier generates and sends a random rv value that can only be accessed by the reader and authorized group of tags, and in the continuation of the session, any entity with the help of this value can check the correctness of the message.
- Man-in-the-middle attack: In the proposed protocol, due to the presence of rv, the tag and the reader can notice the problem in the received messages.
- Secret disclosure attack: In the LSGPP protocol, tags only respond to messages whose timestamp is fresh, and the correct timestamp and group ID values are used in the construction of the $vm3$ message. Therefore, an attacker can never force a tag to send a response by altering the messages, which can gain information from the analysis of that response.
- Denial Of Proof (DoP) attack: In the proposed protocol, the dependency between the tags in the proof occurs in the second round of communication between the tags and the reader, and because of rv, the attacker cannot change the messages between the tag and the reader, and only the correct messages confirmed by the receiver and then the response is generated. So, the proof sent to the verifier has the correct values. If an attacker modifies any message during the protocol execution, the receiver will notice the change and not accept it.
- Unlinkability attack: In the proposed protocol, no tag responds to duplicate $ts||vm2||vm3$ message. The tags of the same group do not respond to the repeated message of the attacker because the timestamp in the packet is old. The tags of other groups also do not respond to old messages since either the timestamp is old or the group ID is irrelevant. Therefore, the attacker cannot track the tag group in this way.
- Forward secrecy: To satisfy the forward secrecy feature in the LSGPP protocol, each tag's key is updated at the end of the grouping proof generation session. Therefore, the attacker cannot obtain the current key of the tag and use it to

generate the values generated in previous sessions and track the tag. By obtaining the tag's current key, an attacker cannot regenerate any of the values generated by the tag in previous sessions, because the previous keys of the tag were used in their generation, and obtaining the previous key of the tag is possible only by breaking the PRNG function.

- Reader compromised attack: In this attack, the reader is assumed malicious. One of the goals of the attacker in the reader compromised attack is to take control of the reader and find out that it is communicating with a specific group of tags in several grouping proof sessions, and in this way, track the group. One of the features of the LSGPP protocol is that the reader is not given any information about the tags and the group it is communicating with. Moreover, in all messages sent to the reader, secret parameters are hidden from the reader using the XOR function and PRNG. It can be said that the reader's knowledge of the group of tags with which it is communicated is no different from the attacker's knowledge which only listens to the communication between entities. Another goal of the reader compromised attack is for the attacker to generate multiple grouping proofs without the help of the verifier, and send them to the verifier at the required time. But in the LSGPP protocol, the reader cannot generate any proof without the help of the verifier, because the reader needs the group ID to start the proof generation process, which is never provided to the reader, and it is not possible to reach this value by analyzing the messages.

According to the contents mentioned in Section 3, and the review carried out on the LSGPP, this protocol is compared against other schemes mentioned in Section 2, in Table 2.

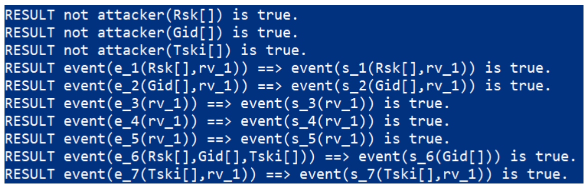
5.2 Formal Security Analysis

Using the Proverif tool [20], it is possible to check confidentiality and authentication in security protocols. For this purpose, the LSGPP protocol was simulated with a special language defined for this tool.

Figure 2 shows the output of the Proverif tool. According to Figure 2, the LSGPP protocol satisfies the confidentiality and authentication features. That is, the values of the reader's secret key, the tag's secret key, and the group ID are not available to the attacker. All messages transferred between entities are authenticated, and the receiver of the message can be sure that the sender of the message is who he expects.

Table 2. Security comparison of protocols similar to LSGPP. ✓: It is resistant to the specified attack, ×: Vulnerable to the specified attack, -: This attack has not been checked, $*$: Description of these attacks can be accessed in reference [13]

	Serial / Parallel	Tracking attack	Anonymity	Eavesdropping attack	Replay attack	Concurrency attack	Impersonation attack			Desynchronization attack	Denial of service (DOS)	Proof forgery attack	Message integrity attack	Man-in-the-middle attack	Secret disclosure attack	Denial of proof (DOP)	Unlinkability attack	Forward secrecy	Reader compromised attack
							Tag	Reader	Verifier										
[4]	S	✓	✓	-	✓	✓	✓	✓	-	×	✓	✓	✓	-	-	✓	-	✓	×*
[6]	S	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	✓	-	-	✓	-	-	×*
[22]	S	×*	-	-	×	-	×	-	-	×	×	-	-	-	-	-	-	-	×*
[14]	S	-	✓	-	×	-	×	×	-	-	-	-	✓	-	-	×	-	-	×*
[23]	S	✓	✓	✓	✓	×	✓	✓	-	✓	-	-	×	-	-	×	-	-	×*
[15]	S	×	×	-	✓	-	✓	✓	-	-	✓	✓	-	-	-	-	-	×	×*
[9]	P	✓	-	-	✓	-	✓	✓	-	-	-	✓	✓	-	-	✓	-	×*	×*
[16]	P	✓	✓	-	✓	-	✓	-	-	-	-	-	-	✓	-	×*	-	-	×*
[11]	P	✓	-	-	✓	✓	✓	✓	-	✓	-	-	✓	-	-	✓	-	-	×*
[5]	P	✓	-	-	✓	✓	✓	✓	-	✓	-	-	✓	-	-	✓	-	×	×*
[17]	P	×*	-	-	×	-	×	×	-	-	-	-	-	-	-	-	-	×	×*
[18]	P	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-	✓	-	-	×	-	-	×*
[19]	P	✓	✓	✓	✓	-	✓	-	-	✓	✓	-	-	✓	✓	-	-	✓	×*
[3]	P	✓	✓	-	×	✓	×	×	-	×	-	-	×	-	-	×	-	✓	×*
[2]	P	×*	✓	✓	×*	✓	✓	-	✓	✓	✓	✓	×*	✓	✓	×*	×*	✓	×*
[10]	P	✓	✓	-	✓	✓	✓	✓	-	✓	-	✓	×	-	-	×	-	×*	×*
[12]	P	×	-	-	×*	-	-	-	-	×*	-	✓	-	✓	-	-	-	✓	×*
LSGPP	P	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓



```
RESULT not attacker(Rsk[]) is true.
RESULT not attacker(Gid[]) is true.
RESULT not attacker(Tski[]) is true.
RESULT event(e_1(Rsk[],rv_1)) ==> event(s_1(Rsk[],rv_1)) is true.
RESULT event(e_2(Gid[],rv_1)) ==> event(s_2(Gid[],rv_1)) is true.
RESULT event(e_3(rv_1)) ==> event(s_3(rv_1)) is true.
RESULT event(e_4(rv_1)) ==> event(s_4(rv_1)) is true.
RESULT event(e_5(rv_1)) ==> event(s_5(rv_1)) is true.
RESULT event(e_6(Rsk[],Gid[],Tski[])) ==> event(s_6(Gid[])) is true.
RESULT event(e_7(Tski[],rv_1)) ==> event(s_7(Tski[],rv_1)) is true.
```

Figure 2. Proverif tool output

6 Computational And Communicational Analysis

In the protocol of Shi, Zhang, and Liu [2], which was considered the basis of the LSGPP protocol, two mechanisms are used to improve the efficiency of the protocol. In the LSGPP protocol, the same mechanisms have been used to improve the efficiency of the protocol aims.

In the first mechanism, which is to reduce the overhead of the reader in the grouping proof generation process, only tags that are members of the group for which the proof is to be generated respond to the reader, and the rest of the tags around the reader remain silent.

The second mechanism aims to reduce tag computational overhead in the process of updating secret values. The verifier sends the value of $tm1_i$ along with $vm4_i$ for the group of tags to update the tag's key. This makes it so that the tag does not need to check all the received $vm4_i$ values to find the correct corresponding value, it can find the corresponding $vm4_i$ by comparing the received $tm1_i$ with the $tm1_i$ value generated by the tag in the same session.

One of the important issues in the protocols related to passive RFID tags is the compliance of the designed protocol with the EPC C1G2 standard. In this standard, it is stated that there is a 16-bit pseudorandom number generator function in passive tags, but 16 bits is too small for security purposes. Also, according to this standard, a maximum of about 3000 gates are available to security protocol designers to provide security [3, 9, 10].

Therefore, some researchers have focused on building pseudorandom number generator functions with sufficient security, fewer gates, and more bits, and have been able to construct a 128-bit pseudorandom number generator function with less than 1500 gates [4]. The authors of [21] have been able to design a 128-bit pseudorandom number generator function with 1435 gates that complies with the EPC C1G2 standard. Therefore, in the LSGPP protocol, the 128-bit PRNG function is used in RFID tags.

hash function has been used in some grouping proof protocols of RFID tags. However, the 128-bit MD5 hash function requires 8001 gates to implement [24, 25], which is not suitable for passive tags. For this reason, some researchers have tried to design hash functions according to the EPC C1G2 standard [24]. The 128-bit hash function designed by the authors of [24] provides 64-bit security against collision and second preimage attack and requires 2392 gates to implement [26].

In Shi, Zhang and Liu's scheme [2], XOR, hash, and PRNG are used for security. But in this research, to reduce the number of required gates, and to comply with the EPC C1G2 standard in the LSGPP protocol, the hash function is not used, and to bring the security to the desired and sufficient level, the tag only uses XOR and PRNG. Table 3 compares the average number of operations required for a single protocol execution for a group with n tags.

As shown in Table 3, the calculations for each tag are almost equal to the base protocol. However, in the LSGPP protocol, unlike the base protocol, the number of operations performed in the reader is increased due to improved security. For the verifier, in the LSGPP protocol, the number of operations has been reduced. For a better comparison of these two protocols, the average number of calculations required for each entity is shown in Table 4 for the number of tags 10, 50, and 100.

Table 3. Comparison of the average number of operations required to execute the protocol. X: XOR, H: Hash, P:PRNG, C: Comparison, R: Rand

	Tag	Reader	Verifier
[2]	$12X + 5H + 3P + (2 + n/2)C$	$(2n - 2)X + 2H$	$(10n + 2)X + (2n + 6)H + (2n + 1)P + (4 + (n(n + 1))/2)C$
LSGPP	$11X + 9P + (4 + n/2)C$	$(4n + 4)X + (2n + 6)P + (2n + 1)C$	$(8n + 11/2)X + (3n + 15/2)P + ((n(n + 1))/4 + 9/2)C + 1R$

Table 4. Comparison of the average number of operations required for 10, 50, and 100 tags per group. X: XOR, H: Hash, P:PRNG, C: Comparison, R: Rand

	Number of group tags	[2]	LSGPP
Tag	10	$12X + 5H + 3P + 7C$	$11X + 9P + 9C$
	50	$12X + 5H + 3P + 27C$	$11X + 9P + 29C$
	100	$12X + 5H + 3P + 52C$	$11X + 9P + 54C$
Reader	10	$18X + 2H$	$44X + 26P + 21C$
	50	$98X + 2H$	$204X + 106P + 101C$
	100	$198X + 2H$	$404X + 206P + 201C$
Verifier	10	$102X + 26H + 21P + 59C$	$85.5X + 37.5P + 32C + 1R$
	50	$502X + 106H + 101P + 1279C$	$405.5X + 157.5P + 642C + 1R$
	100	$1002X + 206H + 201P + 5054C$	$805.5X + 307.5P + 2529.5C + 1R$

Table 5. Comparison of the communication overhead of a single protocol execution

	Communication between tags and reader	Communication between reader and verifier
[2]	$6n + 3$	$5n + 4$
LSGPP	$6n + 5$	$3n + 7$

Table 5 shows the comparison of the communication overhead required for a single execution of the protocol for a group with n tags. As shown in Table 5, in the LSGPP protocol, the communication overhead in the communication channel between the verifier and the reader has been significantly reduced compared to Shi's protocol. The number of messages transmitted in the communication channel between the tags and the reader in the LSGPP protocol has increased by only 2 128-bit messages compared to Shi's protocol.

Passive tags have storage capabilities between 1KB and 64KB [4]. In the LSGPP protocol, for each tag we need to store the tag key, the group ID, and the last session timestamp, each of which is 128 bits. So, each tag requires 384 bits of storage, which is suitable for passive tags.

7 Conclusion

In recent years, many papers have been published in the field of grouping proof of RFID tags. Each of these papers have tried to solve part of the security and privacy problems in this application of RFID tags. Passive RFID tags are usually used in grouping proof protocols. One of the things that is important in the field of passive RFID tags is that due to the limitations of these tags, the protocols designed for them should use lightweight operations and comply with the EPC C1G2 standard.

In this research, in line with the previous work done in this field, an effort was made to provide a secure protocol compliant with the EPC C1G2 standard for the grouping proof of RFID tags, which is secure against known attacks in this field such as tracking, eavesdropping, replay, concurrency, impersonation, desynchronization, denial of service, proof forgery, message integrity, man-in-the-middle, secret disclosure, denial of proof and unlinkability attacks, and provides anonymity and forward secrecy features. Also, in this study, reader compromised attack is proposed with the assumption that the reader is malicious, and it is shown that the LSGPP protocol is also secure against this attack. Next, using the Proverif tool, the confidentiality and authentication features in the LSGPP protocol are also investigated. Next, the proposed protocol was examined in terms of the number of gates, memory, and the number of operations required for each tag, and it was shown that this protocol is suitable for passive tags and complies with the EPC C1G2 standard.

Currently, many cryptographic algorithms are based on hard problems that current powerful computers are unable to solve. However, with the emergence of quantum computers, which have a much higher processing speed than current computers, there is a great need for cryptographic primitives and protocols that are secure against quantum computers. So, designing post-quantum RFID grouping proof protocol will be a noticeable future line of research.

References

[1] Radio-frequency identity protocols generation-2 uhf rfid standard. Technical report, GS1, 2018.

[2] Zhicai Shi, Xiaomei Zhang, and Jin Liu. The lightweight rfid grouping-proof protocols with identity authentication and forward security. *Wireless Communications and Mobile Computing*, 2020:1–12, 2020.

[3] Sarah Abughazalah, Konstantinos Markantonakis, and Keith Mayes. Two rounds rfid grouping-proof protocol. In *2016 IEEE International Conference on RFID (RFID)*, pages 1–14. IEEE, 2016.

[4] Saravanan Sundaresan, Robin Doss, and Wanlei Zhou. Zero knowledge grouping proof protocol for rfid epc c1g2 tags. *IEEE Transactions on Computers*, 64(10):2994–3008, 2015.

[5] Vanya Cherneva and Jerry L Trahan. A secure and efficient parallel-dependency rfid grouping-proof protocol. *IEEE Journal of Radio Frequency Identification*, 4(1):14–23, 2020.

[6] Vanya Cherneva and Jerry L Trahan. Serial-dependency grouping-proof protocol for rfid epc gen2 tags. *IEEE Journal of Radio Frequency Identification*, 4(2):159–169, 2020.

[7] Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. Automated identification of desynchronisation attacks on shared secrets. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*, pages 406–426. Springer, 2018.

[8] Ming-Hour Yang, Yu-Shan Hsu, and Hung-Yu Ko. Dispute resistance multilayered rfid partial ownership transfer with blockchain. *IEEE Access*, 10:123634–123650, 2022.

[9] Zahra Rafati. A secure and scalable grouping proof for lightweight rfid tags. MSc Thesis, Faculty of Computer Engineering ,University of Isfahan, 2021.

[10] Cheng-Ter Hsi, Yuan-Hung Lien, Jung-Hui Chiu, and Henry Ker-Chang Chang. Solving scalability problems on secure rfid grouping-proof protocol. *Wireless Personal Communications*, 84:1069–1088, 2015.

[11] Vanya Cherneva and Jerry L Trahan. Grouping proofs for dynamic groups of rfid tags: A secure and scalable protocol. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0097–0103. IEEE, 2020.

[12] Liu Ya-li, Qin Xiao-lin, Li Bo-han, and Liu Liang. A forward-secure grouping-proof protocol for multiple rfid tags. *International Journal of Computational Intelligence Systems*, 5(5):824–833, 2012.

[13] Fatemeh Borjal Bayatiani. A lightweight rfid grouping proof protocol with forward secrecy and resistant to reader compromised attack. MSc Thesis, Faculty of Computer Engineering, University of Isfahan, 2022.

[14] Lin Qiping, Zhang Fangguo, et al. ecc-based grouping-proof rfid for inpatient medication safety. 2012.

[15] Cunqing Ma, Jingqiang Lin, Yuewu Wang, and Ming Shang. Offline rfid grouping proofs with trusted timestamps. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 674–681. IEEE, 2012.

[16] Shu Cheng, Vijay Varadharajan, Yi Mu, and Willy Susilo. An efficient and provably secure

rfid grouping proof protocol. In *Proceedings of the Australasian computer science week multi-conference*, pages 1–7, 2017.

[17] Junichiro Saito and Kouichi Sakurai. Grouping proof for rfid tags. In *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, volume 2, pages 621–624. IEEE, 2005.

[18] Mike Burmester and Jorge Munilla. An anonymous rfid grouping-proof with missing tag identification. In *2016 IEEE International Conference on RFID (RFID)*, pages 1–7. IEEE, 2016.

[19] Ömer Aydin, Gökhan Dalkiliç, and Cem Kösemen. A novel grouping proof authentication protocol for lightweight devices: Gpapxr+. *Turkish Journal of Electrical Engineering and Computer Sciences*, 28(5):3036–3051, 2020.

[20] Proverif: Cryptographic protocol verifier in the formal model., April 2023.

[21] HangRok Lee and DoWon Hong. The tag authentication scheme using self-shrinking generator on rfid system. *International Journal of Information and Communication Engineering*, 2(6):1242–1247, 2008.

[22] Hsieh-Hong Huang and Cheng-Yuan Ku. A rfid grouping proof protocol for medication safety of inpatient. *Journal of medical systems*, 33:467–474, 2009.

[23] Zhibin Zhou, Pin Liu, Qin Liu, and Guojun Wang. An anonymous offline rfid grouping-proof protocol. *Future Internet*, 10(1):2, 2018.

[24] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. *Journal of cryptology*, 26:313–339, 2013.

[25] Charalampos Manifavas, George Hatzivasilis, Konstantinos Fysarakis, and Konstantinos Rantos. Lightweight cryptography for embedded systems–a comparative analysis. In *International Workshop on Data Privacy Management*, pages 333–349. Springer, 2013.

[26] Seyed Farhad Aghili and Hamid Mala. New authentication/ownership transfer protocol for rfid objects. *Journal of Information Security and Applications*, 49:102401, 2019.

Fatemeh Borjal Bayatiani received her B.Sc. and M.Sc degrees in information technology (IT) engineering from University of Isfahan (UI) in 2019 and 2021, respectively. She is currently a Ph.D. student in IT engineering (information security) at UI. Her research interests include Security Protocols, Network Security, and Post-Quantum Cryptography.

Hamid Mala received his B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Isfahan University of Technology (IUT) in 2003, 2006, and 2011, respectively. He joined the Department of Information Technology Engineering, University of Isfahan (UI) in September 2011, as an assistant professor. He is currently with the Faculty of Computer Engineering, UI, as an associate professor. His research interests include the Design and Cryptanalysis of Block Ciphers, Cryptographic Protocols, and Secure Multiparty Computation.