A New Secure and Efficient Ownership Transfer Protocol based on Quadric Residue and Homomorphic Encryption

Farokhlagha Moazami[1] and Masoumeh Safkhani[2]

[1] Cyberspace Research Institute, Shahid Beheshti University,Tehran, Iran
[2] Computer Engineering Department, Shahid Rajaee Teacher Training University,
Tehran, Iran, Postal code: 16788-15811, Tel/fax:+98-21-22970117,
Safkhani@srttu.edu

Abstract. In systems equipped with radio frequency identification (RFID) technology, several security concerns may arise when the ownership of a tag should be transferred from one owner to another, e.g., the confidentiality of information related to the old owner or the new owner. Therefore, this transfer is usually done via a security protocol called the ownership transfer protocol. If the ownership of several things together transmitted from one owner to another during a single session, the protocol is referred to as the group ownership transfer protocol.

Lee *et al.* recently proposed a new group ownership transfer protocol by using cloud server, as a trusted third-party, and based on homomorphic encryption and quadratic residue. In this paper, at first, we explain some important security attacks against this recently proposed RFID group ownership transfer protocol. The success probability of any attack that is presented in this paper is 1 and the complexity is just a run of the protocol. Zhu *et al.* also in order to provide simultaneous transfer of group of tags in multi-owner environment proposed a lightweight anonymous group ownership transfer protocol. In this paper we show that it suffers from desynchronization attack. The success probability of this attack is "1" and its complexity is only five runs of group ownership transfer protocol.

In addition, to overcome the Lee *et al.* protocol security weaknesses, we present a new group ownership transfer protocol which is resistant against all known active and passive attacks, including the attacks presented in this paper. The provided security proof through informal methods and also formal methods such as Barrows-Abadi-Needham logic and Scyther tool show the proposed protocol's security correctness.

Keywords: RFID, Ownership Transfer Protocol, Quadratic Residue, Secret Disclosure Attack, Traceability Attack, Scyther, Barrows-Abadi-Needham

1 Introduction

Ownership transfer of an object form an owner to another could be a common practice in many applications of RFID technology in the Internet of Things

(IoT), E-health and logistics for instants. However, it also has its own security concerns. A general solution for this issue is the design and implementation of ownership transfer protocols, or in a comprehensive manner, group ownership transfer protocols. The most important features of a secure ownership transfer protocol are the forward untraceability and backward untraceability properties, which are defined as follows:

- Forward untraceability: means that the old owner should not be able to read future data.
- Backward untraceability: means that the new owner should not be able to read old data.

In this direction, recently in [23], in order to provide simultaneous transfer of group of tags in multi-owner environment, a novel lightweight group ownership transfer protocol was proposed according to EPC C1 G2 standard. Their designers claimed, their proposed protocol can satisfy security properties including forward/backward security, anonymity/untraceability and resistance to replay and desynchronization attack. However, in this paper we show that their protocol is not resistant against desynchronization attack.

Moreover, recently [14] proposed a novel group ownership transfer protocol. They also claimed that their protocol provides data privacy, backward secrecy, forward secrecy, ownership privacy, and group ownership integrity. They also have shown the correctness of their scheme using Barrows-Abadi-Needham logic, which is a manual formal method to evaluate the security of a protocol. They used cloud computing in their scheme that leads their scheme to be ubiquitous authentication and also used homomorphic encryption and quadratic residue that made their protocol processes on encrypted data to be efficient. In this paper, we investigate the Lee *et al.* protocol security pitfalls and prove its security vulnerabilities. To overcome the security vulnerabilities of Lee *et al.* protocol, we also propose a new secure group ownership transfer protocol and prove its correctness by using Barrows-Abadi-Needham logic and Scyther tool, which are manual and automatic security formal methods respectively.

1.1 Main Contribution

In this paper, we:

- prove that the Lee *et al.* group ownership transfer protocol is vulnerable against secret disclosure attack;
- show vulnerability of the Lee *et al.* group ownership transfer protocol against reader impersonation attack;
- present traceability attack against Lee *et al.* protocol;
- explain Lee *et al.* protocol does not have forward secrecy. It worth noting that the success probability of any presented attack in this paper is 1 and the complexity is just a run of the group ownership transfer protocol;
- present desynchronization attack against Zhu *et al.* group ownership transfer protocol;

- propose a new secure group ownership transfer protocol;
- prove security correctness of proposed protocol informally and also using Barrows-Abadi-Needham logic and Scyther tool as a manual and automatic formal methods respectively.

1.2 Paper Organization

The rest of the paper is structured as below: Section 2, shows recent works which are done in the field of group ownership transfer protocol design. We review all required preliminaries and also the Lee *et al.* and Zhu *et al.* group ownership transfer protocols in Section 3. Section 4 is dedicated to present several security attacks against Lee *et al.* protocol, including secret disclosure attack, reader impersonation attack, traceability attack and forward secrecy contradiction attack and desynchronization attack against Zhu *et al.* protocol. Our proposed protocol is presented in Section 5. Section 6 shows the correctness of our proposed protocol both informally and formally through Barrows-Abadi-Needham logic and Scyther tool. The comparison between the proposed protocol with other related protocols from security and computational complexity aspects of view is explained in Section 7. Finally, the paper is concluded in Section 8.

2 Related Work

As stated above, ownership transfer protocols are referred to protocols, whereby they transfer ownership of a tag-labeled object from one owner to another one. There are many protocols in this area such as [2,3,7,9,8]. If during the process of ownership transfer, the ownership of several tags is transferred to the new owner at the same time, they are referred to as group ownership transfer protocols. In this regard, efforts have been made in the literature [10,12,19,21,22]. However, the security attacks against them [1,4,16,15,14] showed that none of them has ever been able to meet all security needs. These security attacks, in addition to helping to identify the weaknesses of these protocols, also helped designers to avoid the known attacks and weaknesses in their designs of new protocols. In detail, Yang in [21] proposed a novel group ownership transfer protocol and they assumed the old owner cannot eavesdrop the new owner's transferred massages, which in fact is an unrealistic assumption. They also designed their protocol based on tree structure which forces computational overhead problem and also windowing problem [17]. In [5], it is presented that the Yang's protocol is vulnerable against tag impersonation attack, and back-end server compromising attack. Another lightweight group ownership transfer protocol was proposed by Sundaresan *et al.* in [20] which uses 128-bit output length PRNGs and exclusive or operations. Munilla *et al.* in [15] proved that Sundaresan *et al.*'s protocol is vulnerable against forward secrecy contradiction attack, tag's location privacy contradiction attack, replay attack and desynchronization attack. Recently [14] based on homomorphic encryption function and quadratic residue problem designed a new group ownership transfer protocol for RFID tags and claimed their protocol has suitable security to employment in any application. The use of quadratic residue problem is not new in the design of ownership transfer protocols and there are such protocols like [8], but the use of the homomorphic encryption function is a new idea in the field of designing group ownership transfer protocols that perform operations without the need for decryption the ciphertext. However, the structure of protocol messages is not well designed, resulting in serious vulnerabilities to security attacks. In this paper, while describing these vulnerabilities, we try to resolve these vulnerabilities to achieve a secure group ownership transfer protocol that uses quadratic residue and homomorphic encryption function in its structure.

3 Preliminaries

In this section, we review notations used in the paper and also a brief description of Lee *et al.* [14] and Zhu *et al.* [23] protocols.

3.1 Notations

The notations which are used in this paper are summarized in Table 1. The adversary $\mathcal{A}$ used throughout the paper is passive which can eavesdrop all transferred messages through an insecure channel and also can do offline computations. Lee *et al.* in their protocol used homomorphic encryption which leads the computations on encrypted data to be efficient. Therefore, here we explain homomorphic Encryption.

Definition 1 : Given m_1 and m_2 are plaintext and E is a Homomorphic Encryption function, if we have $E_K(m_1) * E_K(m_2) = E_K(m_1 + m_2)$ where $*$ is addition or multiplication operation then E is homomorphic encryption. In the first case, E is called an additive homomorphic encryption and in the latter is called a multiplicative homomorphic encryption and if have both additive and multiplicative properties is called fully homomorphic encryption.

Lee *et al.* in their protocol used the quadratic residue to protect the messages that transmitted in their protocol. An integer R is called a quadratic residue modulo n if it is congruent to a perfect square modulo n, i.e., if the congruence $x^2 \equiv R \ modn$ has a solution. If n is the product of two prime numbers, using the Chinese Remainder Theorem, the congruence $x^2 \equiv R \ modn$ has no solution or exactly four incongruent solutions. However, one can be able to compute these solutions, which have factorization of n. Due to the difficulty of factoring, it is computationally infeasible to find numbers satisfying $x^2 \equiv R \ modn$ without knowing p and q in which $n = pq$, [18]. Assume that x is replaced with x^2, then if a solution exists for $(x^2)^2 \equiv R \ modn$, it is clear that the solution is required to be a perfect square. That, of the four possible solutions, only one of those would be a quadratic residue modulo n satisfying $x^2 \equiv R \ modn$ [8].

Table 1. Notations used in this paper

Notations	Description
T_i	The i^{th} RFID tag
R_j	The j^{th} RFID reader(owner)
R_{j+1}	The $j+1^{th}$ RFID reader(owner)
$\mathcal{A}$	The adversary
TID_i	The identifier of the i^{th} RFID tag
$r_{i,j}$	The secret value of the i^{th} RFID tag, which is shared with the reader R_j
$r_{i,j+1}$	The secret value of the i^{th} RFID tag, which is shared with the reader R_{j+1}
$KTID_{i,j}$	The secret value of the i^{th} RFID tag, which is shared with the reader R_j
$n_T^i, n_{T_2}^i, n_{T_3}^i, n_r, n_r'$	The random numbers
$PRNG$	The pseudo random number generator
p, q	The odd prime numbers
n	The product of two odd prime numbers
$\Vert$	The concatenation operation
$A \leftarrow B$	Assigning A value to B
$\oplus$	The bit wise exclusive-or operation
h, H	The one-way hash functions
$M(.)$	A simple homomorphic encryption such as $M(x) = g^x \ modq$
$A \sim B$	$A \ mod \ B$
s_i	The secret key between tag and back-end server in Zhu *et al.* protocol
CRC	The cyclic redundancy check function
ID_i, ID_i'	The current and old identifier of i^{th} tag in Zhu *et al.* protocol
k_i, k_i'	The current and old secret key between tag and reader in Zhu *et al.* protocol
ID_{group}	The identifier of each group of tags in Zhu *et al.* protocol
s_i	The shared secret between tag and back-end server in Zhu *et al.* protocol
k_{Mi}	The master key which is used in ownership recovery phase of Zhu *et al.* protocol
$n_{1r}^i, n_{2r}^i, n_s^i, n_T$	The random numbers generated by reader, back-end server and tag respectively in Zhu *et al.* protocol

3.2 Review of the Lee *et al.* protocol

In this section, we briefly introduce Lee *et al.* [14] protocol using notations represented in Table 1. As can be easily seen in Figures 1 and 2 [14] protocol includes two phases. An initialization phase and a group ownership transfer phase. In the initialization phase tags and readers registered with the cloud server. In their protocol, the communication channels between reader and tag are insecure, while the communication channels between reader and cloud server are secure.

Initialization phase In the tag initialization phase, tag T_i transmits Registration Request and its identity TID_i to cloud server via a secure channel. Cloud server computes $h(TID_i)$ and generates $K_{TID_{i,j}}$ randomly as an ownership key. Also, generates $r_{i,j}$ and $r_{i,j+1}$ as shared keys between current and new owner of the tag T_i. Then it generates two random numbers n_j and s_i that n_j is the product of two large primes p_j and q_j. Finally, cloud server transmits $\{h(TID_i), K_{TID_{i,j}}, r_{i,j}, r_{i,j+1}, n_j, s_i\}$ to T_i.

In the reader registration phase, the reader R_j sends Registration Request and its identity RID_j to cloud server via a secure channel. Cloud server upon receiving request and identity, computes hash values of tags that R_j is its owner $\{h(TID_i)\}$ and corresponding ownership keys $\{K_{TID_{i,j}}\}$, shared keys $\{r_{i,j}\}$, and set of pairs of large primes $\{p_j, q_j\}$ that $n_j = p_j q_j$.Then Cloud server transmits $\{\{h(TID_i)\}, \{K_{TID_{i,j}}\}, \{r_{i,j}\}, \{p_j, q_j\}\}$ to R_j. Cloud server stores these secret parameters in the form of an encrypted hash table.

Ownership transfer protocol phase The ownership transfer phase of [14] protocol includes 3 parts. Part 1 refers to the mutual authentication between T_i and R_j and part 2 refers to the mutual authentication between T_i and R_{j+1}. Part 3 is intended to transfer group ownership from R_j to R_{j+1}. In this paper, we are concerned with Part 1 and 2, since our analysis will focus on the vulnerabilities of these two parts. The details of these parts are described in the sequel.

When an ownership transfer protocol is started, Cloud Server sends Group OT Request (GOTR) to Current Mobile Reader R_j through a secure channel then R_j broadcasts OT Request (OTR) to each tag T_i in the group through an insecure channel. Details of part 1 of the proposed protocol is explained as follows:

- Tag $T_i \longrightarrow$ Current Mobile Reader R_j
  Tag T_i upon receiving GOTR, generates a random number n_T^i and computes following values:
  $R_T^i = h(TID_i) \oplus r_{i,j} \oplus n_T^i$,
  $R_T'^i = (R_T^i)^4 mod n_j$,
  $A_1^i = r_{i,j} \oplus n_T^i$.
  Then tag T_i sends $R_T'^i$ and A_1^i to the current mobile reader R_j.
- Current Mobile Reader $R_j \longrightarrow$ Tag T_i
  Reader R_j upon receiving $R_T'^i$ and A_1^i, computes $n_T^i = r_{i,j} \oplus A_1^i$. Then given p_j and q_j, it obtains R_T^i from $R_T'^i$ using Chinese Remainder Theorem(CRT).

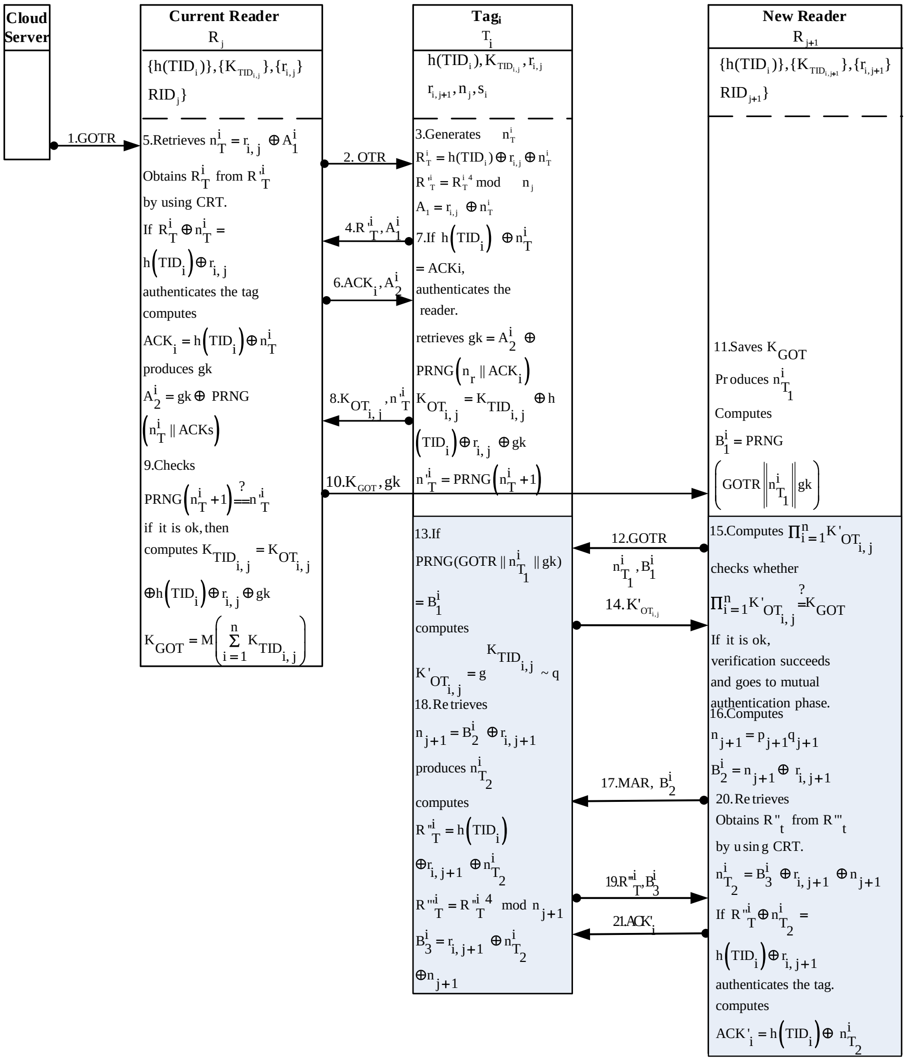

Cloud Server
Current Reader R_j
Tag$_i$ T_i
New Reader R_{j+1}
$\{h(TID_i)\},\{K_{TID_{i,j}}\},\{r_{i,j}\}$ $RID_j\}$
$h(TID_i), K_{TID_{i,j}}, r_{i,j}$ $r_{i,j+1}, n_j, s_i$
$\{h(TID_i)\},\{K_{TID_{i,j+1}}\},\{r_{i,j+1}\}$ $RID_{j+1}\}$
1.GOTR
2. OTR
$4.R'^i_T, A^i_1$
$6.ACK_i, A^i_2$
$8.K_{OT_{i,j}}, n'^i_T$
$10.K_{GOT}, gk$
12.GOTR
$n^i_{T_1}, B^i_1$
$14. K'_{OT_{i,j}}$
$17.MAR, B^i_2$
$19.R'''^i_T, B^i_3$
$21.ACK'_i$
5.Retrieves $n^i_T = r_{i,j} \oplus A^i_1$
Obtains R^i_T from R'^i_T by using CRT.
If $R^i_T \oplus n^i_T = h(TID_i) \oplus r_{i,j}$ authenticates the tag
computes $ACK_i = h(TID_i) \oplus n^i_T$
produces gk
$A^i_2 = gk \oplus$ PRNG $(n^i_T \| ACKs)$
9.Checks $PRNG(n^i_T + 1) \overset{?}{=} n'^i_T$
if it is ok, then computes $K_{TID_{i,j}} = K_{OT_{i,j}} \oplus h(TID_i) \oplus r_{i,j} \oplus gk$
$K_{GOT} = M\left(\sum_{i=1}^{n} K_{TID_{i,j}}\right)$
3.Generates n^i_T
$R^i_T = h(TID_i) \oplus r_{i,j} \oplus n^i_T$
$R'^i_T = R^{i\,4}_T \bmod n_j$
$A_1 = r_{i,j} \oplus n^i_T$
7.If $h(TID_i) \oplus n^i_T$ = ACKi, authenticates the reader.
retrieves $gk = A^i_2 \oplus PRNG(n_r \| ACK_i)$
$K_{OT_{i,j}} = K_{TID_{i,j}} \oplus h(TID_i) \oplus r_{i,j} \oplus gk$
$n'^i_T = PRNG(n^i_T + 1)$
11.Saves K_{GOT}
Produces $n^i_{T_1}$
Computes $B^i_1 = PRNG(GOTR \| n^i_{T_1} \| gk)$
13.If $PRNG(GOTR \| n^i_{T_1} \| gk) = B^i_1$
computes $K'_{OT_{i,j}} = g^{K_{TID_{i,j}}} \sim q$
18.Retrieves $n_{j+1} = B^i_2 \oplus r_{i,j+1}$
produces $n^i_{T_2}$
computes $R''^i_T = h(TID_i) \oplus r_{i,j+1} \oplus n^i_{T_2}$
$R'''^i_T = R''^{i\,4}_T \bmod n_{j+1}$
$B^i_3 = r_{i,j+1} \oplus n^i_{T_2} \oplus n_{j+1}$
15.Computes $\Pi^n_{i=1} K'_{OT_{i,j}}$
checks whether $\Pi^n_{i=1} K'_{OT_{i,j}} \overset{?}{=} K_{GOT}$
If it is ok, verification succeeds and goes to mutual authentication phase.
16.Computes $n_{j+1} = p_{j+1} q_{j+1}$
$B^i_2 = n_{j+1} \oplus r_{i,j+1}$
20.Retrieves
Obtains R''_t from R'''_t by using CRT.
$n^i_{T_2} = B^i_3 \oplus r_{i,j+1} \oplus n_{j+1}$
If $R''^i_T \oplus n^i_{T_2} = h(TID_i) \oplus r_{i,j+1}$ authenticates the tag.
computes $ACK'_i = h(TID_i) \oplus n^i_{T_2}$

Fig. 1. The part 1 and part 2 of Lee *et al.* group ownership transfer protocol. From step 12 onward is the part 2 of Lee *et al.* protocol [14].

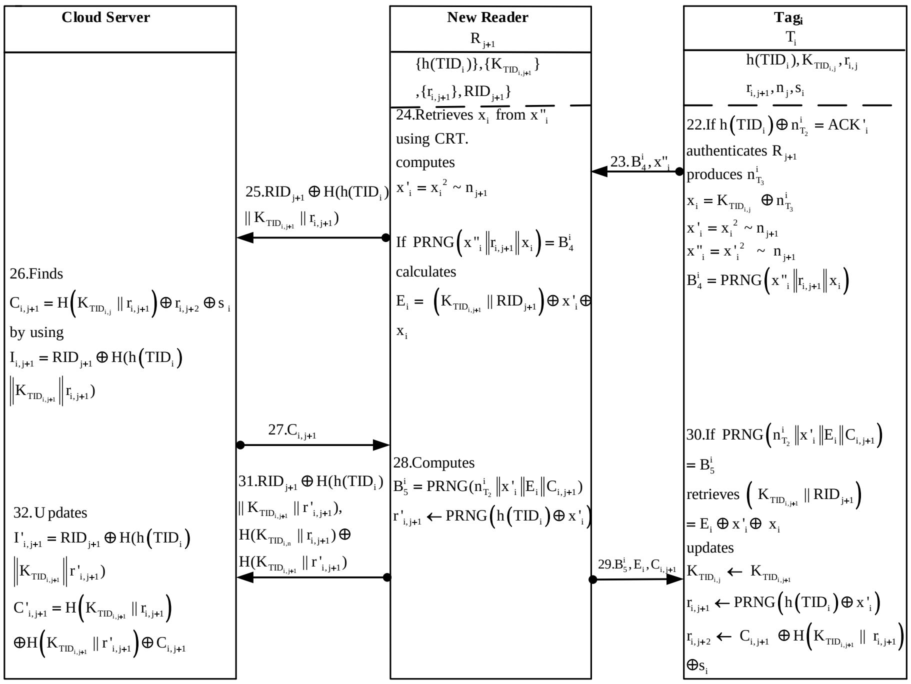

Cloud Server
New Reader
R_{j+1}
Tag
T_i
$\{h(TID_i)\}, \{K_{TID_{i,j+1}}\}$, $\{r_{i,j+1}\}, RID_{j+1}\}$
$h(TID_i), K_{TID_{i,j}}, r_{i,j}$, $r_{i,j+1}, n_j, s_i$
22.If $h(TID_i) \oplus n^i_{T_2} = ACK'_i$ authenticates R_{j+1} produces $n^i_{T_3}$
$x_i = K_{TID_{i,j}} \oplus n^i_{T_3}$
$x'_i = x_i^2 \sim n_{j+1}$
$x''_i = x'^2_i \sim n_{j+1}$
$B^i_4 = PRNG(x''_i \| r_{i,j+1} \| x_i)$
23.B^i_4, x''_i
24.Retrieves x_i from x''_i using CRT.
computes $x'_i = x_i^2 \sim n_{j+1}$
If $PRNG(x''_i \| r_{i,j+1} \| x_i) = B^i_4$ calculates
$E_i = (K_{TID_{i,j+1}} \| RID_{j+1}) \oplus x'_i \oplus x_i$
25.$RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,j+1}} \| r_{i,j+1})$
26.Finds $C_{i,j+1} = H(K_{TID_{i,j}} \| r_{i,j+1}) \oplus r_{i,j+2} \oplus s_i$ by using $I_{i,j+1} = RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,j+1}} \| r_{i,j+1})$
27.$C_{i,j+1}$
28.Computes $B^i_5 = PRNG(n^i_{T_2} \| x'_i \| E_i \| C_{i,j+1})$
$r'_{i,j+1} \leftarrow PRNG(h(TID_i) \oplus x'_i)$
29.$B^i_5, E_i, C_{i,j+1}$
30.If $PRNG(n^i_{T_2} \| x'_i \| E_i \| C_{i,j+1}) = B^i_5$
retrieves $(K_{TID_{i,j+1}} \| RID_{j+1}) = E_i \oplus x'_i \oplus x_i$
updates
$K_{TID_{i,j}} \leftarrow K_{TID_{i,j+1}}$
$r_{i,j+1} \leftarrow PRNG(h(TID_i) \oplus x'_i)$
$r_{i,j+2} \leftarrow C_{i,j+1} \oplus H(K_{TID_{i,j+1}} \| r_{i,j+1}) \oplus s_i$
31.$RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,j+1}} \| r'_{i,j+1})$, $H(K_{TID_{i,n}} \| r_{i,j+1}) \oplus H(K_{TID_{i,j+1}} \| r'_{i,j+1})$
32.Updates
$I'_{i,j+1} = RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,j+1}} \| r'_{i,j+1})$
$C'_{i,j+1} = H(K_{TID_{i,j+1}} \| r_{i,j+1}) \oplus H(K_{TID_{i,j+1}} \| r'_{i,j+1}) \oplus C_{i,j+1}$

Fig. 2. The part 3 of Lee et al. group ownership transfer protocol [14]

If $R^i_T \oplus n^i_T = h(TID_i) \oplus r_{i,j}$, then it authenticates the tag $T_i$and R_j computes $ACK_i = h(TID_i) \oplus n^i_T$. Reader R_j generates randomly gk and computes $A^i_2 = gk \oplus PRNG(n^i_T \| ACK_i)$ and finally sends ACK_i and A^i_2 to the tag T_i.

- Tag $T_i \longrightarrow$ Current Mobile Reader R_j

  Tag T_i checks whether $h(TID_i) \oplus n^i_T \overset{?}{=} ACK_i$, if equality holds then it authenticates R_j.

  Tag T_i computes $gk = A^i_2 \oplus PRNG(n^i_T \| ACK_i)$,$K_{OT_{i,j}} = K_{TID_{i,j}} \oplus h(TID_i) \oplus r_{i,j} \oplus gk$, and $n'^i_T = PRNG(n^i_T + 1)$.Then it sends n'^i_T and $K_{OT_{i,j}}$ to Current Mobile Reader R_j.
- Current Mobile Reader $R_j \longrightarrow$ Cloud Server

  R_j once receipts the message, computes $PRNG(n^i_T + 1)$ and compares it with received n'^i_T. If they are equal, computes $K_{TID_{i,j}} = K_{OT_{i,j}} \oplus h(TID_i) \oplus r_{i,j} \oplus gk$ and $K_{GOT} = M(\sum_{i=1}^{n} K_{TID_{i,j}})$.

  Finally R_j transmits K_{GOT} and gk to the cloud server via a secure channel and cloud server sends aggregated ownership key K_{GOT} and gk to the new mobile reader R_{j+1} via a secure channel. New mobile reader R_{j+1} stores K_{GOT} and generates another random number $n^i_{T_1}$ then computes $B^i_1 = PRNG(GOTR \| n^i_{T_1} \| gk)$.

After above steps, part 2 of the protocol starts as below:

- New Mobile Reader $R_{j+1} \longrightarrow$ Tag T_i

  New mobile reader R_{j+1} transmits Group OT Request(GOTR), $n^i_{T_1}$ and B^i_1 to the tag T_i.
- Tag $T_i \longrightarrow$ New Mobile Reader R_{j+1}

  Tag T_i checks whether $PRNG(GOTR \| n^i_{T_1} \| gk) \overset{?}{=} B^i_1$, if the equality holds then it computes $K'_{OT_{i,j}} = g^{K_{TID_{i,j}}}\ modq$ and transmits $K'_{OT_{i,j}}$ to R_{j+1}.
- New Mobile Reader $R_{j+1} \longrightarrow$ Tag T_i

  New mobile reader first checks whether $\prod_{i=1}^{n} K'_{OT_{i,j}} \overset{?}{=} K_{GOT}$. If they are equal, then it computes $B^i_2 = n_{j+1} \oplus r_{i,j+1}$ and broadcasts Mutual Authentication Request (MAR) and B^i_2 to each tag in the tag group.
- Tag $T_i \longrightarrow$ New Mobile Reader R_{j+1}

  Tag, first, computes $n_{j+1} = B^i_2 \oplus r_{i,j+1}$ then generates $n^i_{T_2}$ and also computes $R''^i_T = h(TID_i) \oplus r_{i,j+1} \oplus n^i_{T_2}$, $R'''^i_T = (R''^i_T)^4\ modn_{j+1}$and $B^i_3 = r_{i,j+1} \oplus n^i_{T_2} \oplus n_{j+1}$ and transmits R'''^i_T and B^i_3 to R_{j+1}.
- New Mobile Reader $R_{j+1} \longrightarrow$ Tag T_i

  Reader R_{j+1} when receiving R'''^i_T and B^i_3, computes $n^i_{T_2} = B^i_3 \oplus r_{i,j+1} \oplus n_{j+1}$ and solves quadratic congruence $R'''^i_T = (R''^i_T)^4\ modn_{j+1}$ and derives R''^i_T. Now if $R''^i_T \oplus n^i_{T_2} = h(TID_i) \oplus r_{i,j+1}$ then authenticates the tag and R_{j+1} computes $ACK'_i = h(TID_i) \oplus n^i_{T_2}$ and sends ACK'_i to the tag T_i.

  The part 3 of Lee *et al.* protocol proceeds as follows:

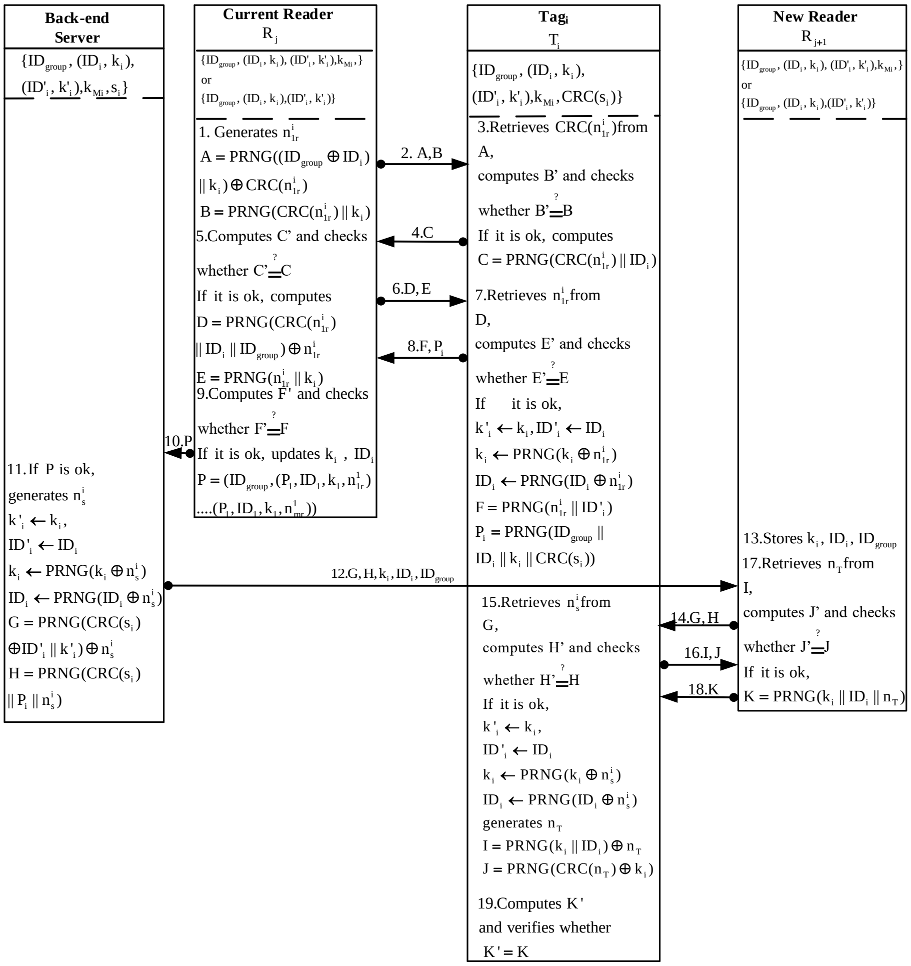
- Tag $T_i \longrightarrow$ New Mobile Reader R_{j+1}
  Once receives the message, the tag T_i checks whether $h(TID_i) \oplus n^i_{T_2} \stackrel{?}{=} ACK'_i$ is or not. If it is not, it finishes the protocol otherwise authenticates R_{j+1} and in order to begin the process of ownership transfer, generates a random number $n^i_{T_3}$ and computes $x_i = K_{TID_{i,j}} \oplus n^i_{T_3}$, $x'_i = x^2_i\ modn_{j+1}$,$x''_i = x^4_i\ modn_{j+1}$ and $B^i_4 = PRNG(x''_i \| r_{i,j+1} \| x_i)$ and sends B^i_4 and x''_i to the R_{j+1}.
- New Mobile Reader $R_{j+1} \longrightarrow$ Cloud Server
  When R_{j+1} receives the message, retrieves x_i from x''_i using Chinese Reminder Theorem (CRT)and then checks whether $PRNG(x''_i \| r_{i,j+1} \| x_i) \stackrel{?}{=} B^i_4$ is or not. If it is not, R_{j+1} terminates the protocol otherwise, generates a new ownership key $K_{TID_{i,j+1}}$, computes $E = (K_{TID_{i,j+1}} \| RID_{j+1}) \oplus x'_i \oplus x_i$ and sends $RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,j+1}} \| r_{i,j+1})$ to the cloud server.
- Cloud Server $T_i \longrightarrow$ New Mobile Reader R_{j+1}
  Once the cloud server receives the message, by using index $I_{i,j+1} = RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,j+1}} \| r_{i,j+1})$ in its encrypted hash table finds the content $C_{i,j+1} = H(K_{TID_{i,n} \| r_{i,j+1}}) \oplus r_{i,j+2} \oplus s_i$ and sends $C_{i,j+1}$ to the reader R_{j+1}.
- New Mobile Reader $R_{j+1} \longrightarrow$ Tag T_i
  Upon receipt of the message, R_{j+1} computes $B^i_5 = PRNG(n^i_{T_2} \| x'_i \| E \| C_{i,j+1})$ and updates its secret values as $r'_{i,j+1} \leftarrow PRNG(h(TID_i) \oplus x'_i)$ and sends B^i_5, E and $C_{i,j+1}$ to the tag T_i and $RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,j+1}} \| r'_{i,j+1})$ and $H(K_{TID_{i,n} \| r_{i,j+1}}) \oplus H(K_{TID_{i,j+1} \| r'_{i,j+1}})$ to the cloud server.
- Tag T_i
  when T_i receives the message, checks whether $PRNG(n^i_{T_2} \| x'_i \| E \| C_{i,j+1}) \stackrel{?}{=} B^i_5$, if it is not, T_i stops the protocol, otherwise performs ownership update process by computing $(K_{TID_{i,j+1}} \| RID_{j+1}) = E \oplus x'_i \oplus x_i$ and then updating secret values as $K_{TID_{i,j}} \leftarrow K_{TID_{i,j+1}}$, $r_{i,j+1} \leftarrow PRNG(h(TID_i) \oplus x'_i)$ and $r_{i,j+2} \leftarrow C_{i,j+1} \oplus H(K_{TID_{i,j+1}} \| r_{i,j+1}) \oplus s_i$.
- Cloud server
  Once cloud server receives the message, updates its hash table index and content as $I'_{i,j+1} = RID_{j+1} \oplus H(h(TID_i) \| K_{TID_{i,n}} \| r'_{i,j+1})$ and $C'_{i,j+1} = H(K_{TID_{i,n}} \| r_{i,j+1}) \oplus H(K_{TID_{i,n}} \| r'_{i,j+1}) \oplus C_{i,j+1}$.

3.3 Review of the Zhu *et al.* protocol

In this section, we briefly introduce Zhu *et al.* protocol [23] which includes four phases: initialization phase, mutual authentication phase, ownership transfer phase and ownership recovery phase.

Initialization phase In initialization phase, secret parameters of each entity are assigned via a secure channel. Back-end sever generates ID_{group} for each group and $CRC(s_i)$ for every tag in which s_i is secret key between tag and back-end server. Back-end server holds tuple $\{ID_{group}, s_i, k_{M_i}, (k_i, ID_i), (k'_i, ID'_i)\}$, in which (k_i, ID_i) are the current secret parameters and (k'_i, ID'_i) are the previous secret parameters. Also, each tag saves the set $\{ID_{group}, CRC(s_i), k_{M_i}, (k_i, ID_i), (k'_i, ID'_i)\}$. In this protocol, the owner that can execute ownership recovery phase saves the set $\{ID_{group}, k_{M_i}, (k_i, ID_i), (k'_i, ID'_i)\}$ and another owner holds only the set $\{ID_{group}, (k_i, ID_i), (k'_i, ID'_i)\}$.

Mutual authentication phase Mutual authentication phase of the Zhu *et al.* protocol as depicted in Figure 3 runs as below:



Back-end Server
$\{ID_{group}, (ID_i, k_i), (ID'_i, k'_i), k_{Mi}, s_i\}$
Current Reader
R_j
$\{ID_{group}, (ID_i, k_i), (ID'_i, k'_i), k_{Mi}, \}$
or
$\{ID_{group}, (ID_i, k_i), (ID'_i, k'_i)\}$
Tag$_i$
T_i
$\{ID_{group}, (ID_i, k_i), (ID'_i, k'_i), k_{Mi}, CRC(s_i)\}$
New Reader
R_{j+1}
$\{ID_{group}, (ID_i, k_i), (ID'_i, k'_i), k_{Mi}, \}$
or
$\{ID_{group}, (ID_i, k_i), (ID'_i, k'_i)\}$
1. Generates n^i_{1r}
$A = PRNG((ID_{group} \oplus ID_i) \| k_i) \oplus CRC(n^i_{1r})$
$B = PRNG(CRC(n^i_{1r}) \| k_i)$
2. A,B
3.Retrieves $CRC(n^i_{1r})$from A,
computes B' and checks whether $B' \overset{?}{=} B$
If it is ok, computes
$C = PRNG(CRC(n^i_{1r}) \| ID_i)$
4.C
5.Computes C' and checks whether $C' \overset{?}{=} C$
If it is ok, computes
$D = PRNG(CRC(n^i_{1r}) \| ID_i \| ID_{group}) \oplus n^i_{1r}$
$E = PRNG(n^i_{1r} \| k_i)$
6.D, E
7.Retrieves $n^i_{1r}$from D,
computes E' and checks whether $E' \overset{?}{=} E$
If it is ok,
$k'_i \leftarrow k_i, ID'_i \leftarrow ID_i$
$k_i \leftarrow PRNG(k_i \oplus n^i_{1r})$
$ID_i \leftarrow PRNG(ID_i \oplus n^i_{1r})$
$F = PRNG(n^i_{1r} \| ID'_i)$
$P_i = PRNG(ID_{group} \| ID_i \| k_i \| CRC(s_i))$
8.F, P_i
9.Computes F' and checks whether $F' \overset{?}{=} F$
If it is ok, updates k_i, ID_i
$P = (ID_{group}, (P_1, ID_1, k_1, n^1_{1r}) ....(P_1, ID_1, k_1, n^1_{mr}))$
10.P
11.If P is ok, generates n^i_s
$k'_i \leftarrow k_i$,
$ID'_i \leftarrow ID_i$
$k_i \leftarrow PRNG(k_i \oplus n^i_s)$
$ID_i \leftarrow PRNG(ID_i \oplus n^i_s)$
$G = PRNG(CRC(s_i) \oplus ID'_i \| k'_i) \oplus n^i_s$
$H = PRNG(CRC(s_i) \| P_i \| n^i_s)$
12.G, H, k_i, ID_i, ID_{group}
13.Stores k_i, ID_i, ID_{group}
14.G, H
15.Retrieves $n^i_s$from G,
computes H' and checks whether $H' \overset{?}{=} H$
If it is ok,
$k'_i \leftarrow k_i$,
$ID'_i \leftarrow ID_i$
$k_i \leftarrow PRNG(k_i \oplus n^i_s)$
$ID_i \leftarrow PRNG(ID_i \oplus n^i_s)$
generates n_T
$I = PRNG(k_i \| ID_i) \oplus n_T$
$J = PRNG(CRC(n_T) \oplus k_i)$
16.I, J
17.Retrieves $n_T$from I,
computes J' and checks whether $J' \overset{?}{=} J$
If it is ok,
$K = PRNG(k_i \| ID_i \| n_T)$
18.K
19.Computes K' and verifies whether $K' = K$

Fig. 3. Mutual authentication and ownership transfer phases of Zhu *et al.* protocol [23]

- Current owner $R_j \longrightarrow$ Tag T_i
  Current owner generates a random number n^j_{1r} and computes the values: $A = PRNG((ID_{group} \oplus ID_i)\|k_i) \oplus CRC(n^i_{1r})$, $B = PRNG(CRC(n^i_{1r})\|k_i)$. Then R_j sends A and B to T_i.
- Tag $T_i \longrightarrow$ Current owner R_j
  When tag T_i receives A and B then retrieves $CRC(n^i_{1r})$ from message $A \oplus PRNG((ID_{group} \oplus ID_i)\|k_i)$ and using k_i computes the value $B' = PRNG(CRC(n^i_{1r})\|k_i)$. Tag T_i checks whether $B' \overset{?}{=} B$ is or not, if equality holds computes $C = PRNG(CRC(n^i_{1r})\|ID_i)$ and sends it to the current owner.
- Current owner $R_j \longrightarrow$ Tag T_i
  Current owner authenticates tag by computing $C' = PRNG(CRC(n^i_{1r})\|ID_i)$ and comparing it with C that has received from tag. After verification, it computes $D = PRNG(CRC(n^i_{1r})\|ID_i\|ID_{group}) \oplus n^i_{1r}$, $E = PRNG(n^i_{1r}\|k_i)$, and sends D and E to the tag T_i.
- Tag $T_i \longrightarrow$ Current owner R_j
  Tag retrieves n^i_{1r} from D and computes E' and compares it with E to authenticate the current owner. After successful current owner authentication, the tag updates (k_i, ID_i) with $(PRNG(k_i \oplus n^i_{1r}), PRNG(ID_i \oplus n^i_{1r}))$ and replaces (k'_i, ID'_i) with (k_i, ID_i). Then it computes $F = PRNG(n^i_{1r}\|ID'_i)$ and $P_i = PRNG(ID_{group}\|ID_i\|k_i\|CRC(s_i))$ and sends them to the current owner.
- Current owner $R_j \longrightarrow$ Tag T_i
  Current owner calculates F' and verifies whether $F' \overset{?}{=} F$ is or not, if equality holds, R_j updates (k_i, ID_i) and calculates
  $P = ((ID_{group}, (P_1, ID_1, k_1, n^1_{1r}) \cdots (P_m, ID_m, k_m, n^1_{mr})))$.
  Then transfers it to the back-end server.

Ownership transfer phase Back-end server after receiving ownership transfer request from new owner, verifies the request and transfers it to the old owner. The old owner after receiving request executes the mutual authentication. Then back-end server verifies P and s_i (secret key between server and tag), after that P was verified, the server updates (k_i, ID_i) using n^i_{1r} that generated by the old owner and compares updated parameters with the received ones. Then as depicted in Figure 3, the server produces a random number n^i_s to update (k_i, ID_i) that is $k'_i \leftarrow k_i, ID'_i \leftarrow ID_i$ and $k_i \leftarrow PRNG(k_i \oplus n^i_s)$, $ID_i \leftarrow PRNG(ID_i \oplus n^i_s)$ and computes $G = PRNG(CRC(s_i) \oplus ID'_i\|k'_i) \oplus n^i_s$, $H = PRNG(CRC(s_i)\|P_i\|n^i_s)$. Then the server transfers G, H, k_i, ID_i and ID_{group} to the new owner.

- New owner $\longrightarrow$ Tag T_i
  New owner keeps k_i, ID_i and ID_{group} and transfers G and H to the tag T_i.
- Tag $T_i \longrightarrow$ New owner
  T_i upon receiving G and H, retrieves n^i_s from G and computes $PRNG(CRC(s_i)\|P_i\|n^i_s)$. If $H = PRNG(CRC(s_i)\|P_i\|n^i_s)$ then T_i updates k_i and ID_i. Tag T_i generates a random number n_T and computes $I = PRNG(k_i\|ID_i) \oplus$

n_T and $J = PRNG(CRC(n_T) \oplus k_i)$. Then, the tag transfers I and J to the new owner.

- New owner $\longrightarrow$ Tag T_i
  The new owner, first, retrieves n_T from I and calculates $PRNG(CRC(n_T) \oplus k_i)$ if it is equal to J then computes $K = PRNG(k_i \| ID_i \| n_T)$ and sends K to the tag.
- Tag T_i $\longrightarrow$ New owner
  Tag T_i computes $PRNG(k_i \| ID_i \| n_T)$ and checks equality with K. If equality in this step holds for all tags in the group then tags accept the new owner as their owner.

Ownership recovery phase

- Old owner $\longrightarrow$ Tag T_i
  Old owner generates a random number n^i_{2r} and computes $A = PRNG((ID_{group} \oplus ID_i) \| k_i \| k_{M_i}) \oplus CRC(n^i_{2r})$ and $B = PRNG(CRC(n^i_{2r}) \| k_i \| k_{M_i})$. Then it sends A, B and recovery request to the tag.
- Tag T_i $\longrightarrow$ Old owner
  Tag T_i finds previous k_i, ID_i and k_{M_i} to retrieve $CRC(n^i_{2r})$ from A. Then it computes $PRNG(CRC(n^i_{2r}) \| k_i \| k_{M_i})$ and compares it with received B. If equality holds, the tag authenticates the old owner and calculates $C = PRNG(CRC(n^i_{2r}) \| ID_i \| k_{M_i})$. The tag transfers C to the old owner.
- Old owner $\longrightarrow$ Tag T_i
  The old owner verifies C and computes $D = PRNG(CRC(n^i_{2r}) \| ID_i \| ID_{group})$ and $E = PRNG(n^i_{2r} \| k_i \| k_{M_i})$ and sends them to the tag.
- Tag T_i $\longrightarrow$ Old owner
  Tag T_i retrieves n^i_{2r} from D and verifies received E if verification is true updates k_i and ID_i as $k_i \leftarrow PRNG(k_i \oplus n^i_{2r})$ and $ID_i \leftarrow PRNG(ID_i \oplus n^i_{2r})$. Then it calculates $F = PRNG(k'_i \| ID'_i \| n^i_{2r})$ and sends it to the old owner.
- Old owner $\longrightarrow$ Tag T_i
  The old owner after receiving F verifies it and if verification is true then updates k_i and ID_i.

4 Security Vulnerabilities of Previous Protocols

Here, we show how some of security claims of Lee *et al.* for their proposed protocol can be violated.

Its weaknesses include secret disclosure or tag data privacy and ownership privacy contradiction attack, forward secrecy contradiction attack and reader impersonation attack. Details of these attacks are explained as below:

4.1 Secret Disclosure Attack

The secret disclosure attack is an attack in which the adversary attempts to use transferred messages in the protocol to obtain secret protocol values. As can be seen in Algorithm 1, for this attack which done in order to contradict data privacy and ownership privacy, it is enough the adversary $\mathcal{A}$ does as below in two phases:

- **Learning Phase:** Eavesdrops part 1 and 2 of the protocol.
- **Secret Disclosure Phase:**
  - Retrieves $n_{T_2}^i$ as $B_2^i \oplus B_3^i$.
  - Obtains $h(TID_i)$ as $h(TID_i) = ACK_i' \oplus n_{T_2}^i$.
  - Retrieves n_T^i as $ACK_i \oplus h(TID_i)$.
  - Then obtains $r_{i,j} =$ as $A_1^i \oplus n_T^i$.
  - Finally, with eavesdropping $K_{OT_{i,j}}$ can obtain $K_{TID_{i,j}}$ as $K_{OT_{i,j}} \oplus h(TID_i) \oplus r_{i,j} \oplus gk$.

So, the adversary can obtain secret parameters of the tag and the old owner i.e. R_j, hence Lee *et al.* protocol cannot preserve data privacy and is vulnerable to secret disclosure attack. The success probability of this attack is 1 and it only needs one run of protocol. It worth noting that by obtaining $K_{TID_{i,j}}$, Lee *et al.* protocol cannot preserve ownership privacy.

Algorithm 1: The algorithm of proposed secret disclosure attack against the Lee *et al.* protocol

```
Data: $A_1^i, n_T^i, A_2^i, B_2^i, B_3^i, ACK_i, ACK_i', K_{OT_{i,j}}, gk$
Result: Obtains $n_{T_2}^i$, $h(TID_i)$ ,$r_{i,j}$, $K_{TID_{i,j}}$
1. $n_{T_2}^i = B_2^i \oplus B_3^i$
2. $h(TID_i) = ACK_i' \oplus n_{T_2}^i$
3. $n_T^i = ACK_i \oplus h(TID_i)$
4. $r_{i,j} = A_1^i \oplus n_T^i$
5. $K_{TID_{i,j}} = K_{OT_{i,j}} \oplus h(TID_i) \oplus r_{i,j} \oplus gk$
```

4.2 Forward Secrecy Contradiction

Forward secrecy is a security feature that points out that if the secret values of a protocol in the current session go down, the adversary should not be able to retrieve all of the secret values used in previous sessions. In order to provide this feature, most security protocols update their secret values used at the session after each run.

Since, during the time that the reader R_j owns the tag T_i, its secret values including TID_i, $h(TID_i)$, $r_{i,j}$ and $KTID_{i,j}$ are not updated, by obtaining these values, we can say that all the secret values of tag and reader that have in the past are also retrieved. Therefore, the above protocol does not meet the forward secrecy feature. For, forward secrecy contradiction, it is enough the adversary follows the steps described in Section 4.1.

4.3 Reader Impersonation Attack

An impersonation attack is an attack in which the adversary attempts to forge a legitimate protocol's party, gain information about the other party to the protocol or gain access to the service. For such example, reader impersonation is an attack in which the adversary places himself behind a legitimate reader and can force the tag to respond to its messages. Here, we show that [14] protocol is also, vulnerable to the reader impersonation attack. The scenario of the attack is as follows:

- **Learning Phase**: In this phase, the adversary eavesdrops a session of protocol and obtains $h(TID_i)$ by using a method described in Section 4.1 and since it has $h(TID_i)$ can obtain gk. Hence it can compute B_1^i.
- **Reader Impersonation Phase**: In this phase, the adversary:
  - intends it is a legal new mobile reader.
  - Then it sends B_1^i, $n_{T_1}^i$ and Group OT Request (GOTR) to the tag T_i instead of the new mobile reader.
  - Obtains $K_{TID_{i,j}}$, so it can check whether $\prod_{i=1}^{n} K'_{OT_{i,j}} \overset{?}{=} K_{GOT}$.
  - Generates B_2^i randomly and broadcasts it to the group of tags.
  - Each tag computes $R_T'''^i$ and B_3^i and sends them to the adversary instead of the legal new mobile reader.
  - Then the adversary obtains the real $n_{T_2}^i$ as $B_2^i \oplus B_3^i$ and computes $ACK_i' = h(TID_i) \oplus n_{T_2}^i$ that can be verified by the tag and tag accepts the adversary as the legal mobile reader R_{j+1}.

It worth noting that the complexity of reader impersonation attack is only eavesdropping one run of protocol and its success probability is 1.

4.4 Traceability Attack

A traceability attack is an attack in which an adversary attempts to access fixed information associated with tag's or reader's identity using the transferred messages in the protocol and then uses this fixed information to trace it. Here we show that the Lee *et al.* ownership transfer protocol is also vulnerable against traceability attack. The traceability attack presented in this section is based on the traceability model provided by [11]. In this traceability model, there is a hypothesis that the adversary has the constant $h(TID_i)$ of two tags, namely $h(TID_0)$ and $h(TID_1)$ and is faced with a game that only contains one of these two tags. Now, if the adversary can correctly guess which tag is in the game, he/she is the winner of the game and it is said that the protocol is vulnerable to the traceability attack. The adversary succeeds in distinguishing between two tags if his chance of guessing is correct from the possibility of random probability (i.e., 0.5) has a significant deviation. In other words, assuming a constant $h(TID_i)$ and the adversary's advantage Adv_{adv} is calculated in the application of traceability attack on the protocol as follows:

$$Adv_{adv}(h(TID_0), (h(TID_1)) = |Pr_{CorrectGuess} - Pr_{RandomGuess}| > \epsilon,$$

where $Pr_{CorrectGuess}$, $Pr_{RandomGuess}$ are the probability of a correct guess and random guess, respectively and ϵ is a negligible function. Our proposed traceability attack as depicted in Algorithm 2 accomplishes as below:

- **Learning Phase:** In this phase, the adversary eavesdrops one run of the protocol.
- **Traceability Attack Phase:** In this phase, the adversary does as below:
  - Retrieves $n_{T_2}^i$ as $B_2^i \oplus B_3^i$.
  - Obtains $h(TID_i)$ as $h(TID_i) = ACK_i' \oplus n_{T_2}^i$.

Algorithm 2: The algorithm of proposed traceability attack against the Lee *et al.* protocol

```
Data: $h(TID_0), h(TID_1), B_2^i, B_3^i, ACK_i'$
Result: Obtains $h(TID_i)$
1. $n_{T_2}^i = B_2^i \oplus B_3^i$
2. $h(TID_i) = ACK_i' \oplus n_{T_2}^i$
3. Compares $h(TID_i)$ with $h(TID_0)$ and $h(TID_1)$
4. Decides that faces with which tag $T_0$ or $T_1$ with the success probability of "1"
```

On the other hand, assuming that the adversary knows the actual value of the $h(TID_0)$ and $h(TID_1)$ which are receptively related to tag T_0 and T_1, then with the probability of 1, by comparing the obtained $h(TID_i)$ with $h(TID_0)$ and $h(TID_1)$, it will know which tag is encountered.

4.5 Desynchronization attack on the Zhu *et al.* Protocol

In this section we show that group ownership protocol proposed by Zhu *et al.* [23] is vulnerable to desynchronization attack.

- Session j
  To simplicity in notations assume that the tag and the reader save (k_i^j, ID_i^j) and (k_i^{j-1}, ID_i^{j-1}) as the current and previous parameters in the j-th session in the mutual authentication phase.
- Session $j+1$
  Assume that a successful session is executed and the tag and the reader update their parameters as (k_i^{j+1}, ID_i^{j+1}) and (k_i^j, ID_i^j) as the current and previous parameters. In this session, the adversary eavesdrops the transferred messages of this session over the insecure channel, i.e. A^{j+1}, B^{j+1}, C^{j+1}, D^{j+1} and E^{j+1}, and saves them.
- Session $j+2$
  In this session, the adversary eavesdrops messages A^{j+2}, B^{j+2}, C^{j+2}, D^{j+2}

and E^{j+2}, saves them and blocks messages D^{j+2} and E^{j+2}. So the authentication is failed and the tag and reader do not update their parameters. If the adversary did not intercept the protocol, the tag and the reader update their shared values to (k_i^{j+2}, ID_i^{j+2}) and (k_i^{j+1}, ID_i^{j+1}). However, following the interception, their shared values remain as (k_i^{j+1}, ID_i^{j+1}) and (k_i^j, ID_i^j), but the tag knows that it has not been authenticated based on (k_i^{j+1}, ID_i^{j+1}).

- Session $j+3$
  Since previous authentication is failed, the tag/reader executes the session with (k_i^j, ID_i^j), and this time the adversary does not interfere with the protocol. So the protocol successfully runs. After this successful run of the protocol, the tag and the reader update their parameters with (k_i^{j+3}, ID_i^{j+3}) and (k_i^j, ID_i^j).
- Session $j+4$
  In the next step of the attack, the adversary impersonates the reader and sends A^{j+1} and B^{j+1} that eavesdropped from the previous sessions, i.e. session $j+1$. So the adversary can successfully impersonate the reader and cause the tag updates its parameters with (k_i^{j+1}, ID_i^{j+1}) and (k_i^j, ID_i^j) while the reader dose not update its parameters and still has (k_i^{j+3}, ID_i^{j+3}) and (k_i^j, ID_i^j).
- Session $j+5$
  In this session, once again the adversary impersonates the reader and sends A^{j+2} and B^{j+2} that eavesdropped from the previous sessions, i.e. session $j+2$. So the adversary can successfully impersonate the reader and cause the tag updates its parameters with (k_i^{j+2}, ID_i^{j+2}) and (k_i^{j+1}, ID_i^{j+1}) while the reader dose not update its parameters and still has (k_i^{j+3}, ID_i^{j+3}) and (k_i^j, ID_i^j). As it can be seen, none of the values stored in the tag and the reader are identical. Hence, the tag and the reader will be desynchronized after the above attack.

The success probability of above attack is almost "1" and its complexity is only five runs of the protocol.

5 Proposed Protocol

In this section, to overcome security vulnerabilities of [14] protocol, we propose a new secure one. As can be easily seen in Figures 4 our protocol same as its predecessor i.e. Lee *et al.* includes two phases. An initialization phase and a group ownership transfer phase. In the initialization phase tags and readers registered with the cloud server. In our proposed protocol, the communication channels between readers and tags are insecure, while the communication channels between readers and cloud server are secure.

5.1 Initialization phase

In the tag initialization phase, the tag T_i transmits Registration Request and its identity TID_i to cloud server via a secure channel. Cloud server computes

$h(TID_i)$ and generates $K_{TID_{i,j}}$ randomly as an ownership key, also, generates $r_{i,j}$ and $r_{i,j+1}$ as shared keys between current and new owner of the tag T_i. Then it generates two random number n_j and s_i that n_j is the product of two large primes p_j and q_j. Cloud server transmits $\{h(TID_i), K_{TID_{i,j}}, r_{i,j}, r_{i,j+1}, n_j\}$ to T_i.

In the reader registration phase, the reader R_j sends Registration Request and its identity RID_j to cloud server via a secure channel. Cloud server upon receiving request and identity, computes hash values of tags that R_j is their owner $\{h(TID_i)\}$ and corresponding ownership keys $\{K_{TID_{i,j}}\}$, shared keys $\{r_{i,j}\}$, and set of pairs of large primes $\{p_j, q_j\}$ that $n_j = p_j \times q_j$.Then cloud server transmits $\{\{h(TID_i)\}, \{K_{TID_{i,j}}\}, \{r_{i,j}\}, \{p_j, q_j\}\}$ to R_j. Cloud server stores these secret parameters in the form of an encrypted hash table.

5.2 Ownership transfer protocol phase

The ownership transfer phase of our proposed protocol includes 3 parts. Part 1 refers to the mutual authentication between T_i and R_j and part 2 refers to the mutual authentication between T_i and R_{j+1}. Part 3 is intended to transfer group ownership from R_j to R_{j+1}. The details of these parts are described in the sequel.

When an ownership transfer protocol is started, Cloud Server sends Group OT Request (GOTR) to Current Mobile Reader R_j through a secure channel then R_j broadcasts OT Request(OTR) to each tag T_i in the group through an insecure channel. Details of part 1 of the proposed protocol are explained as follows:

- Tag $T_i \longrightarrow$ Current Mobile Reader R_j
  T_i upon receiving OT Request, generates a random number n_T^i and computes $R_T^i = h(TID_i) \oplus r_{i,j} \oplus n_T^i$ and $R_T'^i = (R_T^i)^4 \mod n_j$. Then tag T_i sends $R_T'^i$ and n_T^i to the current mobile reader R_j.
- Current Mobile Reader $R_j \longrightarrow$ Tag T_i
  Reader R_j upon receiving $R_T'^i$ and n_T^i, given p_j and q_j, obtains R_T^i from $R_T'^i$ by using Chinese Reminder Theorem (CRT) and verifies its correctness based on the received n_T^i and its records of the tags. Next, if the previous step was successful, the reader authenticates the tag T_i. R_j then generates a random number n_r and, to response T_i, computes $R_r^i = PRNG(n_T^i \| (h(TID_i) \oplus n_r))$ and finally sends R_r^i and n_r to the tag T_i.
- Tag $T_i \longrightarrow$ Current Mobile Reader R_j
  Tag T_i verifies the correctness of R_r^i based on its local data and also the received n_r to authenticate the reader. Then, the tag T_i computes $K_{OT_{i,j}} = K_{TID_{i,j}} \oplus n_r \oplus n_T^i \oplus h(TID_i) \oplus r_{i,j}$, $K'_{OT_{i,j}} = (K_{OT_{i,j}})^2 \mod n_j$ and $C^i = PRGN(K_{TID_{i,j}} \oplus r_{i,j} \oplus n_r)$. Then it sends C^i and $K'_{OT_{i,j}}$ to Current Mobile Reader R_j.
- Current Mobile Reader $R_j \longrightarrow$ Cloud Server
  R_j once receipts the message, obtains $K_{TID_{i,j}}$ from $K'_{OT_{i,j}}$ by using Chinese Reminder Theorem (CRT) and verifies its correctness, also cross checks with

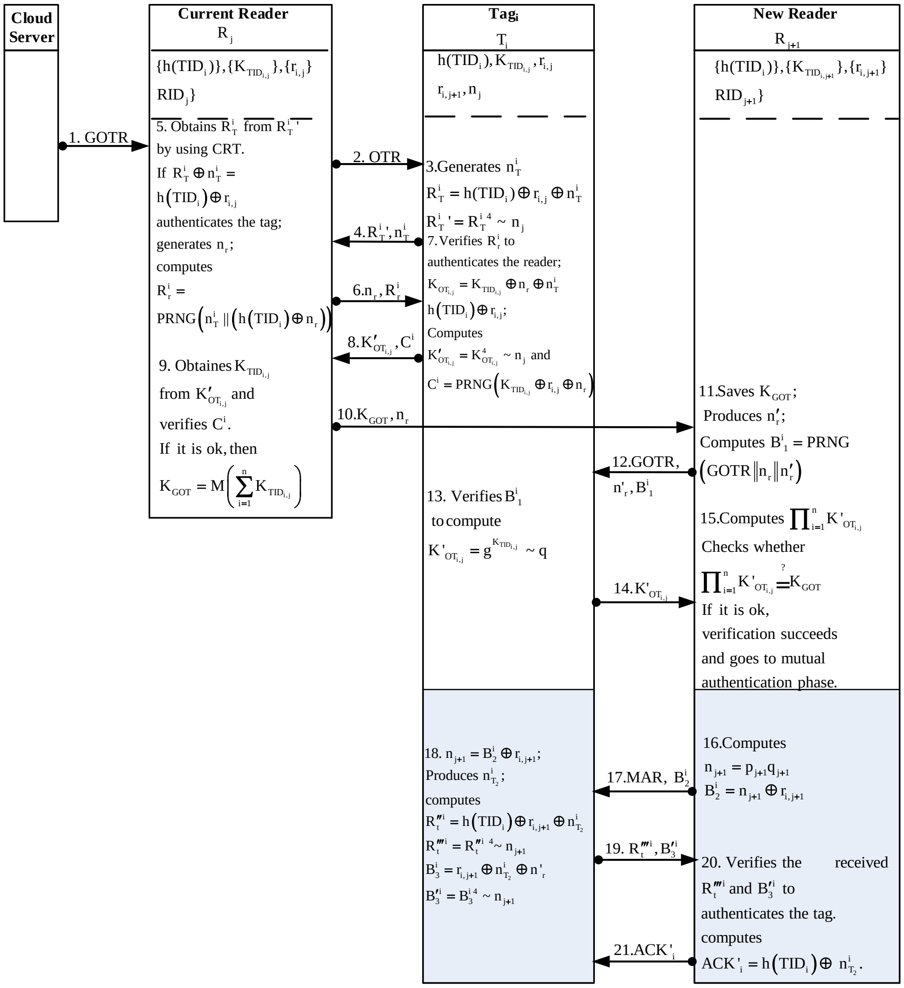

Cloud Server
1. GOTR
Current Reader
R_j
$\{h(TID_i)\},\{K_{TID_{i,j}}\},\{r_{i,j}\}$
$RID_j\}$
5. Obtains R_T^i from $R_T^i{}'$
by using CRT.
If $R_T^i \oplus n_T^i =$
$h(TID_i) \oplus r_{i,j}$
authenticates the tag;
generates n_r;
computes
$R_r^i =$
$PRNG(n_T^i \,\|\, (h(TID_i) \oplus n_r))$
9. Obtaines $K_{TID_{i,j}}$
from $K'_{OT_{i,j}}$ and
verifies C^i.
If it is ok, then
$K_{GOT} = M\left(\sum_{i=1}^{n} K_{TID_{i,j}}\right)$
2. OTR
4. $R_T^i{}', n_T^i$
6. n_r, R_r^i
8. $K'_{OT_{i,j}}, C^i$
10. K_{GOT}, n_r
Tag$_i$
T_i
$h(TID_i), K_{TID_{i,j}}, r_{i,j}$
$r_{i,j+1}, n_j$
3. Generates n_T^i
$R_T^i = h(TID_i) \oplus r_{i,j} \oplus n_T^i$
$R_T^i{}' = R_T^{i\,4} \sim n_j$
7. Verifies R_r^i to
authenticates the reader;
$K_{OT_{i,j}} = K_{TID_{i,j}} \oplus n_r \oplus n_T^i$
$h(TID_i) \oplus r_{i,j}$;
Computes
$K'_{OT_{i,j}} = K^4_{OT_{i,j}} \sim n_j$ and
$C^i = PRNG(K_{TID_{i,j}} \oplus r_{i,j} \oplus n_r)$
13. Verifies B_1^i
to compute
$K'_{OT_{i,j}} = g^{K_{TID_{i,j}}} \sim q$
18. $n_{j+1} = B_2^i \oplus r_{i,j+1}$;
Produces $n_{T_2}^i$;
computes
$R_t''^i = h(TID_i) \oplus r_{i,j+1} \oplus n_{T_2}^i$
$R_t'''^i = R_t''^{i\,4} \sim n_{j+1}$
$B_3^i = r_{i,j+1} \oplus n_{T_2}^i \oplus n'_r$
$B_3'^i = B_3^{i\,4} \sim n_{j+1}$
12. GOTR,
n'_r, B_1^i
14. $K'_{OT_{i,j}}$
17. MAR, B_2^i
19. $R_t'''^i, B_3'^i$
21. ACK'_i
New Reader
R_{j+1}
$\{h(TID_i)\},\{K_{TID_{i,j+1}}\},\{r_{i,j+1}\}$
$RID_{j+1}\}$
11. Saves K_{GOT};
Produces n'_r;
Computes $B_1^i = PRNG$
$(GOTR \,\|\, n_r \,\|\, n'_r)$
15. Computes $\prod_{i=1}^{n} K'_{OT_{i,j}}$
Checks whether
$\prod_{i=1}^{n} K'_{OT_{i,j}} \overset{?}{=} K_{GOT}$
If it is ok,
verification succeeds
and goes to mutual
authentication phase.
16. Computes
$n_{j+1} = p_{j+1} q_{j+1}$
$B_2^i = n_{j+1} \oplus r_{i,j+1}$
20. Verifies the received
$R_t'''^i$ and $B_3'^i$ to
authenticates the tag.
computes
$ACK'_i = h(TID_i) \oplus n_{T_2}^i$.

Fig. 4. The part 1 and part 2 of proposed ownership transfer protocol, where G/OTR and MAR respectively denote "group/ownership transfer request" and "mutual authentication request". From step 16 onward is the mutual authentication phase of the proposed protocol between the tag and the new owner

the received C^i. Finally, R_j transmits K_{GOT} and n_r to the cloud server via a secure channel and cloud server sends aggregated ownership key K_{GOT}, n_r and n_T^i of each tag T_i to the new mobile reader R_{j+1} via a secure channel (please note that in Figure 4 for simplicity we just mentioned messages go from R_j to R_{j+1} but in reality it happens through the cloud server). New mobile reader R_{j+1} stores K_{GOT} and generates another random number n_r' then computes $B_1^i = PRNG(GOTR \| n_r \| n_r')$.

After the above steps, part 2 of the proposed protocol is started.

- New Mobile Reader $R_{j+1} \longrightarrow$ any Tag T_i in the group
  New mobile reader R_{j+1} transmits Group OT Request (GOTR), n_r' and B_1^i to the tags.
- Tag $T_i \longrightarrow$ New Mobile Reader R_{j+1}
  Tag T_i checks whether $PRNG(GOTR \| n_r \| n_r') \overset{?}{=} B_1^i$, if the equality holds then it computes $K'_{OT_{i,j}} = g^{K_{TID_{i,j}}} \mod q$ and transmits $K'_{OT_{i,j}}$ to R_{j+1}.
- New Mobile Reader $R_{j+1} \longrightarrow$ Tag T_i

  Once the new mobile reader received the answers of all tags in the group, it first checks whether $\prod_{i=1}^{n} K'_{OT_{i,j}} \overset{?}{=} K_{GOT}$. If they are equal, then it computes $B_2^i = n_{j+1} \oplus n_T^i$ and broadcasts Mutual Authentication Request (MAR) and B_2^i to each tag in the tag group.
- Tag $T_i \longrightarrow$ New Mobile Reader R_{j+1}

  T_i, first, computes $n_{j+1} = B_2^i \oplus n_T^i$ then generates $n_{T_2}^i$ and also computes $R_T''^i = h(TID_i) \oplus r_{i,j+1} \oplus n_{T_2}^i$, $R_T'''^i = (R_T''^I)^4 \mod n_{j+1}$, $B_3^i = r_{i,j+1} \oplus n_{T_2}^i \oplus n_r'$ and $B_3'^i = {B_3^i}^4 \mod n_{j+1}$ and transmits $R_T'''^i$ and $B_3'^i$ to R_{j+1}.
- New Mobile Reader $R_{j+1} \longrightarrow$ Tag T_i

  Reader R_{j+1} when receiving $R_T'''^i$ and $B_3'^i$ from a tag, retrieves B_3^i from $B_3'^i$ and then it computes $n_{T_2}^i = B_3^i \oplus r_{i,j+1} \oplus n_r'$ and solves quadratic congruence $R_T'''^i = (R_T''^i)^4 \mod n_{j+1}$ and derives $R_T''^i$. Now if $R_T''^i \oplus n_{T_2}^i = h(TID_i) \oplus r_{i,j+1}$ then it authenticates the tag and R_{j+1} computes $ACK_i' = h(TID_i) \oplus n_{T_2}^i$ and sends ACK_i' to the tag T_i. The tag also verifies the received ACK_i' to do the final authentication of R_{j+1}.
  Part 3 of the proposed protocol proceeds the same as part 3 of Lee *et al.* protocol, exclude that the $r_{i,j+2}$ is calculated as $H(C_{i,j+1} \oplus K_{TID_{i,j+1}} \| r_{i,j+1} \oplus s_i)$ while in Lee *et al.* protocol it was computed as $C_{i,j+1} \oplus H(K_{TID_{i,j+1}} \| r_{i,j+1})$ $\oplus s_i$. So we won't repeat the details of this phase.

6 Security Proof of Proposed Protocol

In this section, we prove the security of proposed group ownership protocol using informal method and also manual formal method and automatic formal method. It should be noted we consider the security of whole the group and consider the compromising a single tag as the compromising of whole the group of the tags.

6.1 Informal Security Proof

In this section, we informally prove the proposed protocol has a high level of security and can resist against attacks presented in this paper and also all other active and passive attacks.

Secret Disclosure Attack The adversary can obtain the messages $\{R_T^{\prime i}, n_T^i, n_r, R_r^i, K'_{OT_{i,j}}, C^i\}$ from eavesdropping of the communication between Tag T_i and reader R_j and the messages $\{n'_r, B_1^i, K'_{OT_{i,j}}, B_2^i, R_t^{\prime\prime\prime i}, B_3^{\prime i}, ACK'_i\}$ from the communication between Tag T_i and reader R_{j+1}. The messages $R_T^{\prime i}$, $R_t^{\prime\prime\prime i}$ and $B_3^{\prime i}$ are protected by quadratic residue. Hence, the adversary cannot obtain any sensitive information from these messages. On the other hand the transmitted messages that contain $h(TID_i)$ are $R_T^{\prime i}$, R_r^i, $K'_{OT_{i,j}}$, $R_t^{\prime\prime\prime i}$ and ACK'_i. Messages $R_T^{\prime i}$, $K'_{OT_{i,j}}$ and $R_t^{\prime\prime\prime i}$ are protected by quadratic residue, R_r^i is protected by 128-length output PRNG and ACK'_i is xor of $h(TID_i)$ and the random number $n_{T_2}^i$. The random number $n_{T_2}^i$ is used only in the $R_T^{\prime\prime\prime i}$ and B'_3 that both are protected by quadratic residue. So the adversary cannot obtain $h(TID_i)$ and $n_{T_2}^i$. By the above discussion since the adversary cannot obtain $h(TID_i)$, he/she cannot obtain any information from the message $K_{OT_{i,j}}$. Also since the adversary cannot obtain the random number n_{T2}^i, hence ACK'_i dose not leak any information. Also, no information can leak from the message B_2^i, since it is XOR of $r_{i,j+1}$ and n_{j+1}, that only used in this message and are updated in part 3 and in the initialization phase of the proposed protocol respectively.

Reader/Tag Impersonation Attack Resistance Reader impersonation attack occurs when the adversary can play the role of the reader in the authentication phase to convince tag that accepts it as a legal reader. So the adversary must pass the authentication phase. To do this he/she needs to produce ACK'_i, but by discussion in the subsection 6.1 the adversary could not obtain $h(TID_i)$. So the proposed protocol is resistance against reader impersonation attack.

Tag impersonation attack occurs when the adversary can play the role of tag in the authentication phase to convince the reader that accepts it as a legal tag. So the adversary must pass the authentication phase. To do this he/she needs to produce $\{R_T^{\prime i}, C^i\}$ or $\{B_3^{\prime i}, R_t^{\prime\prime\prime i}\}$ that by discussion in the subsection 6.1 the proposed protocol is resistance against tag impersonation attack.

Traceability Attack Resistance Traceability attack resistance or tag location privacy means that the attacker cannot trace the location of the tag. To do this a scheme should design in which the transmitted messages makes looks irrelevant to the eavesdropper and also an attacker cannot obtain any information from the identity of tag. Since the protocol is designed so that any transmitted messages from the tag has new random numbers so the adversary cannot find any relation between transmitted messages and identity of the tag that transmits it. Also, by discussion in subsection 6.1 one can see that $h(TID_i)$ and any information about it is protected in the protocol.

Forward/Backward Secrecy Forward secrecy is achieved when the adversary cannot obtain any sensitive parameters in the past communication sessions even if the secret key or ownership key of the tag in the current communication session are revealed. By discussion in the subsection 6.1, the adversary could not obtain any sensitive information by eavesdropping. Given that the adversary has ownership key $K_{TID_{i,j+1}}$ of tag T_i and reader R_{j+1} and secret keys $\{r_{i,j+1}, r_{i,j+2}, n_{j+1}\}$ and its goal is to obtain at least one of the secret parameters $\{h(TID_i), K_{TID_{i,j}}, r_{i,j}, n_j\}$. In part 3, $K_{TID_{i,j}}$, $h(TID_i)$ and $r_{i,j+1}$ is used as the seed of the PRNG or hash function. So the adversary cannot use the information of $r_{i,j+1}$, n_{j+1} at all. A similar discussion shows that information about $r_{i,j+2}$ cannot be useful for the adversary. Hence the proposed protocol achieves forward secrecy. Likewise, the proposed protocol achieves backward secrecy.

Preventing Plaintext Miscalculation Lee *et al.* in their protocol in somewhere used equations in the form of $R = x^2 \mod n$ where n is the product of two odd prime numbers which according to Chinese Reminder Theorem has four solutions [18]. Since in the proposed protocol we use the equations in the form of $R = x^4 \mod n$ everywhere except one in the third part of the protocol and as described in [8], the equations in the form of $R = x^4 \mod n$ has only one solution, therefore our proposed protocol is secure against plaintext miscalculation attack which at first published in [13]. In the third part of proposed protocol same as it predecessor on the tag's side, it computes $x'_i = x_i^2 \mod n_{j+1}$, $x''_i = x_i^4 \mod n_{j+1}$ and $B_4^i = PRNG(x''_i \| r_{i,j+1} \| x_i)$ and sends B_4^i and x''_i to the R_{j+1}. New reader retrieves only one answer for x_i and then it computes x'_i as $x_i^2 \mod n_{j+1}$. Equation $x'_i = x_i^2 \mod n_{j+1}$ not going to solve where in the protocol there are going to be four answers. So our proposed protocol same as it predecessor is resistant against plaintext miscalculation.

Desynchronization Attack Resistance To achieve desynchronization attack, the tag and the readers need to update their secret parameters when authenticate each other and also believe that the messages that use to authenticate each other are fresh and belong to the current session. In the proposed protocol tag and readers generate random numbers n_T^i,n_{T2}^i, n_{T3}^i, n_r and n'_r to avoid desynchronization attack.

6.2 Formal security proof

Lee *et al.* protocol based on Borrows-Abadi-Needham (BAN) logic showed their protocol is logically correct and can provide high-level security and privacy protection [14]. In this section, we first argue why the BAN logic in their verification showed their protocol security correctness while in reality, it is not such a secure protocol. Then we prove the proposed protocol's security correctness based on BAN logic. We also use Scyther [6] tool which is an automatic security protocol verifier to assist the results which are retrieved from BAN logic.

On BAN Logic proof of Lee *et al.* protocol

Lee *et al.* in their BAN logic proof of the security of their protocol [14], to describe the transferred messages of protocol in BAN logic notations, have shown messages such as A_1^i which is computed only from XORing some secret values in the form of one message which is encrypted with secret values and based on it the results have arrived, while such messages should not be assumed as encrypted messages. Hence, we did not show messages that do not have sufficient security to be encrypted messages in our BAN logic proof of the proposed protocol.

BAN logic proof of the proposed protocol
BAN logic security correctness proof of protocols includes four steps:

- explaining the protocol messages in BAN logic notation: in this step, the protocol messages, using BAN logic notations introduced in Table 2, are written as depicted in Table 3: It worth noting since the channels between readers and cloud server are secure we use $K_{(CS,R_j)}$ and $K_{(CS,R_{j+1})}$ to show that the messages between R_j and CS and also between R_{j+1} and CS are encrypted with those symmetric secret keys respectively.

Table 2. Notations used in this paper

Notations	Description
$\sharp(X)$	The message X is fresh
$\{X\}_K$	The encrypted of X by using K as the key
$P \lhd X$	P sees the message X
$P \overset{K}{\longleftrightarrow} Q$	K is secretly shared between P and Q
$P\| \overset{K^{-1}}{=>}$	K^{-1} is the private key of P and its public key is K
$R1 : \dfrac{P\| \overset{K^{-1}}{=>}, P \lhd \{X\}_K}{P \lhd X}$	That means if P believes the message K^{-1} is its secret key and K is its public key and received the message $\{X\}_K$ then it entitled that he sees X
$R2 : \dfrac{P\| \equiv P \overset{K}{\longleftrightarrow} Q, P \lhd \{X\}_K}{P\| \equiv Q\| \sim \sharp X}$	That means if P believes the K is securely shared between itself and Q and also sees the $\{X\}_K$, then it entitled that he believes Q sends message X

- Idealization of protocol messages: in this step, the message which do not increase the confidence are omitted as depicted in Table 4.

Table 3. Expressing the messages of proposed protocol in BAN logic

No. of messages	Description
M_1	$R_j \lhd \{GOTR\}_{K_{(CS,R_j)}}$
M_2	$T_i \lhd OTR$
M_3	$R_j \lhd \{h(TID_i), n_T^i\}_{r_{i,j},n_j}, n_T^i$
M_4	$T_i \lhd n_r, R_r^i$
M_5	$R_j \lhd C^i, \{h(TID_i), n_r, n_T^i\}_{K_{TID_{i,j}},r_{i,j},n_j}$
M_6	$R_{j+1} \lhd n_r, K_{GOT}$
M_7	$T_i \lhd\ GOTR, n'_r, B_1^i$
M_8	$R_{j+1} \lhd K'_{OT_{i,j}}$
M_9	$T_i \lhd MAR, \{n_{j+1}\}_{r_{i,j+1}}$
M_{10}	$R_{j+1} \lhd \{h(TID_i), n_{T_2}^i\}_{r_{i,j+1},n_{j+1}}, \{n_{T_2}^i, n'_r\}_{r_{i,j+1}}$
M_{11}	$T_i \lhd ACK'_i$
M_{12}	$R_{j+1} \lhd \{K_{TID_{i,j}}, n_{T_3}^i\}_{r_{i,j+1},n_{j+1}}, \{K_{TID_{i,j}}, n_{T_3}^i\}_{n_{j+1}}$
M_{13}	$CS \lhd \{I_{i,j+1}\}_{K_{(CS,R_{j+1})}}$
M_{14}	$R_{j+1} \lhd \{C_{i,j+1}\}_{K_{(CS,R_{j+1})}}$
M_{15}	$T_i \lhd \{n_{T_2}^i, K_{TID_{i,j}}, n_{T_3}^i, \{K_{TID_{i,j+1}}, K_{TID_{i,j}}, RID_{j+1}, n_{T_3}^i\}_{n_{j+1}}$ $, C_{i,j+1}\}_{n_{j+1}}, \{K_{TID_{i,j+1}}, K_{TID_{i,j}}, RID_{j+1}, n_{T_3}^i\}_{n_{j+1}}, C_{i,j+1}$
M_{16}	$CS \lhd \{C'_{i,j+1}, I'_{i,j+1}\}_{K_{(CS,R_{j+1})}}$

Table 4. Idealization of the messages of proposed protocol in BAN logic

No. of idealized messages	Description
IM_3	$R_j \lhd \{h(TID_i), n_T^i\}_{r_{i,j},n_j}$
IM_4	$T_i \lhd n_r, R_r^i$
IM_5	$R_j \lhd C^i, \{h(TID_i), n_r, n_T^i\}_{K_{TID_{i,j}},r_{i,j},n_j}$
IM_6	$R_{j+1} \lhd n_r, K_{GOT}$
IM_7	$T_i \lhd n'_r, B_1^i$
IM_8	$R_{j+1} \lhd K'_{OT_{i,j}}$
IM_9	$T_i \lhd \{n_{j+1}\}_{r_{i,j+1}}$
IM_{10}	$R_{j+1} \lhd \{h(TID_i), n_{T_2}^i\}_{r_{i,j+1},n_{j+1}}, \{n_{T_2}^i, n'_r\}_{r_{i,j+1},n_{j+1}}$
IM_{11}	$T_i \lhd Ack'_i$
IM_{12}	$R_{j+1} \lhd \{K_{TID_{i,j}}, n_{T_3}^i\}_{r_{i,j+1},n_{j+1}}, \{K_{TID_{i,j}}, n_{T_3}^i\}_{n_{j+1}}$
IM_{15}	$T_i \lhd \{n_{T_2}^i, K_{TID_{i,j}}, n_{T_3}^i, \{K_{TID_{i,j+1}}, K_{TID_{i,j}}, RID_{j+1}$ $, n_{T_3}^i\}_{n_{j+1}}, C_{i,j+1}\}_{n_{j+1}}, \{K_{TID_{i,j+1}}, K_{TID_{i,j}}, RID_{j+1},$ $n_{T_3}^i\}_{n_{j+1}}, C_{i,j+1}$

– expressing the protocol security assumptions and goals: the proposed protocol's assumptions and goals are respectively as depicted in Table 5.

Table 5. Assumption and security goals of proposed protocol

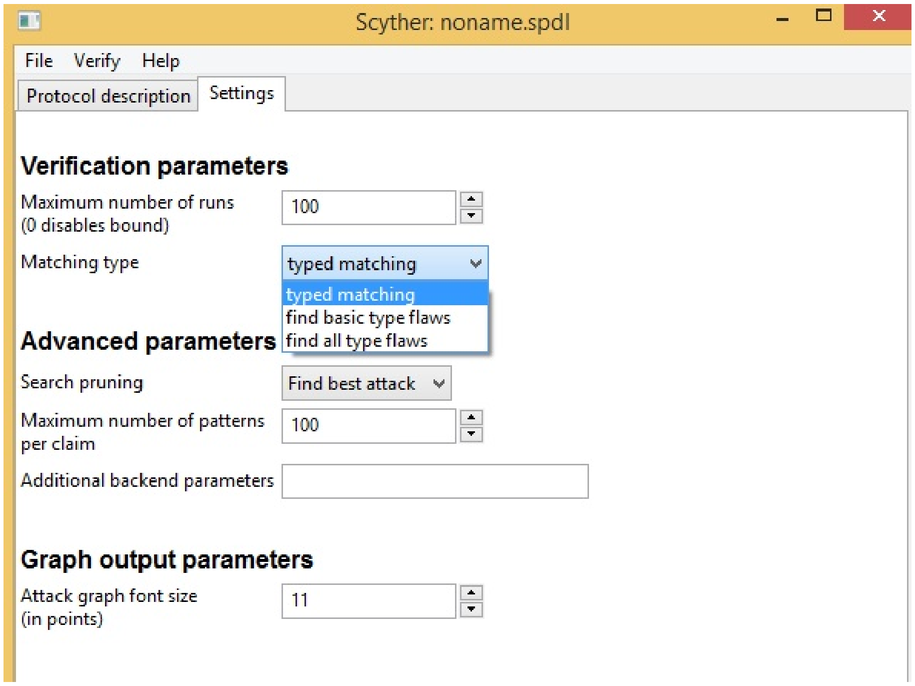
No. of Assumption /Goal	Description	No. of Assumption /Goal	Description
$A1$	$T_i\vert \equiv \sharp n_T^i$	$A14$	$R_j\vert \equiv R_j \overset{r_{i,j}}{\longleftrightarrow} T_i$
$A2$	$T_i\vert \equiv \sharp n_{T_2}^i$	$A15$	$T_i\vert \equiv T_i \overset{r_{i,j+1}}{\longleftrightarrow} R_{j+1}$
$A3$	$T_i\vert \equiv \sharp n_{T_3}^i$	$A16$	$R_{j+1}\vert \equiv R_{j+1} \overset{r_{i,j+1}}{\longleftrightarrow} T_i$
$A4$	$R_j\vert \equiv \sharp n_r$	$A17$	$R_j\vert \overset{n_j^{-1}}{=>}$
$A5$	$R_{j+1}\vert \equiv \sharp n_r'$	$A18$	$R_{j+1}\vert \overset{n_{j+1}^{-1}}{=>}$
$A6$	$R_{j+1}\vert \equiv \sharp K_{TID_{i,j+1}}$	$A19$	$R_j\vert \equiv R_j \overset{K_{(CS,R_j)}}{\longleftrightarrow} CS$
$A7$	$T_i\vert \equiv T_i \overset{K_{TID_{i,j}}}{\longleftrightarrow} R_j$	$A20$	$CS\vert \equiv CS \overset{K_{(CS,R_j)}}{\longleftrightarrow} R_j$
$A8$	$R_j\vert \equiv R_j \overset{K_{TID_{i,j}}}{\longleftrightarrow} T_i$	$A21$	$R_{j+1}\vert \equiv R_{j+1} \overset{K_{(CS,R_{j+1})}}{\longleftrightarrow} CS$
$A9$	$T_i\vert \equiv T_i \overset{h(TID_i)}{\longleftrightarrow} R_j$	$A21$	$CS\vert \equiv CS \overset{K_{(CS,R_{j+1})}}{\longleftrightarrow} R_{j+1}$
$A10$	$R_j\vert \equiv R_j \overset{h(TID_i)}{\longleftrightarrow} T_i$	$G1$	$R_j\vert \equiv T_i\vert \sim \sharp n_T^i$
$A11$	$T_i\vert \equiv T_i \overset{h(TID_i)}{\longleftrightarrow} R_{j+1}$	$G2$	$R_{j+1}\vert \equiv T_i\vert \sim \sharp\{n_{T_2}^i, n_{T_3}^i\}$
$A12$	$R_{j+1}\vert \equiv R_{j+1} \overset{h(TID_i)}{\longleftrightarrow} T_i$	$G3$	$T_i\vert \equiv R_j\vert \sim \sharp n_{j+1}$
$A13$	$T_i\vert \equiv T_i \overset{r_{i,j}}{\longleftrightarrow} R_j$	-	-

– Deducing security goals:

Using $IM3$ and $A17$ based on $R1$, we deduce $D1 : R_j \lhd \{h(TID_i), n_T^i\}_{r_{i,j}}$ After that using $D1$ and $A14$ based on R_2, we deduce $D2 : R_j| \equiv T_i| \sim \sharp n_T^i$ which is the $G1$ security goal. Using $IM10$ and $A18$ based on $R1$, we deduce $D3 : R_{j+1} \lhd \{n_{T_2}^i, n_r'\}_{r_{i,j+1}}$ After that using $D3$ and $A16$ based on R_2, we deduce $D4 : R_{j+1}| \equiv T_i| \sim \sharp n_{T_2}^i$ which is the $G2$ security goal. Using $IM12$ and $A18$ based on $R1$, we deduce $D5 : R_{j+1} \lhd \{K_{TID_{i,j}}, n_{T_3}^i\}_{r_{i,j+1}}$ After that using $D5$ and $A16$ based on R_2, we deduce $D6 : R_{j+1}| \equiv T_i| \sim \sharp n_{T_3}^i$ which is the $G2$ security goal. Using $IM9$ and $A15$ based on R_2, we deduce $D7 : T_i| \equiv R_j| \sim \sharp n_{j+1}$ which is the $G3$ security goal.

Scyther proof of the proposed protocol

Scyther [6] is an automatic security protocol verification tool which receives the protocol implementation in Security Protocol Description Language (i.e. spdl) and based on its verification setting which is depicted in Figure 5, analyses the protocol. If the targeted protocol is not secure it is this ability to show the attack scenario. Our proposed protocol implementation code in spdl are shown in Appendix A and its verification result through Scyther is depicted in Figure 6.



Verification parameters

Maximum number of runs (0 disables bound): 100

Matching type: typed matching

- typed matching
- find basic type flaws
- find all type flaws

Advanced parameters

Search pruning: Find best attack

Maximum number of patterns per claim: 100

Additional backend parameters:

Graph output parameters

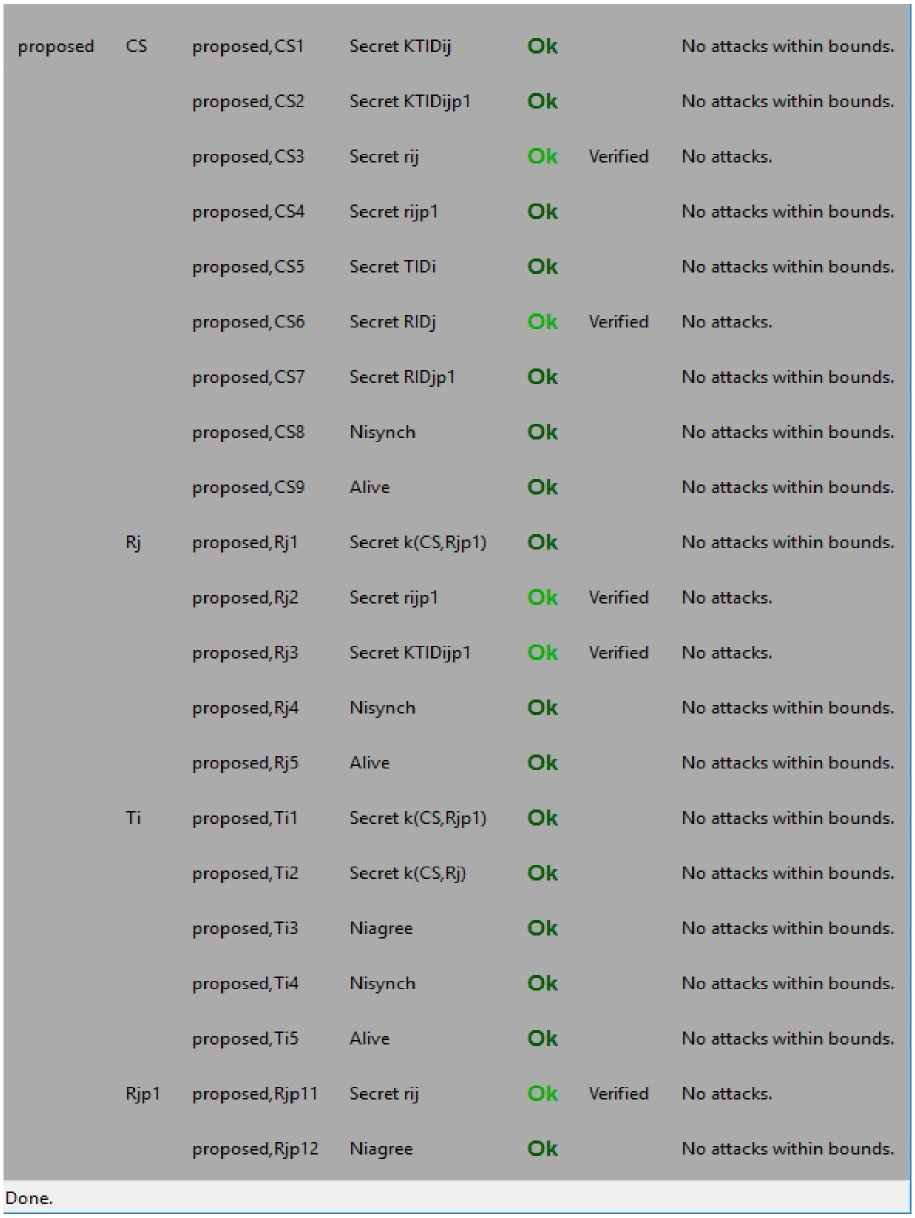
Attack graph font size (in points): 11

Fig. 5. The setting parameters of the Scyther tool

It is easily seen in Figure 6 that the Scyther tool could not find any security attacks against the protocol. So the proposed protocol provides a high level of security for secure group ownership goals.

7 Evaluation of Proposed Protocol

In this section, we compare the improved protocol with several recent group ownership transfer protocols, in terms of security properties and also in terms of computational and communication costs. As shown in Tables 6 and 7 the proposed protocol has been able to achieve a high level of security with little change over its predecessor i.e. [14].



```
proposed  CS    proposed,CS1    Secret KTIDij       Ok             No attacks within bounds.
                proposed,CS2    Secret KTIDijp1     Ok             No attacks within bounds.
                proposed,CS3    Secret rij          Ok   Verified  No attacks.
                proposed,CS4    Secret rijp1        Ok             No attacks within bounds.
                proposed,CS5    Secret TIDi         Ok             No attacks within bounds.
                proposed,CS6    Secret RIDj         Ok   Verified  No attacks.
                proposed,CS7    Secret RIDjp1       Ok             No attacks within bounds.
                proposed,CS8    Nisynch             Ok             No attacks within bounds.
                proposed,CS9    Alive               Ok             No attacks within bounds.
          Rj    proposed,Rj1    Secret k(CS,Rjp1)   Ok             No attacks within bounds.
                proposed,Rj2    Secret rijp1        Ok   Verified  No attacks.
                proposed,Rj3    Secret KTIDijp1     Ok   Verified  No attacks.
                proposed,Rj4    Nisynch             Ok             No attacks within bounds.
                proposed,Rj5    Alive               Ok             No attacks within bounds.
          Ti    proposed,Ti1    Secret k(CS,Rjp1)   Ok             No attacks within bounds.
                proposed,Ti2    Secret k(CS,Rj)     Ok             No attacks within bounds.
                proposed,Ti3    Niagree             Ok             No attacks within bounds.
                proposed,Ti4    Nisynch             Ok             No attacks within bounds.
                proposed,Ti5    Alive               Ok             No attacks within bounds.
          Rjp1  proposed,Rjp11  Secret rij          Ok   Verified  No attacks.
                proposed,Rjp12  Niagree             Ok             No attacks within bounds.
Done.
```

Fig. 6. The Scyther security verification of our proposed group ownership transfer protocol

Table 6. Security Comparison

Protocol	A1	A2	A3	A4	A5	A6
[21]	× [5]	× [5]	✓	✓	✓	✓
[20]	✓	× [15]	× [15]	× [15]	× [15]	✓
[2]	✓	× [14]	× [14]	× [14]	✓	× [2]
[23]	✓	✓	×(Sec. 4.5)	✓	✓	✓
[14]	×(Sec. 4)	×(Sec. 4)	×(Sec. 4)	× (Sec. 4)	×(Sec. 4)	✓
Ours	✓	✓	✓	✓	✓	✓

A1: Secret Disclosure attack; A2:Impersonation and Replay attack;
A3: Desynchronization attack; A4: Traceability attack and Anonymity;
A5: Backward/Forward Secrecy Contradiction ;
A6: Group Ownership Transfer Property;
✓: Resistant × : Vulnerable

Table 7. Complexity Comparison

Protocol	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
[21]	3	-	-	-	-	-	14	-	6	12L
[20]	-	-	-	-	-	90	2	34	-	16L
[2]	18	1	3	-	2	32	34	6	-	28L
[23]	-	-	-	-	-	20	29	29	-	18L
[14]	10	1	2	1	3	46	32	12	-	28L
Ours	10	3	6	1	3	43	29	11	-	28L

F1: Number of $h(.)$; F2: Number of two modular exponentiation
F3: Number of four modular exponentiation; F4: Solving square modular
F5: Solving four modular; F6: Number of$\oplus$
F7: Number of$\|$; F8: Number of $PRNG$
F9: Number of $E(.)/D(.)$ F10:Number of transferred bits

As can be seen in the Table 7, the proposed protocol differs slightly from the Lee *et al.* protocol in terms of the number of operations used, but in turn has been able to provide important security features same as resistance against secret disclosure, forward/backward secrecy, resistance against all kinds of impersonation and replay attacks, untraceability and anonymity property.

8 Conclusion

In this paper we considered security pitfalls of two group ownership transfer protocols i.e. Lee *et al.* and Zhu *et al.* . Precisely, we presented desynchronization attack against Zhu *et al.* protocol with success probability of "1"and the complexity of five runs of protocol. Moreover, we outlined the security breaches of the Lee *et al.* protocol and their revision to the secure ownership transfer protocol, and proved its security in three ways: informal method, manual formal method and automatic formal method. More precisely, we implemented three strong attacks against Lee *et al.* protocol including secret disclosure attack, reader impersonation attack, forward secrecy contradiction attack and traceability attack with the success probability of "1" and the complexity of only one run of the protocol.

The attacks presented in this paper, showed that the Lee *et al.* group ownership protocol was by no means an appropriate protocol for security purposes. It is hoped that such security analysis will lead to further development of the science of security protocol design so that we can see more secure protocols in the future.

References

1. S. Cai, Y. Li, T. Li, and R. H. Deng. Attacks and improvements to an RIFD mutual authentication protocol and its extensions. In *Proceedings of the second ACM conference on Wireless network security*, pages 51–58. ACM, 2009.
2. T. Cao, X. Chen, R. Doss, J. Zhai, L. J. Wise, and Q. Zhao. RFID ownership transfer protocol based on cloud. *Computer Networks*, 105:47–59, 2016.
3. Y. Chen and J.-S. Chou. ECC-based untraceable authentication for large-scale active-tag RFID systems. *Electronic Commerce Research*, 15(1):97–120, 2015.
4. H.-Y. Chien. De-synchronization attack on quadratic residues-based RFID ownership transfer. In *2015 10th Asia Joint Conference on Information Security*, pages 42–47. IEEE, 2015.
5. G. Cong, Z.-j. ZHANG, L.-h. ZHU, Y.-a. TAN, and Y. Zhen. A novel secure group RFID authentication protocol. *The Journal of China Universities of Posts and Telecommunications*, 21(1):94–103, 2014.
6. C. J. F. Cremers. The Scyther tool: Verification, falsification, and analysis of security protocols. In A. Gupta and S. Malik, editors, *Computer Aided Verification*, pages 414–418, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
7. T. Dimitriou. Key evolving RFID systems: Forward/backward privacy and ownership transfer of RFID tags. *Ad Hoc Networks*, 37:195–208, 2016.
8. R. Doss, W. Zhou, and S. Yu. Secure RFID tag ownership transfer based on quadratic residues. *IEEE Transactions on Information Forensics and Security*, 8(2):390–401, 2012.

9. S. Edelev, S. Taheri, and D. Hogrefe. A secure minimalist RFID authentication and an ownership transfer protocol compliant to EPC C1G2. In *2015 IEEE International Conference on RFID Technology and Applications (RFID-TA)*, pages 126–133. IEEE, 2015.
10. H. Jannati and A. Falahati. Cryptanalysis and enhancement of a secure group ownership transfer protocol for RFID tags. In *Global Security, Safety and Sustainability & e-Democracy*, pages 186–193. Springer, 2011.
11. A. Juels and S. A. Weis. Defining strong privacy for RFID. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):7, 2009.
12. G. Kapoor, W. Zhou, and S. Piramuthu. Multi-tag and multi-owner RFID ownership transfer in supply chains. *Decision Support Systems*, 52(1):258–270, 2011.
13. C.-C. Lee, S.-D. Chen, C.-T. Li, C.-L. Cheng, and Y.-M. Lai. Security enhancement on anRFID ownership transfer protocol based on cloud. *Future Generation Computer Systems*, 93:266–277, 2019.
14. C.-C. Lee, C.-T. Li, C.-L. Cheng, and Y.-M. Lai. A novel group ownership transfer protocol for RFID systems. *Ad Hoc Networks*, 91, 2019.
15. J. Munilla, M. Burmester, and A. Peinado. Attacks on ownership transfer scheme for multi-tag multi-owner passive RFID environments. *Computer Communications*, 88:84–88, 2016.
16. S. Qi, Y. Zheng, M. Li, L. Lu, and Y. Liu. Secure and private RFID-enabled third-party supply chain systems. *IEEE Transactions on Computers*, 65(11):3413–3426, 2016.
17. B. R. Ray, J. Abawajy, M. Chowdhury, and A. Alelaiwi. Universal and secure object ownership transfer protocol for the Internet of Things. *Future Generation Computer Systems*, 78:838–849, 2018.
18. K. H. Rosen. *Elementary number theory*. Pearson Education, 2011.
19. S. Sundaresan, R. Doss, W. Zhou, and S. Piramuthu. Secure ownership transfer for multi-tag multi-owner passive RFID environment with individual-owner-privacy. *Computer Communications*, 55:112–124, 2015.
20. S. Sundaresan, R. Doss, W. Zhou, and S. Piramuthu. Secure ownership transfer for multi-tag multi-owner passive RFID environment with individual-owner-privacy. *Computer Communications*, 55:112–124, 2015.
21. M. H. Yang. Secure multiple group ownership transfer protocol for mobile RFID. *Electronic Commerce Research and Applications*, 11(4):361–373, 2012.
22. R. Zhang, L. Zhu, C. Xu, and Y. Yi. An efficient and secure RFID batch authentication protocol with group tags ownership transfer. In *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*, pages 168–175. IEEE, 2015.
23. D. Zhu, W. Rong, D. Wu, and N. Pang. Lightweight anonymous RFID group ownership transfer protocol in multi-owner environment. volume 2018-January, pages 404–411, 2018.

A The Scyther Code of the Proposed Protocol

```
hashfunction h;
hashfunction H;
const xor:Function;
const con:Function;
const inc:Function;
const PRNG:Function;
const MUL:Function;
const MUL2:Function;
const MUL4:Function;
const MULg:Function;
const HM:Function;
const GOTR;
const OTR;
const MAR;
secret TIDi;
secret hTIDi;
secret nj;
secret qj;
secret pj;
secret KTIDij;
secret KTIDnj;
secret rij;
secret RIDj;
secret q;
secret qjp1;
secret pjp1;
secret njp1;
secret rijp1;
secret KTIDijp1;
secret Cijp1;
macro nj=MUL(pj,qj);
macro hTIDi=h(TIDi);
macro Rit=xor(hTIDi,rij,niT);
macro Ritprim=MUL4(Rit,nj);
macro Rir=PRNG(con(niT,xor(hTIDi,nr)));
macro Ci=PRNG(xor(KTIDij,rij,nr));
macro KOTij=xor(KTIDij,hTIDi,rij,nr,niT);
macro KOTijprim=MUL4(KOTij,nj);
macro nTprim=PRNG(inc(nT));
macro KGOT=HM(KTIDij,KTIDnj);
macro Bi1=PRNG(con(GOTR,nr,nrprim));
macro KOTijprim=MULg(KTIDij,q);
macro Bi2=xor(njp1,rijp1);
macro Rtizegond=xor(hTIDi,rijp1,niT2);
macro Rti3zegond=MUL4(Rtizegond,njp1);
macro Bi3=xor(rijp1,niT2,nrprim);
macro Bi3prim=MUL4( Bi3,njp1);
macro ACKiprim=xor(hTIDi,niT2);
macro x=xor(KTIDij,niT3);
macro xprim=MUL2(x,njp1);
macro xzegond=MUL4(x,njp1);
macro Bi4=PRNG(con(xzegond,rijp1,x));
macro Iijp1=xor(RIDjp1,H(con(hTIDi, KTIDijp1,rijp1)));
macro E=xor(con(KTIDijp1,RIDjp1),xprim,x);
macro Bi5=PRNG(con(niT2,xprim,E,Cijp1));
macro rijp1prim=PRNG(xor(hTIDi,xprim));
macro Iijp1prim=xor(RIDjp1, H(con(hTIDi,KTIDijp1,rijp1prim)));
macro Cijp1prim=xor(H(con(KTIDnj,rijp1)),H(con(KTIDijp1,rijp1prim)));
protocol proposed(CS,Rj,Ti, Rjp1){
role CS{
secret KTIDijp1;
secret TIDi;
secret hTIDi;
secret rijp1;
secret RIDjp1;
secret Cijp1;
secret KTIDij;
var niT3;
secret njp1;
secret KTIDnj;
send_1(CS,Rj,{GOTR}k(CS,Rj));
recv_13(Rjp1,CS, {Iijp1}k(CS,Rjp1));
send_14(CS,Rjp1,{Cijp1}k(CS,Rjp1));
recv_16(Rjp1,CS, {Iijp1prim, Cijp1prim}k(CS,Rjp1));
claim(CS, Secret, KTIDij);
claim(CS, Secret, KTIDijp1);
claim(CS, Secret, rij);
claim(CS, Secret, rijp1);
claim(CS, Secret, TIDi);
claim(CS, Secret, RIDj);
claim(CS, Secret, RIDjp1);
claim(CS,Nisynch);
claim(CS,Alive);
}
role Rj{
var niT;
fresh nr;
secret TIDi;
secret hTIDi;
secret KTIDij;
secret KTIDnj;
secret rij;
secret RIDj;
secret qj;
secret pj;
secret q;
```

```
recv_1(CS,Rj,{GOTR}k(CS,Rj));
send_2(Rj,Ti,OTR);
recv_3(Ti,Rj,Ritprim,niT);
send_4(Rj,Ti,nr,Rir);
recv_5(Ti,Rj,Ci,KOTijprim);
send_6(Rj,Rjp1,KGOT,nr);
claim(Rj,Secret,k(CS,Rjp1));
claim(Rj,Secret, rijp1);
claim(Rj,Secret, KTIDijp1);
claim(Rj,Nisynch);
claim(Rj,Alive);
}
role Ti{
fresh niT;
fresh niT2;
fresh niT3;
var nrprim;
secret TIDi;
secret rij;
secret nj;
secret hTIDi;
secret KTIDij;
secret qj;
secret pj;
var nr;
secret q;
secret rijp1;
secret njp1;
secret qjp1;
secret pjp1;
secret KTIDijp1;
secret RIDjp1;
secret Cijp1;
recv_2(Rj,Ti,OTR);
send_3(Ti,Rj, Ritprim,niT);
recv_4(Rj,Ti,nr,Rir);
send_5(Ti,Rj,Ci,KOTijprim);
recv_7(Rjp1,Ti, GOTR ,nrprim,Bi1);
send_8(Ti,Rjp1, KOTijprim);
recv_9(Rjp1,Ti,  MAR ,Bi2);
send_10(Ti, Rjp1,Rti3zegond,Bi3prim);
recv_11(Rjp1,Ti,ACKiprim);
send_12(Ti,Rjp1,Bi4,xzegond);
recv_15(Rjp1,Ti,Cijp1,E,Bi5);
claim(Ti,Secret,k(CS,Rjp1));
claim(Ti,Secret,k(CS,Rj));
claim(Ti,Niagree);
claim(Ti,Nisynch);
claim(Ti,Alive);
}
role Rjp1{
var nr;
secret KTIDij;
secret KTIDnj;
fresh nrprim;
secret q;
secret rijp1;
secret njp1;
secret qjp1;
secret pjp1;
var niT2;
var niT3;
secret TIDi;
secret hTIDi;
secret KTIDijp1;
secret RIDjp1;
secret Cijp1;
recv_6(Rj,Rjp1,KGOT,nr);
send_7(Rjp1,Ti, GOTR ,nr,Bi1);
recv_8(Ti,Rjp1, KOTijprim);
send_9(Rjp1,Ti, MAR ,Bi2);
recv_10(Ti, Rjp1,Rti3zegond,Bi3prim);
send_11(Rjp1,Ti,ACKiprim);
recv_12(Ti,Rjp1,Bi4,xzegond);
send_13(Rjp1,CS, {Iijp1}k(CS,Rjp1));
recv_14(CS,Rjp1,{Cijp1}k(CS,Rjp1));
send_15(Rjp1,Ti,Cijp1,E,Bi5);
send_16(Rjp1,CS, {Iijp1prim, Cijp1prim}k(CS,Rjp1));
claim(Rjp1,Secret, rij);
claim(Rjp1,Niagree);
claim(Rjp1,Secret, KTIDij);
claim(Rjp1,Nisynch);
claim(Rjp1,Alive);
}
}
```