EPC Gen2v2 RFID Standard Authentication and Ownership Management Protocol

Haifeng Niu, Eyad Taqieddin, Member, IEEE, and S. Jagannathan, Senior Member, IEEE

Abstract—Providing security in passive RFID systems has gained significant attention due to their widespread use. Research has focused on providing both location and data privacy through mutual authentication between the readers and tags. In such systems, each party is responsible of verifying the identity of the other party with whom it is communicating. For such a task to succeed, the tags and readers are initialized with shared secret information which is updated after a successful authentication session. Ownership management, which includes transfer and delegation, builds upon mutual authentication. Here, the use of security in RFID is extended to encompass the more practical case where a tagged item is shifted from one owner to another. As such, we propose a new authentication and ownership management protocol that is compliant with the EPC Class-1 Generation-2 Version 2 standard. The protocol is formally analyzed and successfully implemented on hardware. The implementation shows that the use of such protocol adds security with little added overhead in terms of communication and computation.

Index Terms—authentication, ownership transfer, RFID, security

I. INTRODUCTION

Radio Frequency Identification (RFID) systems are deployed in numerous automated asset management applications. Examples of such applications include libraries, warehouses, and border control to name a few. In a RFID system, the identification information of the tracked objects is stored in a nonvolatile memory on passive tags. These tags are queried by readers which transmit an RF signal to energize the tags so as to get the backscattered information. The readers are connected to backend servers which store and process the data.

An important aspect to be considered in RFID systems is the data and location privacy. Given that the communication between the tags and readers is wireless, various attacks may be launched by an unauthorized user to either collect information about the tagged items or cause a disruption of the system operation. As a result, the communicating parties, a tag and a reader, must authenticate each other before any data exchange. Moreover, the data should be concealed from unauthorized access through encryption. As such, both the reader and the tag need to share secret information.

• Haifeng Niu and Sarangapani Jagannathan are with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65401. E-mail: [hnhy6, sarangap]@mst.edu.
• Eyad Taqieddin is with the Department of Network Engineering and Security, Jordan University of Science and Technology, Irbid, Jordan 22110. E-mail: eyadtqa@just.edu.jo
• Research supported in part by Intelligent Systems Center.

Besides authentication, ownership management (i.e., transfer or delegation) (OT) is also an important aspect of RFID security as most tagged items will change owners at least once during their lifetime. For example, the ownership of the tagged item is transferred from the manufacture to the retailer, and then to the customer. Special attention to the security must be paid because this process is relatively vulnerable to attacks due to the exchange of secret keys or passwords. Further, it is desired that the ownership management protocol would protect the privacy of the new owner from tracking by the previous owner(s) and to guarantee that the new owner will not be able to retrieve the previous secret keys used by the old owner.

To add security features to the passive tags, the EPC Class-1 Generation-2 standard (EPC Gen2v1) [1] introduces the access and kill password. The access password is used whenever the reader wishes to read/write data in a tag’s memory. On the other hand, the kill password along with the kill command is issued to stop the tag from responding to any subsequent queries. These basic security mechanisms are easily defeated because the passwords are XORed with a random number that is sent in plaintext, which can easily be retrieved.

Recently, the EPC Class-1 Generation-2 standard version 2 (EPC Gen2v2) [2], has been ratified. Backward-compatible with the old version, the new one provides a series of features intended to improve security of the tag by allowing the manufacturers to customize the cryptographic authentication methods to verify identity and provenance, as well as avoid unauthorized access. Similar to the previous standard, EPC Gen2v2 supports the use of a pseudo random number generator (PRNG), a cyclic redundancy check (CRC) function, and XOR operation.

A security protocol is usually considered as “EPC compliant” if it solely uses one or more of these functions. However, these functions by themselves are not cryptographic functions. Other measures should be taken to provide an acceptable level of security considering their computational capabilities since there are only 500 – 5000 gate elements on the tag, of which 200 – 2000 can be used for security-related functions [3]. The Advanced Encryption Standard (AES), for example, requires about 3000 gate elements to be implemented. Hash functions like MD5 and SHA-256 require even more gate elements, 8000 – 10000 [4]. Therefore, securing information among RFID devices is a major challenge due to the limited storage and computational capabilities on the passive tags.

A. Related Work

A comprehensive survey [5] examines several aspects
related to RFID security. Mainly, the importance of mutual authentication and secret information sharing is emphasized. In [6], a classification of RFID authentication protocols, based on the cryptographic/logical functions, is presented. These protocols range from full-fledged protocols in which symmetric, asymmetric, and hash functions are supported [7]-[12] to the least computationally demanding class called the ultra-lightweight, where basic bitwise logical and shift operations are employed [13]-[16].

In [17], an EPC compliant mutual authentication protocol based on CRC exchange followed by update on secret information after each authentication session is proposed to provide privacy, anonymity, and to resist replay and denial of service (DoS) attacks. However, [18] and [19] indicated that [17] did not achieve its intended goals. The work of [18] detailed the steps to successfully impersonate a valid tag either temporarily or permanently and how to run a DoS attack. These attacks are shown to be practical due to the short length of the data units exchanged. In [19], the impersonation attack is extended to include the back-end database as well as the tags. The analysis shows how the location of the tag can be identified and tracked.

The authors in [20] proposed a new protocol called Azumi to overcome the security flaws of [21] and claim that it is capable of defending against location tracking, DoS attacks, counterfeit reader or tag, and man-in-the-middle (MitM) attacks. However, it is shown that the work in [20] is vulnerable to tag impersonation and secret disclosure attacks. An enhanced version Azumi+ was proposed in [22] as a solution.

Several research efforts considered the problem of ownership management. One of the earliest ownership transfer protocols appeared in [23]. However, the old owner privacy cannot be guaranteed due to the way the shared keys are updated, leading to a de-synchronization attack. Around the same time, the authors in [24] proposed a scalable, delegated pseudonym protocol enabling ownership transfer. However, as pointed out in [25], the keys shared by several tags become a weakness that reduces security.

In [26], a protocol based on the use of hash functions, symmetric cryptography, and the XOR operation is proposed. The protocol is shown to be vulnerable to tracking and DoS attacks by manipulating the value of the random number sent to the tag [27]. Moreover, in [28], an attacker can add noise to the final message exchange resulting in the tag holding incorrect secret information due to which any subsequent authentication would fail.

Another protocol appeared in [29] referred to as product-flow ownership-transfer protocol (POP). This protocol supports querying, disabling, or updating the secret keys on the tag. However, this protocol does not provide privacy to the new owner because the old owner will still be able to access the tag by exploiting his knowledge of the shared secret keys. In addition, it is prone to de-synchronization attacks similar to [30], [31].

As for ownership delegation protocols, for example, the work in [30] assumes that the channel from the tag to the reader is secure and that any ownership transfer/delegation will be securely accomplished. This is an impractical assumption and cannot be relied upon. Another variant of [26] was proposed in [33] as an ownership delegation protocol. Delegation is possible because the message containing the new key uses the old key as a variable. As such, the old owner will be able to keep track of the key updates and modify its keys accordingly.

The ownership management protocols mentioned above [23]-[30], as well as in [32]-[35], are not EPC compliant due to the nature of the cryptographic functions used in computing the messages. An EPC compliant lightweight protocol is given in [36] wherein PRNG and XOR functions are used on the tag side. However, the protocol is sensitive to replay and MitM attacks. Another EPC compliant ownership transfer protocol is proposed in [37] where the authors add a modular division operation to the functions of the tag because such a function would not require a large number of gate elements. However, a potential attacker can disguise as an owner who can update the secret keys in the same way as the new owner does, thus eliminating the security.

The other ownership transfer protocols [37]-[41] conforming to EPC standards use CRC as the encryption method and cannot guarantee security because of the complete linearity property of CRC. In fact, as analyzed in [19] and [39][42], the attacker is able to trace, impersonate and eventually disclose all the information stored in tags with very few interactions. In summary, an EPC compliant secure authentication and owner management protocol is yet to be developed for passive tags.

B. Contributions

In this paper, a lightweight mutual authentication and ownership management protocol is proposed. The protocol is compliant with the EPC Gen2v2 standard. The basic supported operations, along with permutation, are used as basic operations to provide the cryptographic functionality.

The protocol is designed to fit within the computational abilities of the tag as well as the scarce energy resources. The details of the protocol are given along with formal security proof of its correctness. Further, the protocol is implemented by using EPC compliant tags and is shown to add minimal overhead to the standard message exchanges.

This paper is an extended version of work published in [43]. We extend our previous work by making the following improvements. 1) In addition to the basic ownership transfer scheme introduced in [43], the protocol presented in this work also supports ownership delegation. 2) A mathematical proof of both authentication and secrecy with strand space theory is provided. 3) A detailed description on how the proposed protocol is implemented in hardware is offered. 4) More experiments are conducted to evaluate the performance of the proposed protocol, such as time consumption analysis for multiple-tag ownership transfer and resistance evaluation to the brutal force attack.

The main contributions of this work include: 1) the development of a novel lightweight authentication and ownership transfer protocol for passive RFID systems by taking into account both delegation and ownership transfer into consideration, 2) the demonstration of how the proposed
protocol is compliant with the EPC Gen2v2 standard, 3) the security analysis of the protocol by using strand space, and 4) hardware implementation and evaluation, which, to the best knowledge of the authors, is the first hardware based evaluation for ownership transfer protocols.

The rest of the paper is organized as follows. In Section II, the detailed description of proposed protocol is given followed by the security analysis given in Section III and a comparison with pervious work in Section IV. The hardware implementation and evaluation is given in Section V. The paper is concluded in Section VI.

II. PROPOSED PROTOCOL

In addition to the limited functions supported by the EPC standard, the available power on the tag for various computations and transmissions needed as part of the security protocol implementation is an important constraint. Moreover, the limited available time for executing the steps for the authentication and ownership management protocol is an added challenge. Finally, the protocol has to be implemented in a practical setting in which hundreds or thousands of tags are present with several tags simultaneously performing exchange and this should be completed within the allowed timeslot.

To enhance the functionality of the protocol, the ultra-lightweight permutation operation (Per) [16] is added to the existing functions on the tag. This operation offers diffusion of the bits and helps overcome any problem occurring because of the nature of bitwise operations. The operation is defined as follows:

Definition 1 [16]: For two n-bit strings, X and Y , in the form

\[ X = x_1 x_2 \cdots x_n, \quad x_i \in \{0,1\}, \quad i = 1,2,\ldots,n \]

\[ Y = y_1 y_2 \cdots y_n, \quad y_i \in \{0,1\}, \quad i = 1,2,\ldots,n \]

The Hamming weight of Y , \( w(Y) \), is \( m (0 \leq m \leq n) \) and

\[ y_1 = y_2 = \cdots = y_m = 1, \quad y_{m+1} = y_{m+2} = \cdots = y_n = 0, \]

where

\[ 1 \leq k_1 < k_2 < \cdots < k_m \leq n, \quad 1 \leq k_{m+1} < k_{m+2} < \cdots < k_n \leq n \]

Then, the permutation of X according to Y , denoted as Per\((X,Y)\) , is given by

\[ \text{Per}(X,Y) = x_{k_1} x_{k_2} \cdots x_{k_m} x_{k_{m+1}} \cdots x_{k_n} x_{k_{n+1}} \]

\[ (1) \]

The following assumptions are made in designing the protocol:

1) The link between the readers is secure. Also, the link between any reader and the trusted third party (TTP) is assumed to be secure. This is a reasonable and quite common assumption as the readers are built with more powerful processors which can take advantage of complex encryption algorithms to guarantee secure data transmission.

2) The link between the tag and any other entity is considered insecure.

3) The current owner and the tag share a secret key that is only known to them.

A. Initialization

The tag is initialized with the following values:

a) \( K \): secret key shared with both current and new owners, as well as delegates, if any.

b) \( K_M \): master key only shared with the tag owner. A reader with \( K_M \) is able to modify key \( K \), but a reader with key \( K \) does not have access to \( K_M \).

c) \( K_{TP} \): key shared between the tag and the TTP.

d) EPC: electronic product code, the static identifier of a tag.

e) \( R_{ID} \): The ID of the reader \( i \) currently owning the tag.

f) \( IDS \): In the protocol, index pseudonym (IDS) is exchanged instead of using the tag identifier (ID). The IDS is a pointer to a database entry in which the information of the tag is stored. Such an entry may include the identifying information and the keys related to that tag. We use the IDS instead of concealing the EPC in the messages, for the following two reasons: 1) The EPC value is constant and its use in multiple runs of the protocol may reveal information about the tag and its secret values. 2) Tracking the EPC by the old owner is possible.

Note that for compliance with the EPC standard, all data units in the protocol are 96 bits long. For the convenience of implementation, these 96-bit data are broken into six 16-bit words. For example, a 96-bit parameter \( A \) is broken into six words, denoted as \( A(1), A(2), \ldots A(i), \ldots A(6) \), where \( A(i) \) is the \( i \)-th 16-bit subunit. As a result, all the computations are executed six times in order to get the complete 96-bit data.

The current owner is initialized with \( K, K_M, IDS, R_{ID} \) and EPC. As mentioned earlier, the proposed ownership management protocol takes both delegation (details in Section C) and complete ownership transfer (Section D) into consideration. However, it is important to notice that before either delegation or complete ownership transfer take place, mutual authentication is needed to verify the authority of all parties involved.

B. Phase I: Mutual authentication

<table>
<thead>
<tr>
<th>Old Owner</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secret</td>
<td>Secret</td>
</tr>
<tr>
<td>K, K_M, EPC, R_{ID}</td>
<td>K, K_M, EPC, R_{ID}, IDS</td>
</tr>
</tbody>
</table>

Generate random number \( rnd_1, rnd_2 \), calculate \( A, B, C \):

\[ A(i) = rnd_1(i) \oplus \text{PRNG}(K(i) \oplus \text{IDS}^*(i)), \quad i = 1 \text{ to } 6 \]

\[ D \]

Calculate \( D' \):

\[ D'(i) = \text{PRNG}(K^*(i) \oplus \text{IDS}^*(i)) \quad i = 1 \text{ to } 6 \]

If \( D = D' \), tag is authorized.

Fig. 1 Mutual authentication and keys update [43].

A general scenario for an authentication session starts with the reader querying a tag. In response, the tag sends an index pseudonym (IDS). A sequence of exchanges follows such that
the reader securely sends random numbers to the tag by using the shared key, the tag authenticates the reader and vice versa, and the keys and IDS are updated. The transactions that take place are shown in Fig. 1.

The purpose of the authentication phase is to: 1) prove the possession of shared secret key to each other without disclosing it; 2) pass the nonces that are used to update the keys. To achieve this, the reader generates two 96-bit random values \((rnd_1, rnd_2)\) as the nonces, then computes \(A\), \(B\), and \(C\) in a way described in Fig. 1. Particularly, in the computation of \(A\) and \(B\), the secret key is part of the input of PRNG function so that the key is protected while the tag can verify the readers’ possession of the key by doing the same computation. Furthermore, message \(C\) is used to check if the tag has retrieved the correct nonces \((rnd_1, rnd_2)\) from messages \(A\) and \(B\). It is very important to note that the PRNG is a nonlinear function, meaning that if an attacker flips one bit of \(R_{ID2}\), the tag will get a totally different (and incorrect) \(rnd_1\). Moreover, since \(rnd_1\) is used to retrieve \(rnd_2\) from \(B\), therefore \(rnd_2\) derived by the tag will be incorrect. As a result, even if the attacker flips the same bit of \(B\), it will not get \(C^*\) that equals to \(C\).

If \(C\) equals to \(C^*\), then it is believed that the reader does have the secret key and the tag has retrieved \(rnd_1\) and \(rnd_2\) successfully. Then the new key and IDS are computed in a way specified in Fig. 1. Similarly, we use message \(D\) to: 1) prove the tags’ possession of the secret key; 2) inform the reader that the tag has computed the new key and IDS.

Upon receiving message \(D\), the reader will compute \(D^*\) in the same manner and determines whether \(D\) equals to \(D^*\) or not. If that is true, then the tag is authenticated. Consequently, the reader and the tag update the new computed key and IDS for future uses. It should be noted, however, that both the reader and the tag should maintain a copy of the old key and IDS to avoid desynchronization problems (more explanation can be found in Section IV).

### C. Phase II, Case 1: Delegation

At this point, \(R_{ID1}\) is ready to delegate its rights over the tag to \(R_{ID2}\). For that purpose, we introduce the use of the ticket. This is used by the delegate reader to prove to the tag that it is a valid reader and that it had received sufficient credentials from the current owner to allow it to access the tag.

In the proposed delegation protocol, both \(R_{ID1}\) and the tag compute the ticket as shown in Fig. 2.

![Fig. 2 Ticket computation on an old owner and tag.](image)

\(R_{ID1}\) uses a secure link with \(R_{ID2}\) and passes to it the EPC, IDS, \(K\), and ticket. A valid ticket allows \(R_{ID2}\) to query the tag and to run mutual authentication sessions with it. Fig. 3 shows how the delegate \(R_{ID2}\) uses the ticket to query and update the tag. Note that the ticket value becomes an integral part of the message computations.

In the case of delegation, \(R_{ID1}\) may wish to restore its sole ownership of the tag. This means that it has to revoke the ticket such that \(R_{ID2}\) will not pass the test of equality between \(B\) and \(B^*\). When that happens, no update will take place and the tag will not run further session with the revoked reader. The proposed approach for this is to modify the value of \(K_M\) such that the ticket given to \(R_{ID2}\) will not match with the computed value. Note that the value of the ticket is updated with every session because the values of \(rnd_1\) and \(rnd_2\) are changed.

Delegation is suitable for those cases where certain “guest readers” need to access the tag temporarily. In other cases, however, the old owner needs to give up the ownership completely and transfer it to the new owner. This process is presented in the following section.

### D. Phase II, Case 2: Complete ownership transfer

In this case, we propose the use of a TTP to guarantee the correctness of the protocol. The need for the TTP arises from the fact that the old owner holds the same values shared between the new owner and the tag. This means that any update taking place by \(R_{ID2}\) may be mirrored by \(R_{ID1}\). This violates an important property of ownership transfer which is backward privacy.

However, it is worth to note that the EPC Gen2v2 standard introduces a new “untraceable” command, which allows the tag to reduce its operating range for all readers. This function, to some degree, may give a practical solution of releasing the use of TTP by reducing the operating range so that only the new owner can reach the tag. As a result, the old owner cannot repeat the key update process and thus the backward privacy is guaranteed.

In this protocol, the goal is to change the value of \(K_M\) stored on the tag such that it matches that stored on \(R_{ID2}\). After that, \(R_{ID1}\) will have no access to the tag anymore. This proposed approach adds an extra functionality that we may use the reverse process in case we wish to satisfy the ownership repossessing property. As presented in Fig. 4, the outline of the protocol includes those steps:

1. TTP generates a random number \(rnd\) and uses it to update \(K_M\) to \(K_M^*\). This will become the new master key shared between the tag and the new owner, \(R_{ID2}\).
2) TTP sends $K_m^*$ to $R_{ID2}$ using the secure channel.

3) The challenging part for the TTP becomes to send $K_m^*$ to the tag. For that, we propose the use of messages $A$ and $B$ shown in Fig. 4. Similar to what we have done in the authentication phase, the secret key is set as the input of the nonlinear PRNG function while the nonce is XORed with the PRNG output so that the key will not be disclosed and the nonce can be passed to the tag safely. Message $B$ is used for the tag to verify TTP's possession of the secret key and to check the correctness of the nonce.

4) The tag retrieves $md_1$ from $A$ and verifies that $B$ is equal to $B'$.

5) The value of $md_1$ is used by the tag to update $K_m^*$ in a manner similar to that used by the TTP.

6) The new owner and tag need to challenge each other to verify that both have the same value of $K_m^*$.

The reader verifies the value of $D$ by comparing with $D'$. If they match then the tag and reader both have successfully authenticated each other and updated their secret key and $IDS$ values.

The ownership transfer phase follows a similar manner. Messages $A$ and $B$ are used between the TTP and the tag to convey a random number and to prove to the tag that the messages originate from an authentic source, the TTP. To illustrate, assume that the TTP generates $rnd_1$ as $md_1 = 0x411895D3C7772A68D368159E$

Then $A$ and $B$ will be $A = 0x330B34429236E6B83E1BD20C$ $B = 0x4DC01DE6C69F8F68E8E025D0$

The tag retrieves $md_1$ and then updates $K_m^*$ to $K_m^* = 0x232F1EBB84FED34E175A0797$

The same key is already in the possession of the new owner through the secure channel with the TTP. Thus, the tag and the new owner can communicate with each securely using the new key. Note that the old owner will not be able to compute $K_m^*$ since it does not have the $md_1$ value.

III. SECURITY ANALYSIS

In this part, we use the strand space analysis to prove the correctness of the protocol. A strand is a sequence of events that a single principal may engage in, while a strand space is a set of strands [44]. Here, “principal” stands for any participant that may be involved in the protocol such as old/new owner, tag, attacker, or TTP [44]. In the following analysis, we use some of the definitions and lemmas provided in [44]. We analyze the security of the authentication phase only. The analysis of the other two phases is either part of or identical to that of the authentication phase.

Let $T_{name}$ be the set of names such as $R_{ID1}, R_{ID2},$ and $IDS$. $Key_x$ is the set of keys known by the principal $x$. Let $m$ be a message and $K$ is a key, then we represent the encryption of message $m$ using $K$ as $[m]_K$. Also, $K^{-1}$ is the corresponding decryption key of $K$. Now, for simplicity, we rewrite messages $A,B,C$ and $D$:

$$A = md_1 \oplus [K, R_{ID1}]_{K_{a1}} \oplus [K, R_{ID2}]_{K_{a2}}$$

$$B = md_1 \oplus [K, md_1]_{K_{a1}}$$

$$C = \{md_1, R_{ID1}\}_{K_{a2}} \oplus \{md_2, R_{ID2}\}_{K_{a3}}$$

$$D = [K', IDS']_{K_{a1}}$$

where $K_{a1}, K_{a2}, K_{a3}, K_{b1}^{-1}, K_{b2}^{-1}, K_{c1}^{-1}, K_{c2}^{-1}$, and $K_{d}^{-1}$ are unknown to all the principals because of the one-way property of PRNG function. We can show that under the following Assumption 1, this presentation is equivalent to the original one in Fig. 1 in the sense of security.

**Assumption 1** If $y = PRNG(x)$ and $y$ is known to a principal $P$, then the probability that $P$ is able to compute the value of $x$ is negligible.

According to EPC Gen2v2 standards [2], the PRNG function...
shall meet the following randomness criteria:

- The probability \( P \) that any RN16 has value \( RN16 = j \), for any \( j \), should be bounded by \( 0.8/2^{16} < P < 1.25/2^{16} \).
- For a tag population of up to 10,000 tags, the probability that any two or more tags simultaneously generate the same sequence of RN16s shall be less than 0.1%.
- An RN16 drawn from a Tag’s PRNG 10ms after the rise time shall not be predictable with a probability greater than 0.025% if the outcomes of prior draws from the PRNG, performed under identical conditions, are known.

In our protocol, the 96-bit random number consisting of six 16-bit random numbers is used which means that the probabilities defined in the above criteria are much smaller (new probability \( P' \) equals to \( P^6 \), not just \( 6P \)). Therefore, this assumption is reasonable. Taking the computation of message \( A \) as an example, we can conclude that even if a penetrator managed to get the value of both \( PRNG(K \oplus R_{i12}) \) and \( PRNG(K \oplus R_{i22}) \) (in fact he can only know the XOR results of them), by Assumption 1, he still cannot compute the value of \( K, R_{i12} \) and \( R_{i22} \).

Next, we will introduce the definition of the proposed ownership transfer strand space \( S_{ot} \).

**Definition 2** An infiltrated strand space \( \Sigma, P \) is an \( S_{ot} \) space if it is the union of three kinds of strands:

1. Penetrator strands \( s \in P \), the set of keys known by \( P \) is \( Key_P \);
2. “Initiator strands” \( s \in Init[K, rnd1, rnd2, R_{i12}, R_{i22}] \) with trace:
   \[ <+[R_{i12}, R_{i22}],-\{IDS\},+[ABC],-D> \]
   where \( A, B, C, \) and \( D \) are defined in (1) to (4) and the sign “\(+\)” means sending out a message while “\(-\)” means receiving. The principal associated with this strand is the old owner. We will use \( Init[\cdot] \) to denote the set of all the strands shown above. The set of keys known by \( Init \) is \( Key_{I} \).
3. Complementary “responder strands” \( s \in Resp[K, rnd1, rnd2, R_{i12}, R_{i22}] \) with trace:
   \[ <-[R_{i12}, R_{i22}],+\{IDS\},-[ABC],+D> \]
   The principal associated with this strand is the tag. Similarly we will use \( Resp[\cdot] \) to denote the set of all the strands shown above. The set of keys known by \( Resp \) is \( Key_{R} \).

Fig. 5 shows the strand space representation of the proposed ownership transfer protocol. In the next two parts, we prove the two aspects of correctness respectively: authentication and secrecy.

**A. Authentication**

In [45], G. Lowe introduces four reasonable meanings of the word “authentication.” They are, from the weakest to the strongest, aliveness, weak agreement, Non-injective agreement and agreement. In this paper, we prove that the proposed protocol satisfies the strongest definition: agreement.

**Definition 3** (Agreement [45]) A protocol guarantees to an initiator \( A \) agreement with a responder \( B \) on a set of data items if, whenever \( A \) completes a run of the protocol, apparently with \( B \), which apparently has previously been running the protocol with \( A \) as a responder. If the two agents agreed on the data values corresponding to all the variables in the data items, and each such run of \( A \) corresponds to a unique run of \( B \).

It should be noticed that this definition only guarantees to an initiator agreement with a responder. To complete the proof of the authentication, it is also necessary to prove that the protocol guarantees to a responder agreement with an initiator. We will start with the proof of the latter one. Additionally, since the first two data exchanges \( \{R_{i12}, R_{i22}, IDS\} \) are broadcasted in the form of cleartext and do not contain any secrets, we will not include them in the following analysis.
that $K_{c1}^{-1}$ and $K_{c2}^{-1}$ are unknown, we have $rnd1, rnd2$, and $K \subset (ABC)$ and thus

$$rnd1, rnd2, K \subset \text{term}(n')$$  (5)

Now if we can show “$K \subset \text{term}(n)$” then we are able to conclude that $n \in \text{Init}[]$. This is because

i) $K \not\in \text{Key}_{p}$ which implies that $n \not\in P$;

ii) Although $K \in \text{Key}_{g}$, $rnd1$ and $rnd2$ do not originate from $\text{Resp}[]$ according to (5). Based on the definition of node $n$, it follows that $n \not\in \text{Resp}[]$.

Therefore, the problem becomes to prove $K \subset \text{term}(n)$.

Now we assume $K \not\subset \text{term}(n)$; from (5) we know $K \subset \text{term}(n')$, then there exists at least one node $n'$ that proceeds $n'$ from which $K$ uniquely originates and hence $K \subset \text{term}(n')$. Since $K \not\in \text{Key}_{p}$, it follows that $n'$ lies either in the responder’s or the initiator’s strand. However, according to the definition of $S_{or}$ strand space, the form of $K$ is either $\{ K, R_{id1} \}_{K_{s}} \oplus \{ K, R_{id2} \}_{K_{s}}$ or $\oplus \{ K, rnd1 \}_{K_{s}}$ where $rnd1$ and $rnd2$ are fresh. In other words, $rnd1$ and $rnd2$ also originate from $n'$, which contradicts with the fact that $rnd1$ and $rnd2$ originate from $n$. Therefore, we have $K \subset \text{term}(n)$ and hence $n \in \text{Init}[]$.

Moreover, “$rnd1$ and $rnd2$ originate from $n$” also gives the conclusion that the sign of $\text{term}(n)$ is positive (Lemma 2.8 in [44]). Together with $n \in \text{Init}[]$ and the structure of $S_{or}$, we can get that $\text{term}(n) = +\{ABC\}$.

**Lemma 2** Upon receiving $D$ if the node $n$ is able to update $K$ and $IDS$, then $n$ belongs to $\text{Init}[]$ and $n_1$ (defined in Lemma 1) proceeds node $n$. In addition, we designate this particular node $n$ as $n_1$.

Proof: The proof of this lemma is almost identical to the proof for Lemma 2. Basically we will show that $\{ K', IDS' \} \subset \text{term}(n)$ in the form of cleartext. Then it follows that $K \subset \text{term}(n)$ and thus we eliminate the case that $n \not\in P$. Again since the sign of $\text{term}(n)$ is positive, together with the form of $S_{or}$ we are able to conclude that $n$ belongs to $\text{Resp}[]$.

**Lemma 3** Let $n$ be the node in which $D$ originates from in $\Sigma$, If $D = D'$ for the node $n_2$ (defined in Lemma 2), then $n$ belongs to $\text{Resp}[]$. In addition, we designate this particular node $n$ as $n_3$.

Proof: The proof of this lemma is almost identical to the proof for Lemma 2. Basically we will show that $\{ K', IDS' \} \subset \text{term}(n)$ in the form of cleartext. Then it follows that $K \subset \text{term}(n)$. Thus we eliminate the case that $n \not\in P$. Again since the sign of $\text{term}(n)$ is positive, together with the form of $S_{or}$ we are able to conclude that $n$ belongs to $\text{Resp}[]$.

**Lemma 4** There exists a unique node $n$ in $\text{Resp}[]$ proceeding $n_{1}$, such that $\text{term}(n) = -(ABC)$, where $ABC$ is given in Lemma 1. In addition, we designate this particular node $n$ as $n_1$.

Proof: In Lemma 3 we have shown that $\{ rnd1, rnd2, K \} \subset \text{term}(n_{1})$. Let $n$ be the $\omega_{\text{minimal}}$ member of node $n_{3}$ in $\text{Resp}[]$. Then by the definition of $\omega_{\text{minimal}}$ [44], we have $\{ rnd1, rnd2, K \} \subset \text{term}(n)$. Since $rnd1$ and $rnd2$ uniquely originate in $\Sigma$ from node $n_{1}$ which is proven in Lemma 1, then we have this relationship

$$n_{1} \longrightarrow \{ rnd1, rnd2, K \} \longrightarrow \cdots \forall n$$  (6)

Therefore the sign of $\text{term}(n)$ is negative. Given that $n \in \text{Resp}[]$, exploring all the forms of responder strands, we
have \( \text{term}(n) = \{ABC\} \). Since \( \{ABC\} \) is computed directly based on \( \text{rnd}1 \) and \( \text{rnd}2 \), it follows that \( \{ABC\} \) also originates uniquely from node \( n_1 \). Hence \( \{ABC\} \) in \( \text{term}(n) \) is the same term that originated from \( n_1 \).

Proposition 2 follows directly from Lemma 3 and 4. And together with Proposition 1, we have completed the proof of authentication.

B. Secrecy

Definition 5 (Secrecy [46]) A message \( m \) is considered secret if in every bundle of the protocol the penetrator cannot receive \( m \) in clear text. In other words, there exists no node \( n \) such that \( \text{term}(n) = m \).

Proof of secrecy for the proposed protocol is straightforward because of special treatment with the secret key \( K \). From (1) to (4) we can see that, in all messages, every sub-term containing \( K \) is in the form of \( \{K^\cdot\} \). where \( K \) belongs to \( \{K_{A1}^{-1}, K_{A2}^{-1}, K_{B1}^{-1}, K_{D1}^{-1}\} \) and is unknown to all principals. Therefore, under Assumption 1, we can guarantee the secrecy of \( K \).

IV. COMPARISON WITH RELATED PROTOCOLS

The previous section confirmed the correctness of the protocol. Given the proven authentication process and secrecy of data, the protocol is guaranteed to resist the tag impersonation, reader impersonation, replay, and MitM attacks. Such resistance of attacks is an essential requirement in authentication and ownership management protocols.

However, there are several other distinctive requirements for any authentication and ownership management protocol. These requirements include forward and backward privacy, desynchronization and windowing avoidance, and location privacy. To perform a comparison between the proposed ownership management protocol and the previous work, we give an analysis of the protocol in terms of these requirements.

- **Backward privacy**: An important aspect to consider with ownership transfer is the privacy of the new owner. The old owner should not be able to update the secret keys in order to have copies of the keys of the new owner. In the proposed protocol, the use of TTP guarantees that only the new owner can update the keys. The access of the old owner is permanently revoked upon ownership transfer.

- **Forward privacy**: Similarly, the new owner of the tag should not be able to deduce the keys that were used by the old owner. If such a case arises, then all previous transactions can be decrypted, which violates the privacy of the old owner. In the proposed protocol, the key update operations depend on the PRNG function which is irreversible. This guarantees that the no message exchanges prior to the ownership transfer would be decrypted.

- **Desynchronization avoidance**: The desynchronization problem cannot be completely prevented because the adversary can always choose to block the last confirmation message and consequently one party updates the keys while the other one does not. Our solution is that the TTP should always keep a copy of the previous secret keys and the corresponding tag IDS in case of confronting desynchronization attacks. In that case, the new owner will not be able to authenticate the tag and then TTP should attempt to resend the key update message until the ownership transfer succeeds.

- **Windowing avoidance**: The windowing problem occurs when the new and old owner share possession of the same keys within the same timeslot. Both parties would have access to the tag and problems may arise if, for example, the ownership transfer is interrupted. In such a case, both parties would have access to the tag and can act as its owners. In the proposed protocol, the old owner and the new owner never possess the master key at the same time.

- **Location privacy**: Instead of using the unique and life-long static identifier EPC, the proposed protocol uses IDS which is updated after every successful authentication. As a result, the adversary cannot identify the location of the target tag.

A comparison with previous related work is shown in Table I, where a “Y” means the scheme satisfies the requirement while an “N” indicates the opposite. From the table it can be concluded that among the non-EPC-compliant protocols, Kapoor’s [27] has the best performance but it still suffers from the windowing problem and is not suitable for low-cost RFID tags due to the use of hash functions. On the other hand, the existing EPC compliant protocols either fail to provide backward privacy or are vulnerable to replay attack because of using CRC as the encryption method. In contrast, the proposed protocol not only conforms to the EPC standards, but also satisfies the security requirements. Furthermore, our protocol also supports delegation, which is desirable in many scenarios where temporal ownership sharing is needed.

V. HARDWARE IMPLEMENTATION AND EVALUATION

In this section, the proposed authentication and ownership management protocol is implemented and evaluated in hardware. Since the new EPC Gen2v2 protocol was ratified very recently, there is no reader available in the market supporting the new standard yet. Our solution is to use a Gen2v1 RFID tag and emulate the Gen2v2-only commands (“Authenticate”, “KeyUpdate”) by using the “BlockWrite” and “Read” commands. Note that “BlockWrite” command allows the reader to send as long as 256 words of data to the tag and therefore is capable of emulating the above Gen2v2-only commands. As these commands take similar amounts of bits,
theoretically the differences in terms of processing time and energy consumption are negligible.

A. Implementation Details

The mutual authentication and OT is executed through the use of command/response set defined by the EPC Gen2v2 standard as shown in Fig. 7. The current (old) owner sends “select” and “query” command (and “QueryAdjust”, “QueryRep” commands, if necessary) in order to identify the target tag from a large population of tags. As a result, the target tag replies with a new 16-bit random number RN16 and transfers its state from “ready” to “reply”. Note that before identifying the target tag, a probabilistic collision management method is adopted as specified in the standards while after identifying the target tag, RN16 works as a kind of session ID indicating a specified tag to avoid collision. Then the reader issues an ACK command containing the same RN16 and the tag replies with its IDS and other information, which can be found in the EPC Gen2v2 standard specifications. Upon receiving the “Req_RN” command with the correct RN16 and access key, the tag backscatters the new RN16 and enters the “open” state.

![Fig. 7 Mutual authentication under EPC Gen2v2 standard](image)

Next, we make use of the “Authenticate” and “KeyUpdate” commands, which are newly introduced in EPC Gen2v2, to complete the mutual authentication phase. As specified in [2], the “Authenticate” command should contain fields listed in Table II. In particular, we define the contents of “message” field in the “Authenticate” command as described in Table III. The “command ID” is used to indicate that this command will send the necessary security parameters (RID, A, B…) and start the authentication phase.

![Table II: AUTHENTICATE COMMAND](image)

![Table III: “MESSAGE” FIELD IN “AUTHENTICATE” COMMAND](image)

The value D is contained in the response message of the “Authenticate” command, as described in Table IV. A non-zero value in the “status” field indicates that the tag has retrieved the nonces and computed the new key and IDS.

Upon receiving a response with the “status” of success, the reader will compute D' in the same manner of computing D. If D equals to D', then the tag is authenticated. Consequently, the reader issues a “KeyUpdate” command to the tag for confirmation. As a result, the tag commits to the newly computed key and IDS for future uses.

![Table IV: RESPONSE MESSAGE OF THE “AUTHENTICATE” COMMAND](image)

The implementation details of the delegation phase and the ownership transfer phase under EPC Gen2v2 framework are omitted as it is similar to what we have presented in the authentication phase.

A common RFID platform presented in [44] is chosen to implement and analyze the proposed protocol. Operating in the UHF frequency range, this platform is designed based on the Wireless Identification and Sensing Platform (WISP), developed by Intel Research Seattle [44]. Similar to the WISP tags, the program running in the modified WISP tags is also written strictly conforming to the EPC Gen2v1 standard [1]. Therefore, the tag can communicate with most of the off-the-shelf UHF RFID readers.

On the modified WISP tag, shown in Fig. 8, a “bow tie” antenna and a four-order Dickson charging pump are adopted to convert the RF signal to DC power to support the whole on-board circuitry. The 16-bit microprocessor MSP430F2132 has an ultra-low power consumption (only 600µA at 1.8V and 4MHz). It can execute an instruction in as little as 0.25µs. Further, the 1Mbit EEPROM 24AA1026 embedded only on the modified WISP tags ensures enough space for storing the data such as secret keys. Therefore, these features including its ability of re-programming, relatively strong computation capacity, and large memory space make it a decent platform to evaluate customized protocols. In fact, the WISP tag was utilized to demonstrate the feasibility or performance of security protocols [49], [50].

![Fig. 8 Modified WISP: Class-1 Generation-2 UHF passive RFID tag platform](image)

Fig. 9 shows the software structure of our experimental platform. On the reader side, the protocol is implemented with Java in Eclipse, above the “Reader library” and “LLRP [51]” layers. On the tag side, we implement the proposed protocol in a higher level in order to stay compliant with the EPC standards. The IAR Workbench for MSP430 is used for debugging and downloading the program. The reader used in the experiments is Impinj Speedway Revolution R220, with...
transmission power set to 30dBm with a receiving sensitivity set to -70dBm.

![Diagram](Fig 9. Software structure of the evaluation platform.)

**B. Ownership transfer operation time, with sufficient energy**

First, it is of interest to measure the execution time for a complete ownership transfer process when there is sufficient energy on the tag. To do this, the tag is placed as close as 0.5m away from the reader antenna to ensure it can harvest enough energy. Note that no matter how complicated one protocol is, it can be broken into steps that belong to one of the four categories: a) computation on tags, b) computation on readers (here consider TTP as a reader), c) data exchange between tags and readers (\(T \leftrightarrow R\)), d) data exchange between two different readers (\(R \leftrightarrow R\)).

In our case, both the computation on readers and data exchange between two different readers can be negligible. From the results presented in Table V, it can be seen that the total time of on-tag computation plus the data exchange between the reader and the tag is \(T_{\text{tag}} = 146.14\text{ms}\), which is quite close to the actual measured total time \(T_{\text{total}} = 167.28\text{ms}\). In fact, the time spent for the on-tag computation is only 16.66ms (49980 instruction cycles @ 3MHz) for the authentication and ownership transfer phase and 10.83ms for the delegation phase (32490 instruction cycles @ 3MHz), which confirms the ultra-lightweight property of the proposed protocol.

![Table](TABLE V. MEASURED TIME AND INSTRUCTION CYCLES)

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
<th>Value</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_{\text{TR}})</td>
<td>Number of (T \leftrightarrow R) rounds</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>(T_{\text{TR}})</td>
<td>Time for each (T \leftrightarrow R) round</td>
<td>43.16ms</td>
<td>-</td>
</tr>
<tr>
<td>(T_{\text{auth}})</td>
<td>Time of computation on tag during authentication phase</td>
<td>12.39ms</td>
<td>37170</td>
</tr>
<tr>
<td>(T_{\text{tran}})</td>
<td>Time of computation on tag during ownership transfer phase</td>
<td>4.27ms</td>
<td>12810</td>
</tr>
<tr>
<td>(T_{\text{tag}})</td>
<td>Total time of on-tag computation</td>
<td>(T_{\text{tag}} = N_{\text{TR}} \times T_{\text{TR}} + T_{\text{auth}} + T_{\text{tran}})</td>
<td>146.14ms</td>
</tr>
<tr>
<td>(T_{\text{total}})</td>
<td>Actual measured total time</td>
<td>167.28ms</td>
<td>-</td>
</tr>
</tbody>
</table>

**C. Ownership transfer operation time, with insufficient energy**

Since passive RFID tags are powered by the RF signal emitted by the reader antenna, the energy being harvested decreases when moving away from the antenna. It is also of interest to measure the execution time when there is insufficient energy. For this purpose, the tag is placed at different distances away from the reader antenna and the corresponding number of successful ownership transfer sessions per minute is taken. In contrast, the experiments are repeated using the same tag running the protocol, with all the computations eliminated. In other words, the control group only executes instructions to perform the same number of data exchanges.

From Fig. 10, it can be seen that when the distance is within 1m, the number of successful ownership transfer session per minute is almost constant because enough energy has been harvested within this short distance. As the distance increases, the number of successful ownership transfer sessions goes down due to failure of data exchange when there is insufficient power. As a consequence, the reader will either start over a new ownership transfer session or request for a retransmission, which both consume longer time. When the distance is larger than 3m, the proposed protocol with or without retransmission can only be executed for a very limited number of sessions due to the lack of energy. However, the most important conclusion is that, if one compares the two curves with each other, the number of sessions executed per minute for the proposed protocol with computation is only slightly less than that the one without computation, which shows that the on-tag computation involved due to the proposed protocol is insignificant.

![Figure](Fig. 10 Number of successful ownership transfer sessions per minute.)

**D. Ownership transfer of multiple tags, with sufficient energy**

In most applications, there may be more than one tag whose ownerships should be transferred. The previous analysis focused on a single tag ownership transfer. However, it is of interest to investigate the performance when multiple tags are exchanged, given that collisions or interference may happen. In this experiment, we place the tags at a distance of 0.5 m from the reader antenna. The maximum number of tags in this test is 13 in order to ensure that each tag receives sufficient energy from the reader antenna. We initially start with one tag and add more tags until we reach the maximum of 13 tags. For each set of present tags, we measure the number of successful ownership transfer sessions. For comparative purposes, we also examine the performance with and without the cryptographic computations of the protocol. The results are shown in Fig. 11.

From Fig. 11, it is evident that as the number of tags increases, the number of successful ownership transfer session decreases. This drop is due to the added extra time caused by the reader when it isolates one specific tag from all the tag population. However, the drop is not that significant in terms of performance. For example, in the case of the protocol without the computations the drop in the successful sessions is about 5.6% and for that with the computation the reduction reaches
values ranges from 36 to as high as 67. As a result, the attacker cannot determine the presence of the tag by analyzing the values of IDS. Next the reader impersonation aspect is considered.

F. Reader Impersonation

In this scenario, the attacker impersonates an owner attempting to deceive the tag to believe that the attacker is authentic. Assuming the IDS of the tag has been disclosed, the attacker generates two random numbers ($r_{da}$, $r_{db}$), guesses a secret key, computes the values of $A$, $B$, and $C$, and sends them to the tag. Upon receiving $A$ and $B$, the tag retrieves $r_{da}$ and $r_{db}$ using the authentic secret key, then computes the value of $C'$. The tag is compromised if $C'$ equals to $C$. Note that the attacker does not necessarily have to possess the exact authentic key to make $C'$ equals to $C$. In some circumstances, if the protocol is not well designed, some other different values other than the authentic key could also result in $C'$ equals to $C$, this is normally referred to as a collision.

If such a scenario happens, the tag will update its secret keys and IDS although the actual owner has not initiated the session. Thus, the owner and the tag will be desynchronized. Therefore, it is of interest to examine how long it takes the attacker to compromise the tag. Note that it is unrealistic to measure the elapsed time if the length of all data units is 96 bits as it takes too long. To solve this, we truncate the data length to 16 bits, measure the elapsed time, and based on that, estimate the theoretical time for when the data units are 96-bit long. We repeat the attack 50 times and compute the average time for compromise.

The results are shown in Fig. 13, where we see when the length of all data is 16 bits; the average time to compromise the tag is 18 hours, on average. As shown before, in an authentication round, the computation takes much less time than the wireless messages exchange. Therefore, if we assume that the time for computing 96-bit long numbers ($A, B$...) is very close to that for computing 16-bit long numbers, we roughly estimate that the time for the attacker to compromise the tag through brute force when the data units are 96-bit long is $(18/2^{96}) \times (2^{16}) \approx 2.1 \times 10^{35}$ hours.
Therefore, it is safe to say that the proposed protocol is able to resist against the reader impersonation attacks.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a new EPC Gen2v2 compatible protocol by using limited cryptographic functionality was presented for mutual authentication and ownership management. This was done by employing the ultra-lightweight permutation operation and the PRNG function. Such use of a simple operation adds a minimal level of computation and energy consumption while, at the same time, supports the cryptographic goals of the protocol.

The protocol was examined both from a security point of view as well as with a hardware implementation. The analysis indicated that the transactions in the protocol do not expose the secret key information nor does the protocol depend on previously used secret keys, thus guaranteeing that replay or disclosure attacks are not possible. The comparison with previous work shows that the proposed protocol not only conforms to the EPC standards, but also satisfies the security requirements. The hardware implementation supports our initial goal of adding security to the existing EPC Gen2v2 based tags such that the system would be secure both in the case of being used by a single owner or in the more practical cases of having multiple owners during the lifetime of a tagged item.

The next steps in this work include examining the use of various ultra-lightweight or lightweight functions that would possibly fit on the very limited number of gate elements on the tag.

REFERENCES


Y. Li and J. Pang. An Inductive Approach to Strand Spaces, Formal


G. Lowe “rfidDOT: RFID delegation and ownership transfer made

strong and robust authentication enabling


Y. Li and J. Pang. An Inductive Approach to Strand Spaces, Formal


H. Niu and S. Jagannathan, “High memory passive RFID tags with multimodal sensor design and application to asset monitoring in-transit”,


Haifeng Niu received the Bachelor’s degree in Electrical Engineering from Northeastern University, Shenyang, China, in 2010. He was a master student from 2010 to 2012 in Department of Control and Engineering in Zhejiang University, Hangzhou, China. Now he is working on his Ph.D. degree in Missouri University of Science and Technology, Rolla, US. His current research interests include RFID networks, wireless networking security, and cyber-physical systems.

Eyad S. Tajiieddin received his BSc in Electrical Engineering from Jordan University of Science and Technology (JUST), Irbid, Jordan in 1999. He received his M.Sc. and Ph.D. in Computer Engineering from Missouri University of Science and Technology (formerly the University of Missouri-Rolla) in 2002 and 2007, respectively. He worked as an Assistant Professor at the department of Computer Engineering at JUST from 2007 to 2009. Currently, he is an Associate Professor at the Department of Network Engineering and Security at JUST. His research interests are in the areas of RFID Security, Wireless Multimedia Sensor Networks, Network Security, and Embedded Systems.

Jagannathan Sarangapani (referred here as S. Jagannathan) is at the Missouri University of Science and Technology (former University of Missouri-Rolla) where he is a Rutledge-Emerson Endowed Chair Professor of Electrical and Computer Engineering and Site Director for the NSF Industry/University Cooperative Research Center on Intelligent Maintenance Systems. He has coauthored 129 peer reviewed journal articles most of them in IEEE Transactions and 235 refereed IEEE conference articles, several book chapters, 3 books and holds 20 US patents. He supervised to graduation.
around 18 doctoral and 29 M.S level students and his funding is in excess of $14 million from various US federal and industrial members. His research interests include neural network control, adaptive event-triggered control, secure networked control systems, prognostics, and autonomous systems/robotics. He was the co-editor for the IET book series on control from 2010 through 2013 and now serving as the editor-in-chief for Discrete Dynamics and Society and on many editorial boards. He received many awards and has been on organizing committees of several IEEE Conferences. He is the IEEE CSS Tech Committee Chair on Intelligent Control. He is a Fellow of the Institute of Measurement and Control, UK.