Secure Improved Cloud-Based RFID Authentication Protocol

Sarah Abughazalah(✉), Konstantinos Markantonakis, and Keith Mayes

Smart Card Centre-Information Security Group (SCC-ISG), Royal Holloway, University of London, Egham, UK
{Sarah.AbuGhazalah.2012,K.Markantonakis,Keith.Mayes}@rhul.ac.uk

Abstract. Although Radio Frequency IDentification (RFID) systems promise a fruitful future, security and privacy concerns have affected the adoption of the RFID technology. Several studies have been proposed to tackle the RFID security and privacy concerns under the assumption that the server is secure. In this paper, we assume that the server resides in the cloud that might be insecure, thus the tag's data might be prone to privacy invasion and attacks. Xie et al. proposed a new scheme called "cloud-based RFID authentication", which aimed to address the security and privacy concerns of RFID tag's data in the cloud. In this paper, we showed that the Xie et al. protocol is vulnerable to reader impersonation attacks, location tracking and tag's data privacy invasion. Hence, we proposed a new protocol that guarantees that the tag's data in the cloud are anonymous, and cannot be compromised. Furthermore, the proposed protocol achieves mutual authentication between all the entities participating in a communication session, such as a cloud server, a reader and a tag. Finally, we analysed the proposed protocol informally, and formally using a privacy model and CasperFDR. The results indicate that the proposed protocol achieves data secrecy and authentication for RFID tags.

Keywords: RFID · Cloud server · Privacy · Security protocol · Privacy model · CasperFDR

1 Introduction

Radio Frequency IDentification (RFID) is a wireless technology that uses radio signals to identify tags attached to objects [1]. A typical RFID system is composed of three main components, namely a tag, a reader and a backend server. The reader broadcasts radio frequency (RF) signals to power, send and receive data from passive RFID tags without physical contact. The RFID tag is an identification device attached to an item that transmits stored information to the nearby reader(s) through the RF channel. The reader sends the tag's data to the backend server, which stores data about the RFID tags it manages [1].

There are increasing concerns around the security and privacy of the RFID systems [2]. Communication between the reader and tag is vulnerable to interception, modification, fabrication and replay attacks. Therefore, there are extensive

J. Garcia-Alfaro et al. (Eds.): DPM/SETOP/QASA 2014, LNCS 8872, pp. 147–164, 2015.
DOI: 10.1007/978-3-319-17016-9_10

studies attempting to achieve a mutual authentication between all the parties involved in an RFID communication session. Some of these studies can be found in [3–11].

In [11], the authors divided the proposed RFID mutual authentication studies into two approaches:

- Server-based RFID mutual authentication [3,4,6,7,9]: In this approach, the tag and reader depend on the backend server to be authenticated. When the reader receives a message from the tag, it forwards the tag's message to the server to be processed. The server stores secret data related to the tags. The researchers in this scheme assume that the server is secure, and their main focus is to secure the data transmission between the tags and the readers.
- Server-less RFID mutual authentication [5,8,10]: This approach takes into account an offline authentication, where the reader authenticates the tag offline without the need to contact the server. This is based on tag authentication credentials previously stored in the reader. For instance, the reader contacts the Certification Authority (CA) during the initialisation phase to retrieve a list of legitimate tags' data.

A new approach, where the RFID tags' data can be stored in a remote server residing in the cloud, has gained increasing attention. The tags' data stored in the cloud might be vulnerable to data breach and/or privacy invasion by a malicious attacker or internal employees. Therefore, while integrating RFID into the cloud, the security and privacy of the RFID tags must be considered.

To this end, the authors in [11] proposed a new scheme called "cloud-based RFID authentication". In this scheme, the authors aimed to address the security and privacy concerns of RFID tag's data in the cloud. Three entities participate in this scheme, namely a tag, a reader and a cloud server. This scheme is shown in Fig. 1. The reader is connected to the tag through a wireless channel, while the communication between the reader and the cloud server uses a Virtual Private Network (VPN). The authors assumed that the tag's and reader's data are stored in an encrypted hash table in an untrusted cloud server. The authors proposed an RFID authentication protocol that preserves the reader's and tag's privacy against an untrusted cloud server. Improvements to this study will be the focus of this paper. The protocol details can be found in Sect. 2.

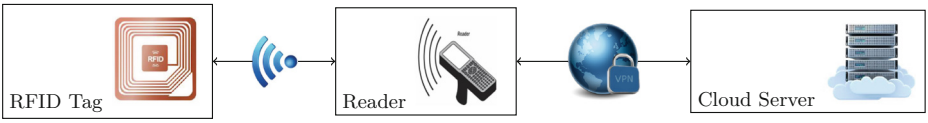


Fig. 1. The cloud-based RFID authentication approach

The authors in [11] claimed that their proposed protocol resists reader impersonation attacks as only the legitimate reader can compute the authentication messages. Furthermore, they also claimed that their protocol preserves privacy

by hashing the tag's data with a random number, which provides communications with confidentiality and freshness.

In this paper, we examined the cloud-based RFID authentication protocol [11], and we discovered the following:

1. An attacker is able to impersonate the reader without compromising the secret data shared with the tag, thus causing the tag to be updated with the wrong values and permit tracking the tag's location.
2. By using a privacy model, we found that the attacker can confirm that two sessions are related to the same tag, thus allowing unauthorised tracking.
3. By using CasperFDR, we found that the tag's secret data is not protected as the attacker can perform man-in-the-middle attacks and obtain the secret data at the end of the protocol session.

Hence, we propose a new protocol that uses some of the notations in [11] while utilising new notions that improve the cloud-based RFID authentication protocol. We called our protocol "improved cloud-based RFID authentication".

The rest of this paper is organised as follows: in Sect. 2, we demonstrate the cloud-based RFID authentication protocol. In Sect. 3, we discuss the detailed attacks on the cloud-based RFID authentication protocol. In Sect. 4, we show the improved cloud-based RFID authentication protocol in detail. In Sect. 5, we analyse the proposed protocol with respect to informal analysis, performance analysis, mechanical formal analysis (using CasperFDR) and a privacy model. In Sect. 6, we offer concluding remarks.

2 Review of the Cloud-Based RFID Authentication Protocol

This section reviews the cloud-based RFID authentication protocol as proposed in [11].

2.1 Notations

The notations are defined as follows:

R: denotes the identity of an RFID reader
T: denotes the identity of an RFID tag
S: denotes the number of authentication sessions between a reader and a tag, with bit length L
M: denotes the last number of sessions between a reader and a tag
Nr: denotes a random number generated by a reader
Nt: denotes a random number generated by a tag
PRNG(): denotes the Pseudo Random Number Generation (PRNG) function
H(): denotes a secure one-way hash function with output length L, that is, H(): $\{0,1\}^* \rightarrow \{0,1\}^L$

E(): denotes an encryption function using a symmetric algorithm with a reader secret key
D(): denotes a decryption function using a symmetric algorithm with a reader secret key
$\oplus$: denotes the XOR operation
$\|$: denotes the concatenation operation stop

2.2 The Cloud-Based RFID Authentication Protocol

During the registration phase, the reader encodes the tag with three secret values, namely R, T and S. The reader also adds unique initialised records, i.e.{H(R$\|$T$\|$S) and E(R$\|$T$\|$S)} into the cloud server. The authors assume that the registration is secure and performed in a "closed" environment. The protocol works as follows:

- The tag generates H(R$\|$T$\|$S) as an authentication request and sends it to the reader.
- The reader retrieves E(R$\|$T$\|$S) from the cloud by sending the index H(R$\|$T$\|$S) to the server. Then, the reader decrypts D(E(R$\|$T$\|$S)), verifies R and obtains T and S.
- The reader generates a random number Nr as a challenge to the tag, and sends Nr to the tag.
- The tag calculates H(R$\|$T$\|$Nr) as a response and generates a random number Nt as a challenge to the reader.
- The reader verifies the tag's response, and if valid, the next step is started; otherwise, the protocol is terminated.
- The reader tries to read the next record indexed by H(R$\|$T$\|$(S + 1)) from the cloud server and checks the integrity. If there is a valid record, this implies that the tag has been desynchronized. The reader attempts to read the $S + 2^{th}$ record indexed by H(R$\|$T$\|$(S + 2)), until finding the last valid record.
- The reader writes E(R$\|$T$\|$M') with the index H(R$\|$T$\|$M') into the cloud server, where M' = M + 1.
- The cloud sends H(R$\|$T$\|$M') $\oplus$ H(E(R$\|$T$\|$M')) to the reader to confirm that the update process has been successful.
- The reader sends the authentication messages H(R$\|$T$\|$Nt) $\oplus$ M', and H(T$\|$R$\|$ M') to the tag.
- The tag calculates H(R$\|$T$\|$Nt) XORed with the received H(R$\|$T$\|$Nt) $\oplus$ M' to obtain M', and then it calculates and verifies H(T$\|$R$\|$M'). If successful, this implies that M' is not modified by an attacker, and subsequently synchronisation is achieved again by updating S=M' on the tag. Moreover, the validity of M' also means that the reader is authenticated by the tag.

3 Weaknesses of the Cloud-Based RFID Authentication Protocol

In this section, we show the main weaknesses found in the cloud-based RFID authentication protocol.

3.1 Reader Impersonation Attack

In [11], the authors claim that their proposed protocol achieves mutual authentication between the tag and the reader, as only the legitimate reader knows the data (R, T, M) and can compute the authentication messages. However, we found that the attacker can impersonate a legitimate reader and be successfully authenticated by the tag without compromising the internal tag's data. The scenario for accomplishing this attack is as follows:

- The tag starts a normal session with the reader and sends H(R‖T‖S).
- The reader sends Nr to the tag.
- The tag generates Nt and sends H(R‖T‖Nr), and Nt to the reader.
- After the reader authenticates the tag, it asks the tag to update its value by sending N = H(R‖T‖Nt) ⊕ M, and G = H(T‖R‖M).
- The attacker blocks N and G from reaching the tag and obtains N and G for further use. As a result, the tag will not update the S value.
- Since the tag did not update the S value, the attacker will track the tag's location.
- The tag starts a new session with the reader, and sends H(R‖T‖S) to the reader.
- The reader sends Nr' to the tag.
- The tag generates Nt' and sends H(R‖T‖Nr'), and Nt' to the reader.
- After the reader authenticates the tag, it asks the tag to update its value by sending N' = H(R‖T‖Nt') ⊕ M', and G' = H(T‖R‖M'), where M' = S + 1.
- The attacker obtains N' and G', and calculates the following:
  1. Since M' = M + 1, the attacker changes the 2 least significant bits (LSB) of N to be compatible with N' when it is incremented by 1 and assigns the result to N". In other words, if for example N is 111000 and N' is 101011, the attacker changes N to N" = 111010.
  2. N" ⊕ N' = H(R‖T‖Nt) ⊕ M ⊕ H(R‖T‖Nt') ⊕ M'
  3. (N" + 1) ⊕ N' = (H(R‖T‖Nt) ⊕ M + 1) ⊕ H(R‖T‖Nt') ⊕ M'. Note that M + 1 = M'.
  4. (N" + 1) ⊕ N' = H(R‖T‖Nt) ⊕ H(R‖T‖Nt')
  5. ((N" + 1) ⊕ N') ⊕ N = (H(R‖T‖Nt) ⊕ H(R‖T‖Nt')) ⊕ H(R‖T‖Nt) ⊕ M
  6. ((N" + 1) ⊕ N') ⊕ N = H(R‖T‖Nt') ⊕ M
- The attacker impersonates the reader and sends H(R‖T‖Nt') ⊕ M and the obtained G, i.e. G = H(T‖R‖M) to the tag.
- The tag calculates H(R‖T‖Nt') XORed with the received H(R‖T‖Nt') ⊕ M to obtain M, then calculates and verifies H(T‖R‖M); if successful, the tag authenticates the attacker not the legitimate reader.

3.2 Location Tracking (Privacy Analysis)

The researchers have proposed a number of privacy models to evaluate the privacy of the RFID protocols. In this paper, we will focus on a well-known and frequently cited privacy model proposed in [12]. The authors developed

an untraceable privacy (UPriv) model that demonstrates how the attacker can trace a tag's location. Their model is summarised as follows: An adversary (A) controls the communication channel between a tag (T) and a reader (R) by interacting either passively or actively with them. The adversary can run the following queries:

- Execute (R, T, i) query: The adversary can passively eavesdrop on a session (i) and obtain access to the exchanged messages between R and T.
- Send (V, U, m, i) query: The adversary can perform active attacks by impersonating an entity such as U ∈ T and sends a message (m) to entity V ∈ R during session (i). Also, the adversary can alter or block the exchanged message(s).
- Corrupt (T, K) query: The attacker can physically access the tag's memory T and read the tag's secret value (K).
- Test (T, i) query: When this query is invoked for session (i), depending on a randomly chosen bit b ∈ {0, 1}, A is given T_b from the set $\{T_0, T_1\}$. A succeeds if it can guess the bit b.

Untraceable privacy (UPriv) is defined as a game (g) played by the adversary (A) and the reader and tag instances. The game consists of three phases:

1. Learning phase: A can send the Execute, Send, and Corrupt queries to any random T_0 and T_1 tags.
2. Challenge phase: A is given a tag $T_b \in \{T_0, T_1\}$, and sends any Execute, Send and Corrupt queries to T_b.
3. Guess phase: A terminates the game and outputs a bit b', which is its guess of the value of b. The success of A in winning g and thus breaking the notion of UPriv is quantified in terms of A's advantage in distinguishing whether A received T_0 or T_1, i.e. it correctly guesses b.

We evaluated the privacy of the cloud-based RFID authentication protocol using this model, and we found that the adversary can correlate two sessions to the same tag, thus allowing the tag's location tracking as shown below:

1. Learning phase: A impersonates a reader R and sends the Send (R, T_0, Nr, i + 1) query in the session (i + 1) by sending Nr to tag T_0 and thus obtains H(R ‖ T_0 ‖ Nr), Nt_{T_0}.
2. Challenge phase: A chooses two fresh tags T_0, T_1 to be tested and sends a Test (i + 1, T_0, T_1) query. Depending on a randomly chosen bit b ∈ {0, 1}, A is given a tag T_b from the set $\{T_0, T_1\}$. A sends the Send (R, T_b, Nr, i + 1) query in the session (i + 1) by sending the same random number (Nr) to tag T_b and thus obtains H(R ‖ T_b ‖ Nr), Nt_{T_b}.
3. Guess phase: A terminates the game if:
   $T_b = T_0$, then H(R ‖ T_b ‖ Nr) = H(R ‖ T_0 ‖ Nr)
   Hence, it is highly likely that this is the same tag where the adversary had initially eavesdropped, i.e. $T_b = T_0$. Else $T_b = T_1$.
   Therefore, the attacker successfully correlates two sessions to the same tag, and thus allowing unauthorised tracing.

3.3 Analysing Data Secrecy in the Cloud-Based RFID Authentication Protocol Using CasperFDR

We used CasperFDR to formally analyse the cloud-based RFID authentication protocol between the reader and the tag. CasperFDR is a compiler developed by Gavin et al. [13], which takes a high-level description of the protocol and analyses the protocol description against the stated specification (goals) to show whether the protocol meets the main requirements. CasperFDR has been used to model communication and security protocols and verify the authentication and secrecy requirements of the protocol, which are the main goals of the Xie et al. protocol. CasperFDR has confirmed its capability to find vulnerabilities in many protocols such as in [14–16].

We prepared a CasperFDR script. In the script, we assume that the reader knows about the tag's data. The communication between the reader and server is secure, therefore we did not check the protocol in this area. In the #Free variables Section, the reader (R) and tag (T) are defined as Agent; the random numbers Nr and Nt are defined as Nonce; and TID (tag identifier), RID (reader identifier), S, and M are defined as Data.

#Free variables
T, R: Agent
Nr: Nonce
Nt: nonce
TID, RID, S, M: Data
h: HashFunction
InverseKeys = (h, h)

As mentioned in Sect. 2, the main goals of the cloud-based RFID authentication protocol are authenticating the reader to the tag, and vice versa, and verifying that the data, such as R, T, S and M remain secret between the reader and tag. These goals are shown in the script in the #Specification Section, where data secrecy is depicted as Secret, such as Secret(R, M, [T]), which means that M should be a secret between R and T. The goal predicate authentication takes the form of Agreement, such as Agreement(T, R, [TID, RID]), which means that the tag is authenticated to the reader, and both parties agreed on the data values TID and RID.

#Specification
Agreement(T, R, [TID, RID])
Secret(R, M, [T])
Secret (T, S, [R])
Secret (T, TID, [R])
Secret(T, RID, [R])

In addition, in the #Intruder information Section, the intruder is defined to be Mallory, who can take full control of the session; he can impersonate any entity in the protocol, read the messages transmitted in the network, intercept, analyse, and/or modify messages.

After compiling the CasperFDR script and feeding the output to the verifier tool called Failure-Divergence Refinement (FDR), a man-in-the-middle attack is found. It states that the attacker will recover the value of M, which should be a secret value. We used the Casper compiler to analyse the attack found by FDR. The attack is illustrated below:

1. $T \rightarrow Mallory : H(R \parallel T \parallel S)$
2. $Mallory \rightarrow R : H(R \parallel T \parallel S)$
3. $R \rightarrow Mallory : Nr$
4. $Mallory \rightarrow T : Nr$
5. $T \rightarrow Mallory : H(R\|T\|Nr), Nt$
6. $Mallory \rightarrow R : H(R\|T\|Nr), Nr$
7. $R \rightarrow Mallory : N = H(R\|T\|Nr) \oplus M,\ G = H(T\|R\|M)$

The attacker performs a man-in-the-middle attack and eavesdrops on a session between the tag and the reader. The attacker impersonates the reader to obtain the tag's messages and then impersonates the tag to send the tag's messages to the reader. Casper shows that the attacker can replace the tag's random number (Nt) with the reader's generated random number (Nr), and at the end of the protocol run the attacker can calculate $M = H(R\|T\|Nr) \oplus N$ and obtain M.

As a result, in Sect. 4, we propose a new improved cloud-based RFID authentication protocol.

4 The Improved Cloud-Based RFID Authentication Protocol

In this section, we explain the proposed protocol in detail.

4.1 Goals of the Proposed Protocol

The proposed protocol aims to protect the tag's data from being revealed by any entity except the legitimate reader. The privacy of the reader is beyond the scope of the paper. The proposed protocol should meet the following goals:

- Mutual authentication: The protocol should provide mutual entity authentication, where the communication should take place between valid tag, reader and cloud server. The readers should authenticate the cloud server before making any further process. At the end of the protocol, the tag should receive a message from the reader that confirms the legitimacy of the reader.
- Tag data anonymity: The RFID tag should support a mechanism for concealing the tag's data from any entity except the legitimate readers.
- Untraceability: If the data being sent from the tag to the reader is static or linked to data sent previously, the tag holder's location could be tracked. Therefore, the RFID tag's responses should be anonymous and unlinkable in order to prevent such an attack.

- Resistance to replay attacks: An adversary can eavesdrop on the communication between the reader and tag, reuse the data and send it repeatedly. Therefore, the generated messages should be fresh to the protocol session.
- Resistance to desynchronisation attacks: An adversary can eavesdrop on the communication between the reader and tag, modify the flow of data and prevent messages from reaching their target. Therefore, the server should store the old and new values of the tag in order to authenticate the tag and reach synchronisation even if the attacker blocks the exchanged messages.
- Resistance to impersonation attacks: An attacker can respond to a reader query and can claim that this response is coming from a legitimate tag, and this fabrication enables the attacker to masquerade as a legitimate tag. Similarly, an attacker may impersonate the legitimate reader and attempt to obtain access to the tag's data. Hence, to prevent such attacks, the tag's data should be protected during transmission.

4.2 Assumptions

We present an improved cloud-based RFID authentication protocol, which operates under the following assumptions:

- The reader contacts the tag through a wireless channel that is susceptible to attacks.
- The communication channel between the reader and the cloud server is secure.
- There are multiple readers in the system, so a tag can be read in many different locations.
- This scheme only supports readers that are tamper-resistant, for example, they have a secure memory and a rigid access control mechanism.
- The tag's data are stored in non-volatile memory, such as EEPROM or Flash memory, where they can be updated.
- All the operations in the tag are atomic, i.e. either all of the operations or none are processed. Although this operation might be expensive to implement, it will defend the tags if the attacker kills the electromagnetic field between the reader and the tag or simply the tag moves from the reader's signal.
- The cloud server is not trusted, it might reveal the tag's data to intruders.

4.3 Protocol Behaviour

The main protocol features are discussed below:

- Tags are capable of computing XOR, generating a pseudo-random number and calculating hash functions.
- The reader can compute XOR, generate a pseudo-random number, calculate hash functions and perform symmetric operations.
- The proposed protocol uses random numbers in an attempt to prevent location tracking and replay attacks.
- After a successful authentication between the cloud server and tag, both parties update their values to be used in the next transaction.

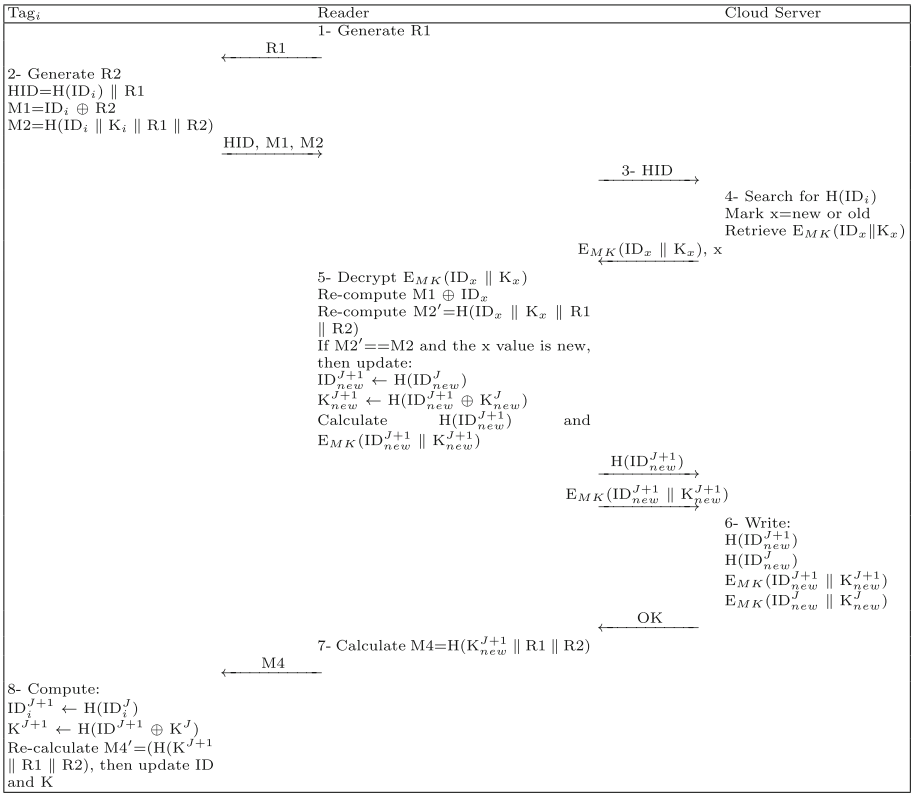
- The cloud server does not store the tag's ID and tag's secret key; it stores the hash of the tag's ID and the encryption of the tag's ID and tag's secret key to provide confidentiality and anonymity to the tag's data.
- The cloud server stores both the old and the new tag's data in order to prevent desynchronization attacks.
- Additional tag memory is necessary in our protocol to store a list of random numbers received from previous queries, which can be done by adding extended on-chip non-volatile memory on the RFID tags.
- Each legitimate reader contains a master key used for a symmetric encryption.
- The reader does not store any data related to the tags.
- The system operator is responsible for managing the system. The system operator encodes the RFID tag's data into the tags and the cloud server, and assigns the master key to all the readers it manages during the initialisation phase.

4.4 Notation

The notation used in the proposed protocol are presented below:

1. The notation related to the tag is:
   - ID_i: The i^{th} tag's ID
   - K_i: The i^{th} tag's secret key
2. The notation related to the reader is:
   - r_n: The n^{th} reader in the system
   - MK: The master key shared by all the legitimate readers used for a symmetric encryption and decryption
3. The notation related to the cloud server is:
   - $H(ID_{new})$: A unique hash of the updated ID
   - $E_{MK}(ID_{new} \parallel K_{new})$: The updated data ($ID_{new}$, K_{new}) associated with the i^{th} tag encrypted with the master key
   - $H(ID_{old})$: A unique hash of the old ID
   - $E_{MK}(ID_{old} \parallel K_{old})$: The old data ($ID_{old}$, K_{old}) associated with the i^{th} tag encrypted with the master key
4. Other notation used in the proposed protocol is:
   - x: The value kept as either new or old to show whether the tag uses the old or new values of ID and K
   - R1: A pseudo-random number generated by the reader
   - R2: A pseudo-random number generated by the tag and serving as a temporary secret for the tag
   - $E_k(M)$: A message M encrypted with a key
   - $A \leftarrow B$: The value of A is updated to that of B
   - J: The transaction number
   - i: The number of the tag in the system

Table 1. The proposed improved cloud-based RFID mutual authentication protocol (successful run)



4.5 Protocol Description

The scheme consists of two phases, namely initialisation and authentication.

Initialisation Phase. We assume that the initialisation phase is carried out via a secure channel in a secure environment. The initialisation process is summarised below:

1. For each tag the system operator manages, the system operator assigns a unique H(ID$_i$), which serves as an index, and E$_{MK}$(ID$_i$ ‖ K$_i$) in the cloud server.
2. For the i$_{th}$ tag, the system operator assigns ID$_i$ and K$_i$ in the ith tag.
3. Initially, H(ID$_{old}$), and E$_{MK}$(ID$_{old}$ ‖ K$_{old}$) in the cloud server are set to null.
4. The system operator assigns the master key in each reader the system manages.

Authentication Phase. The authentication process is shown in Table 1 and is described as follows:

1. Reader: The reader starts the session by generating a random number R1 and sending it to the tag.
2. Tag$_i$: The tag performs the following:
   - Generates R2 as a temporary secret for this session.
   - Computes the following messages:
     $\mathrm{HID} = \mathrm{H}(\mathrm{ID}_i) \parallel \mathrm{R1}$, which serves as an index message
     $\mathrm{M1} = \mathrm{ID}_i \oplus \mathrm{R2}$
     $\mathrm{M2} = \mathrm{H}(\mathrm{ID}_i \parallel \mathrm{K}_i \parallel \mathrm{R1} \parallel \mathrm{R2})$, which serves as an authentication message
   - Sends HID, M1 and M2 to the reader.
3. Reader: The reader sends HID to the cloud server.
4. Cloud server: The cloud server performs the following:
   - For all the stored $\mathrm{H}(\mathrm{ID}_{new})$ and $\mathrm{H}(\mathrm{ID}_{old})$, it searches for $\mathrm{H}(\mathrm{ID}_i)$ until there is a match. Marks x = new or old based on the matched $\mathrm{H}(\mathrm{ID}_i)$.
   - Retrieves the associated data, i.e. $\mathrm{E}_{MK}(\mathrm{ID}_x \parallel \mathrm{K}_x)$.
   - Sends $\mathrm{E}_{MK}(\mathrm{ID}_x \parallel \mathrm{K}_x)$ and x to the reader.
5. Reader: The reader performs the following:
   - Decrypts $\mathrm{E}_{MK}(\mathrm{ID}_x \parallel \mathrm{K}_x)$ using the master key, and obtains ID_x and K_x.
   - Re-computes $\mathrm{M1} \oplus \mathrm{ID}_x$ to obtain R2.
   - Re-computes $\mathrm{M2'} = \mathrm{H}(\mathrm{ID}_x \parallel \mathrm{K}_x \parallel \mathrm{R1} \parallel \mathrm{R2})$. If there is a match, the reader authenticates the tag. Furthermore, the reader confirms that the data within the server's message is correct and authenticates the cloud server.
     If $\mathrm{M2'} == \mathrm{M2}$ and the received x value is new, this implies that the tag's data are synchronised with the server's data, then the reader updates (ID and K) to be used in the next transaction $(\mathrm{J}+1)$:
     $\mathrm{ID}_{new}^{J+1} \leftarrow \mathrm{H}(\mathrm{ID}_{new}^{J})$
     $\mathrm{K}_{new}^{J+1} \leftarrow \mathrm{H}(\mathrm{ID}_{new}^{J+1} \oplus \mathrm{K}_{new}^{J})$
   - Calculates $\mathrm{H}(\mathrm{ID}_{new}^{J+1})$, and encrypts the new values, i.e. $\mathrm{E}_{MK}(\mathrm{ID}_{new}^{J+1} \parallel \mathrm{K}_{new}^{J+1})$ using the master key.
   - Notifies the server to update their values by sending:
     $\mathrm{H}(\mathrm{ID}_{new}^{J+1})$, and $\mathrm{E}_{MK}(\mathrm{ID}_{new}^{J+1} \parallel \mathrm{K}_{new}^{J+1})$
6. Cloud server: The cloud server performs the following:
   - Writes the following data in its database:
     $\mathrm{H}(\mathrm{ID}_{new}) \leftarrow \mathrm{H}(\mathrm{ID}_{new}^{J+1})$
     $\mathrm{E}_{MK}(\mathrm{ID}_{new} \| \mathrm{K}_{new}) \leftarrow \mathrm{E}_{MK}(\mathrm{ID}_{new}^{J+1} \| \mathrm{K}_{new}^{J+1})$
     $\mathrm{H}(\mathrm{ID}_{old}) \leftarrow \mathrm{H}(\mathrm{ID}_{new}^{J})$
     $\mathrm{E}_{MK}(\mathrm{ID}_{old} \| \mathrm{K}_{old}) \leftarrow \mathrm{E}_{MK}(\mathrm{ID}_{new}^{J} \| \mathrm{K}_{new}^{J})$
     Hence, the cloud server writes the new values in the new data fields ($\mathrm{H}(\mathrm{ID}_{new})$, $\mathrm{E}_{MK}(\mathrm{ID}_{new} \| \mathrm{K}_{new})$), and writes the old entries in the old data fields ($\mathrm{H}(\mathrm{ID}_{old})$, $\mathrm{E}_{MK}(\mathrm{ID}_{old} \| \mathrm{K}_{old})$).
   - Sends an OK message to notify the reader that the update process has been successful.
7. Reader: The reader performs the following:
   - If the reader received the OK message from the cloud server, the reader notifies the tag to update its data such as (ID_i, K_i) by calculating $\mathrm{M4} = \mathrm{H}(\mathrm{K}_{new}^{J+1} \parallel \mathrm{R1} \parallel \mathrm{R2})$ using the updated tag's secret key.

- Sends M4 to the tag.

8. Tag_i: The tag performs the following:
   - Computes $ID_i^{J+1} \leftarrow H(ID_i^J)$ and $K_i^{J+1} \leftarrow H(ID_i^{J+1} \oplus K_i^J)$.
   - Re-calculates $M4' = (H(K_i^{J+1} \parallel R1 \parallel R2)$, and if it is equal to the received value of M4, then it authenticates the reader and updates ID and K to:
     $ID_i^{J+1} \leftarrow H(ID_i^J)$
     $K_i^{J+1} \leftarrow H(ID_i^{J+1} \oplus K_i^J)$

If M2' == M2 and the received value of x is old, the reader still authenticates the tag but this implies that the tag's data has been desynchronised, thus the reader does not update the current values of ID and K. It sends no update to the server and sends $M4 = H(K_{new}^J \parallel R1 \parallel R2)$ to the tag using the current value of the tag's secret key. Then, the tag re-computes M4 using the current values. If there is a match with the received M4, the tag authenticates the reader and updates its data, as shown in the previous step.

If there is no match with the received M4 using the current or new values of ID_i and K_i, then the tag does not authenticate the reader and will not update its data.

5 Protocol Analysis

5.1 Informal Analysis of the Improved Cloud-Based RFID Authentication Protocol

Our proposed protocol meets the following requirements:

- Mutual authentication: If the reader successfully calculates the tag's responses M1 and M2, it authenticates the tag, as only the legitimate tag can calculate such responses. Similarly, if the tag calculates M4 and it finds a match with the received M4, it confirms that the reader is legitimate and authenticates it. Furthermore, the reader decrypts the server's message $(E_{MK}(ID_x \parallel K_x))$, and if the tag's messages M1 and M2 are authenticated, this means that the cloud server sends legitimate data within the message, and hence the reader authenticates the server.
- Tag data anonymity: The tag stores two values, namely ID and K that are supposed to be secret and that are not revealed to any entity except the legitimate readers. The tag's data are not sent in the clear, as they are protected using fresh random numbers (HID, M1 and M2) and sent within the hash function (HID and M2). Therefore, only the legitimate reader can extract these values. Furthermore, if the cloud server is a malicious entity, this will not affect the tag's data privacy, as the cloud server stores the hash of the tag's ID and the encrypted tag's data; and without the master key, the cloud server cannot disclose the tag's data.
- Tag location privacy (untraceability): In the proposed protocol, the tag's responses are changed in each session using the updated tag's values and fresh random numbers (R1 and R2), and thus the attacker will obtain new

responses every time he eavesdrops on a session. Moreover, if the previous authentication session failed, the tag's responses will change due to the existence of new fresh random numbers.

However, in case that the tag did not update its values due to desynchronisation attacks or any other means of interruption, the attacker can track the location of the tag as the tag always replies with H(ID), which remains the same until the tag interacts with a legitimate reader and updates its data.

- Resistance to replay attacks: The proposed protocol utilises a challenge-response scheme, in which each party maintains a set of random numbers that it encountered from a previous protocol run in order to avoid repeated random numbers. Therefore, if the tag received an already used random number it ignores the query.
- Resistance to desynchronisation attacks: In the proposed protocol, the desynchronisation attack is avoided by storing the previous values of the tag's data in the cloud server, and hence it achieves synchronisation. For instance, if the attacker blocks M4 more than once, the tag will not update its data. In the next session, the reader contacts the desynchronised tag and sends the tag's HID message to the cloud server, then the cloud server will find a match with the tag's old data and send $E_{MK}(ID_{old} \parallel K_{old})$ and x = old, as HID matches $(H(ID_{old}))$; and thus synchronisation is still achieved.
- Resistance to tag and reader impersonation attack: To impersonate the tag, the attacker must be able to compute a valid response (HID, M1 and M2) to a reader query. However, it is hard to compute such responses without knowledge of ID and K. Similarly, the attacker needs to be in possession of ID, K and R2 to impersonate the legitimate reader and send M4.
- Compromising the reader: The only risk that the system may encounter is compromising the reader; hence the attacker will gain the master key. However, in the Assumption Sect. 4.2, we assumed that the proposed protocol only supports readers that are tamper-resistant, for example, they have a secure memory and a rigid access control mechanism.

Table 2 shows how our proposed protocol provides more security and privacy features than the cloud-based RFID authentication protocol. Based on the discovered weaknesses, we found that the cloud-based RFID authentication protocol is vulnerable to reader impersonation attacks; hence mutual authentication is not achieved. Moreover, we used CasperFDR to analyse the cloud-based RFID authentication protocol, and it showed that an attacker can discover the secret value (M); hence the tag data anonymity is compromised. In addition, we used a privacy model to analyse the tag location privacy in the cloud-based RFID authentication protocol, and we found that the attacker can successfully correlate two sessions to the same tag, which allows unauthorised tracing.

5.2 Performance Analysis

In this section, we conduct a comparative analysis of the performance cost regarding storage cost, and communication cost.

Table 2. Comparison between the cloud-based RFID authentication protocol and our proposed protocol

RFID system main requirements	Cloud-based protocol	Our proposed protocol
Tag data anonymity	No	Yes
Tag location privacy	No	Yes under assumption that the tag has updated its values
Resistance to replay attack	Yes	Yes
Resistance to desynchronisation	Yes	Yes
Resistance to tag impersonation	Yes	Yes
Resistance to reader impersonation	No	Yes
Mutual authentication	No	Yes

- Storage cost: Due to the limitation of tag memory, the tag should store a minimum amount of data. In the proposed protocol, the tag stores two values in a rewritable flash memory namely (ID, K), as they change in different authentication sessions, each of which has a length of 224 bits. Since the tag's memory can store 1 Kilobyte of data, in our protocol the tag securely stores $224 * 2 = 448$ bits in the memory. Additional tag memory is necessary in our protocol to store a list of random numbers received from previous queries, for example by adding extended on-chip non-volatile memory on the RFID tags.
- Communication cost: In the proposed protocol, the tag sends three messages (HID, M1 and M2) in order to be successfully authenticated. A total of 896 bits are sent over the channel. Hence, it provides a relatively low communication cost.

5.3 Analysing Data Secrecy and Authentication in the Proposed Protocol Using CasperFDR

To formally analyse the proposed protocol and confirm that the secrecy and authenticity between the reader and tag are achieved, we used CasperFDR [13].

We prepared a CasperFDR script. In the script, we assume that the reader knows about the tag's data. The communication between the reader and server is secure, therefore we did not check the protocol in this area. In the #Free variables Section, the tag (T) and reader (R) are defined as Agent; the random numbers R1 and R2 are defined as Nonce; and ID (tag identifier), K (tag key) are defined as Data.

```
#Free variables
T, R: Agent
R1: Nonce
R2: nonce
ID, K: Data
h: HashFunction
InverseKeys = (h, h)
```

As mentioned in Sect. 4.1, the main goals of our protocol are authenticating the reader to the tag, and vice versa, as well as verifying that the data, such as ID, K and R2 are secure. These goals are shown in the script in the #Specification Section as shown below:

#Specification
Agreement(T, R, [ID, K, R2])
Secret (T, ID, [R])
Secret (T, K, [R])
Secret(T, R2, [R])

In addition, in the #Intruder information Section, the intruder is defined as Mallory, who can take full control of the session; he can impersonate any entity in the protocol, read the messages transmitted in the network, intercept, analyse and/or modify messages.

After compiling the CasperFDR script and feeding the output to FDR, CasperFDR did not find any feasible attack, which means that mutual authentication is achieved successfully between the reader and the tag, and the tag's data are protected and transfered securely.

5.4 Privacy Analysis

We deployed the same privacy model described in [12] to evaluate the privacy of the proposed protocol. We found that the adversary cannot invade the privacy of the tag and trace its location as shown below:

- Learning phase: The adversary eavesdrops on a valid session between R and T_0. He sends the Execute command and then maintains the following values:
  R1, $HID = H(ID_0) \parallel R1$
  $M1 = ID_0 \oplus R2$
  $M2 = H(ID_0 \parallel K_0 \parallel R1 \parallel R2)$
- Challenge phase: A is given a tag $T_b \in \{T_0, T_1\}$ randomly. He starts a new session with T_b by impersonating the reader, sends R1 to T_b within the Send query and terminates the session. T_b responses can be:
  $HID = H(ID_b) \parallel R1$
  $M1 = ID_b \oplus R2$
  $M2 = H(ID_b \parallel K_b \parallel R1 \parallel R2)$

  A will not be able to guess the correct tag (bit b), as the received messages M1 and M2 contain a random number (R2) generated by the tag, which changes in every session and is not known to the adversary. Moreover, regarding the HID message, if the tag encounters a repeated random number, such as R1, it will terminate the session.

6 Conclusion

In this paper, we examined the cloud-based RFID authentication protocol, and we found that the protocol is prone to reader impersonation attacks, and location

tracking. Moreover, we formally analysed the cloud-based RFID authentication protocol using CasperFDR, which demonstrated that the secrecy of the protocol is not protected as the attacker can perform a man-in-the-middle attack and obtain the secret data. Therefore, we proposed an improved cloud-based RFID authentication protocol that is based on the strengths in the cloud-based RFID authentication protocol while avoiding its security and privacy issues. The proposed protocol has been analysed informally, and we showed that it is more immune to reader impersonation attack and location tracking attacks than the Xie et al. protocol. Furthermore, it can resist other forms of attacks, such as reply attacks, desynchronisation attacks, and impersonation attacks. Also, we illustrated that the tag data anonymity is preserved, and hence the cloud server and the attackers will not be able to obtain the tag's data. In addition, the communication session between the reader and the tag was formally analysed using CasperFDR, and it did not find any feasible attack. Finally, we showed that the proposed protocol can comply with the required tag's memory storage cost and communication cost.

Acknowledgment. Sarah Abughazalah is supported by the Ministry of Higher Education and King Khaled University in Saudi Arabia.

References

1. Finkenzeller, K.: RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification, 2nd edn. John Wiley & Sons Inc, New York (2003)
2. Juels, A.: RFID security and privacy: a research survey. IEEE J. Sel. Areas Commun. **24**(2), 381–394 (2006)
3. Chien, H., Chen, C.: Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards. Comput. Stand. Interfaces **29**(2), 254–259 (2007)
4. Song, B., Mitchell, C.: RFID authentication protocol for low-cost tags. In: Proceedings of the First ACM Conference on Wireless Network Security, pp. 140–147. ACM (2008)
5. Tan, C., Sheng, B., Li, Q.: Secure and serverless RFID authentication and search protocols. IEEE Trans. Wirel. Commun. **7**(4), 1400–1407 (2008)
6. Poulopoulos, G., Markantonakis, K., Mayes, K.: A secure and efficient mutual authentication protocol for low-cost RFID systems. In: International Conference on Availability, Reliability and Security, ARES 2009, pp. 706–711. IEEE (2009)
7. Li, J., Wang, Y., Jiao, B., Xu, Y.: An authentication protocol for secure and efficient RFID communication. In: 2010 International Conference on Logistics Systems and Intelligent Management, vol. 3, pp. 1648–1651. IEEE (2010)
8. Chun, J., Hwang, J., Lee, D.: RFID tag search protocol preserving privacy of mobile reader holders. IEICE Electron. Express **8**(2), 50–56 (2011)
9. Yoon, E.: Improvement of the securing RFID systems conforming to EPC Class 1 Generation 2 standard. Expert Syst. Appl. **39**(1), 1589–1594 (2012)
10. Lee, C.F., Chien, H.Y., Laih, C.S.: Server-less RFID authentication and searching protocol with enhanced security. Int. J. Commun. Syst. **25**(3), 376–385 (2012)
11. Xie, W., Xie, L., Zhang, C., Zhang, Q., Tang, C.: Cloud-based RFID authentication. In: 2013 IEEE International Conference on RFID (RFID), pp. 168–175. IEEE (2013)

12. Ouafi, K., Phan, R.C.-W.: Privacy of recent RFID authentication protocols. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 263–277. Springer, Heidelberg (2008)
13. Lowe, G.: Casper: a compiler for the analysis of security protocols. In: Proceedings of 10th IEEE Computer Security Foundations Workshop, pp. 18–30 (1997)
14. Alshehri, A., Briffa, J.A., Schneider, S., Wesemeyer, S.: Formal security analysis of NFC m-coupon protocols using casper/fdr. In: 2013 5th International Workshop on Near Field Communication (NFC), pp. 1–6. IEEE (2013)
15. Aiash, M., Mapp, G., Phan, R.W., Lasebae, A., Loo, J.: A formally verified device authentication protocol using Casper/FDR. In: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1293–1298. IEEE (2012)
16. Kumari, V.V., Raju, K.K.: Formal verification of IEEE 802.11w authentication protocol. Procedia Technology **6**, 716–722 (2012). 2nd International Conference on Communication, Computing and Security [ICCCS-2012]