Prove You Owned Me: One Step beyond RFID Tag/Mutual Authentication

Shaoying Cai, Yingjiu Li, Changshe Ma, Sherman S. M. Chow, Robert H. Deng Fellow, IEEE

Abstract—Radio Frequency Identification (RFID) is a key technology used in many applications. In the past decades, plenty of secure and privacy-preserving RFID tag/mutual authentication protocols as well as formal frameworks for evaluating them have been proposed. However, we notice that a property, namely proof of possession (PoP), has not been rigorously studied till now, despite it has significant value in many RFID applications. For example, in RFID-enabled supply chains, PoP helps prevent dishonest parties from publishing information about products/tags that they actually have never processed.

We propose the first formal framework for RFID tag/mutual authentication with PoP after correcting deficiencies of some existing RFID formal frameworks. Our framework is based on a new privacy notion—unp-privacy, and a new security notion—PoP credential unforgeability. We provide a generic construction to transform an RFID tag/mutual authentication protocol to one that supports PoP using a cryptographic hash function, a pseudorandom function (PRF) and a signature scheme. We prove that the constructed protocol is secure and privacy-preserving under our framework if all the building blocks possess desired security properties. Finally, we show an RFID mutual authentication protocol with PoP. Arming tag/mutual authentication protocols with PoP is an important step to strengthen RFID-enabled systems as it bridges the security gap between physical layer and data layer, and reduces the misuses of RFID-related data.

I. INTRODUCTION

Radio Frequency Identification (RFID) technology has greatly facilitated collection and management of identification information in a wide range of applications, from supply chain and access management to stock tracing and payments. RFID systems consist of three main components: tags, readers, and backend servers. Tags are radio transponders attached to physical objects. Readers are radio transceivers that communicate with tags to identify or authenticate them based on information stored on backend servers. RFID technology enables automatic identification and information collection due to the wireless communication property. When combined with internet and networking technology, RFID-related information can be integrated, shared, and queried in real time.

The wireless communication property of RFID is a double-edged sword. Despite enhancing efficiencies and reducing costs on manpower, it also causes RFID-enabled systems to be vulnerable to a variety of attacks. An adversary may eavesdrop, replay, and manipulate RFID communications to obtain tag identifiers, track tag locations, impersonate RFID tags and RFID readers, and trigger denial of service without tag owners’ awareness. Also, if an adversary compromises any RFID tags (e.g., via side-channel attack [1]), they may access all secret information stored on the tags.

Plenty of efforts have been devoted to securing communications between RFID readers and tags [2]. Secure and privacy-preserving tag/mutual authentication is the most fundamental functionality to protect RFID systems against various attacks. RFID tags should be identified with assurance in the presence of attacks, and meanwhile without disclosure of any valuable information. Hundreds of RFID tag/mutual authentication protocols (e.g., [3]–[16]) as well as dozens of formal frameworks (e.g., [17]–[22], [24]–[37]) for evaluating them have been proposed. However, an indispensable property, proof of possession (PoP), which was briefly discussed in [48], has not been rigorously studied till now.

PoP is highly valuable to RFID applications in which information about real-world events related to RFID tags are stored for future use and/or shared over networks. For example, access management systems require that the logs of visiting events related to authenticated access cards be kept for identifying suspicious visitors when anomalies happen. In supply chain management, visibility event data related to authenticated tags may be shared among supply chain parties through various platforms such as EPCglobal Network [38], [39] and blockchain-based product management platforms [40], [41]. These application scenarios require RFID systems to not only authenticate tags, but also prove to other parties that they indeed have authenticated the tags. Otherwise, the information about real-world events related to RFID tags may be manipulated even if the underlying tag/mutual authentication protocol works well. For example, malicious access system administrator may manipulate the logs of visiting events related to authenticated access cards, and dishonest supply chain parties may make up visibility data about certain tags/products without actually processing them.

We propose to extend tag/mutual authentication protocols to support PoP. Our major contributions are summarized below.

• We study the existing formal frameworks for RFID tag/mutual authentication protocols. We refine Deng et al.’s RFID system model [30] to allow terminations during protocol executions. We correct Yang et al.’s claim [29] about the relationship between two major RFID privacy notions, unp*-privacy and ind-privacy, and discuss the deficiencies of their privacy notion, unp*-privacy.
• We propose the first formal framework for RFID tag/mutual authentication with a new security notion, named unp\#-privacy. Unp\#-privacy can be applied for analysing any RFID reader-tag communication protocols.
• We provide a generic construction to transform an RFID tag/mutual authentication protocol to additionally support PoP by using a cryptographic hash function, a pseudo-random function (PRF), and a signature scheme. We conduct formal security analysis of our construction, and then discuss its practicability.
• We refine a secure and unp*-privacy-preserving RFID mutual authentication protocol, and then extend it to support PoP according to our generic construction. We prove that the refined protocol with PoP is secure and privacy-preserving under our framework. We also discuss its implementation.

II. RELATED WORK
A. RFID authentication protocols
The existing RFID tag/mutual authentication protocols can be classified in two categories: symmetric key-based and PKC-based. With symmetric key-based protocols, a reader and a tag conduct unidirectional or bidirectional authentication based on some shared secrets. Current symmetric key-based protocols include cyclic redundancy code (CRC) checksum-based ones (e.g., [4, 5]), one-way hash function-based ones (e.g., [6–9]), and symmetric encryption algorithms-based ones (e.g., [5]), to name a few. However, symmetric key-based tag/mutual authentication protocols inherently cannot support PoP.

Elliptic curve cryptography (ECC) is the most lightweight PKC and has been shown to be applicable in resource-constrained RFID settings [42, 43]. Researchers have proposed many ECC-based RFID authentication protocols. However, most of them are shown to be vulnerable [44], and only a few remain secure till now [10–14]. We discover that some ECC-based protocols (e.g., [12–14]) are actually symmetric key-based in terms of tag authentication, thus cannot support PoP as well. Only a couple of protocols [10, 11] can be potentially extended to support PoP, but have not been further explored yet.

B. RFID formal frameworks
Formal RFID security and privacy frameworks are fundamental to the design and analysis of robust RFID protocols. In general, an RFID tag/mutual protocol should satisfy (a) correctness, which means a valid tag reader should always be accepted; (b) security, which means an invalid tag reader should always be rejected; and (c) privacy, which means tags should not be identifiable or traced by unauthorized entities. Till now, many formal RFID frameworks have been proposed (e.g., [17–22, 24–37]).

Correctness and security definitions in existing RFID formal frameworks appear to be, to a large extent, equivalent. Among them, Deng et al.’s [30, 32] is considered more elaborate than others [19]. It is full of subtleties in developing rigorous and precise privacy notions. Dozens of other privacy notions have been proposed, and are systematically discussed in [33, 35–37]. Below we briefly introduce typical RFID privacy notions.

a) Indistinguishability-based privacy notion: Intuitively, it requires that any adversary cannot distinguish two uncorrupted tags [17, 18] or two groups of tags [19]. It is easy to apply for proving the privacy of protocols which are built with ind-secure primitives, such as an IND-CCA secure encryption scheme.

b) Unpredictability-based privacy notion: Intuitively, it requires that any adversary cannot distinguish protocol messages from random strings. The unpredictability-based privacy notions are easy to apply for proving the privacy of symmetric key-based protocols, which form the majority of existing RFID protocols. We will discuss more on unpredictability-based privacy notions [25–29] in Section III.

c) Vaudenay’s privacy notion [20]: Intuitively, it requires that for any adversary, there exists a blinded adversary such that the advantage of the adversary towin the privacy game over the blinded one’s is negligible, where the blinded adversary does not ‘use’ the communication captured during the protocol run in order to determine its output. Vaudenay defined the most comprehensive adversary types. There are following works [21–23] to consolidate adversary type, extend the definitions to address mutual authentication, and etc.

d) Zero-knowledge-based privacy notion: Intuitively, it requires that whatever information an adversary can obtain from interacting with a target tag, there exists a simulator who can provide indistinguishable similar information without interacting with the tag. Zk-privacy was proposed by Deng et al. [30]. Moriyama et al. [33] showed that zk-privacy is equivalent to ind-privacy [18] which was proposed by Juels and Weis.

e) Universal composable (UC) model-based privacy notion: UC is a powerful notion proposed by Canetti [51] to describe cryptographic protocols that behave like ideal functionality, and can be composed in arbitrary way. This is known as the strongest (computational) security model for cryptographic protocols. Several UC-based frameworks have been proposed for achieving RFID privacy [34–37].

III. DISCUSSIONS ON UNPREDICTABILITY-BASED PRIVACY NOTIONS
Each category of privacy notion has its own advantages. The unpredictability-based privacy notions can be easily applied for analysing symmetric key-based protocols. These protocols rely on relatively resource-friendly building blocks such as hash function and block cipher, and are suitable for low-cost RFID tags.

The first unpredictability-based privacy notion, called unp\*-privacy, was proposed by Ha et al. [25], and further strengthened to unp\#-privacy [26], then unp\#-privacy [28], and finally unp\#-privacy [29]. In [29], Yang et al. claimed that unp\#-privacy does not imply ind-privacy, which is in contrast to the previous belief that unp\#-privacy is stronger. We will show that their claim is not sound, and discuss the deficiency of unp\#-privacy.
A. Unp*-privacy

We first briefly review the RFID system model and the adversary model of unp*-privacy. An RFID system consists of a reader \( R \) and a set of tags \( T \). An RFID tag/mutual authentication protocol contains three rounds. A reader first sends a challenge \( c \) to a tag, then the tag responses with a message \( r \), and finally the reader sends the last message \( f \). \( P_c, P_r, \) and \( P_f \) are \( c, r, \) and \( f \)’s message spaces respectively.

An adversary is given access to the following oracles:

- \( O_1 \): Upon queried, the reader initializes a session, and returns \((sid, c)\).
- \( O_2 \): On inputs \((T_i, sid, c)\), it returns a message \( r \).
- \( O_3 \): On inputs \((sid, c, r)\), it returns a message \( f \).
- \( O_4 \): On an input \( T_i \), it returns the tag \( T_i \)’s secret keys and internal state information.

Let \( O \) denote the set of the four oracles \( \{O_1, O_2, O_3, O_4\} \) specified above. An adversary is a \((t, n_1, n_2, n_3, n_4)\)-adversary, if it makes oracle queries to \( O_1 \) without exceeding \( n_i \) times respectively, where \( 1 \leq i \leq 4 \), and the running time is at most \( t \).

We use the following notations. If \( A(\cdot, \cdot, \cdot) \) is a randomized algorithm, then \( y \leftarrow A(x_1, x_2, \ldots; \rho) \) means that \( y \) is assigned with the unique output of algorithm \( A \) on inputs \( x_1, x_2, \ldots \) and coins \( \rho \), while \( y \leftarrow A(x_1, x_2, \ldots) \) is a shorthand for first picking \( \rho \) at random and then setting \( y \leftarrow A(x_1, x_2, \ldots.) \leftarrow A(O_{1}, \ldots, O_{i}(x_1, x_2, \ldots) \) denotes that \( y \) is assigned with the output of algorithm \( A \) which takes \( x_1, x_2, \ldots \) as inputs and has oracle access to \( O_{1}, \ldots, O_{i} \). \( \Pr[E] \) denotes the probability that an event \( E \) occurs.

Now we introduce unp*-privacy. Intuitively, achieving unp*-privacy requires protocol transcripts to be unpredictable, and protocol execution results to be unobservable. The experiment \( \text{Exp}_{\mathcal{A}}^{\text{unp}^*}[\kappa, l, n_1, n_2, n_3, n_4] \), denoted as \( \text{Exp}_{\mathcal{A}}^{\text{unp}^*} \) for short, is illustrated in Figure 1. Given the security parameter \( \kappa \), an RFID system is set up with a reader \( R \) and a set of \( l \) tags, where \( l \) is polynomial to \( \kappa \). An adversary \( A \) can launch oracle queries without exceeding \( n_1, n_2, n_3, \) and \( n_4 \) overall calls to \( O_1, O_2, O_3, \) and \( O_4 \) respectively throughout the experiment. \( \mathcal{A} \) consists of two algorithms, \( A_1 \) and \( A_2 \), which run in two stages, the learning stage and the guess stage, respectively. In the learning stage, \( A_1 \) queries the four oracles, and outputs an uncorrupted challenge tag \( T_c \) and state information \( st \). Then the experiment chooses \( b \in_R \{0,1\} \). In the guess stage, if \( b = 1 \), the experiment forwards \( A_2 \)’s queries to the oracles and returns the results, so that \( A_2 \) can really interact with the reader and \( T_c \); else, the experiment returns random values from appropriate message spaces. Finally, \( A_2 \) guesses \( b \)’s value and outputs \( b^* \). The experiment outputs \( 1 \) if \( b^* = b \), and outputs \( 0 \) otherwise.

**Definition 3.1:** The advantage of adversary \( \mathcal{A} \) in the experiment \( \text{Exp}_{\mathcal{A}}^{\text{unp}^*} \) is defined as:

\[
\text{Adv}_{\mathcal{A}}^{\text{unp}^*}(\kappa, l, n_1, n_2, n_3, n_4)
= |\Pr[\text{Exp}_{\mathcal{A}}^{\text{unp}^*}(\kappa, l, n_1, n_2, n_3, n_4) = 1] - \frac{1}{2}|,
\]

where the probability is taken over the choice of the tag set \( T \) and the coin tosses of the adversary \( \mathcal{A} \).

![Fig. 1: Unp*-Privacy Experiment](image)

**Definition 3.2:** An adversary \( A(e, t, n_1, n_2, n_3, n_4) \)-breaks the unp*-privacy of the RFID system \((R, T)\) if the advantage \( \text{Adv}_{\mathcal{A}}^{\text{unp}^*}(\kappa, l, n_1, n_2, n_3, n_4) \) of \( \mathcal{A} \) in the experiment \( \text{Exp}_{\mathcal{A}}^{\text{unp}^*} \) is at least \( \epsilon \), and the running time of \( \mathcal{A} \) is at most \( t \).

**Definition 3.3 (Unp*-Privacy):** An RFID system \((R, T)\) is said to be \((e, t, n_1, n_2, n_3, n_4)\)-unp*-private, if for all sufficiently large \( \kappa \) there exists no adversary who can \((e, t, n_1, n_2, n_3, n_4)\)-break the unp*-privacy of \((R, T)\) for any \((e, t)\), where \( t \) is polynomial in \( \kappa \) and \( e \) is non-negligible in \( \kappa \).

B. Correction on the relation between ind-privacy and unp*-privacy

Li et al. proved that unp*-privacy implies ind-privacy [28]. However, Yang et al. claimed that unp*-privacy does not imply ind-privacy [29]. To support this claim, they provided a counterexample, formally proved that it satisfies unp*-privacy, and then showed that it does not satisfy ind-privacy through a traceability attack. However, we discover that the counterexample does not satisfy unp*-privacy in the first place.

We review the counterexample first. Let \( F : \{0,1\}^{16} \times \{0,1\}^{16} \rightarrow \{0,1\}^{16} \) be a PRF family, \( \text{ctr} \in \{0,1\}^{16} \) be a counter, and \( \text{pad} \in \{0,1\}^{16} \) be a padding, where \( l_c + l_{\text{pad}} = l_4 \) and \( l_4 \) is the length of the challenge. Each tag \( T_i \) has a unique identity \( ID_i \), and is assigned with a secret key \( k_i \in_R \{0,1\}^{16} \). \( T_i \) stores \( k_i \), a counter \( \text{ctr}_i \) with an initial value 1, and a one-bit tag state \( st_i \) with an initial value 0. The protocol works as follows.

1. The reader \( R \) chooses \( c \in_R \{0,1\}^{16} \) and sends it to \( T_i \).
2. Upon receiving \( c \), the tag \( T_i \) chooses \( r \in_R \{0,1\}^{16} \) first. Then \( T_i \) calculates \( r_1 = F_k(c || \text{pad}) || \text{ctr}_i \) if \( st_4 = 0 \); else \( r_1 = F_k(c || r_2) \oplus \text{ctr}_i \). Finally, \( T_i \) updates the counter as \( \text{ctr}_i = \text{ctr}_i + 1 \); sets \( st_4 = 1 \); and sends \((r_1, r_2)\) to \( R \).
3. Upon receiving \((r_1, r_2)\) from \( T_i \), the reader \( R \) searches for the matching tag in its database. If \( R \) discovers a tuple \((k, \text{ctr}, ID)\) such that \( \text{ctr} = F_k(c || \text{pad}) \) in the database, then \( R \) accepts \( T_i \) as the tag with \( ID \). Then \( R \) updates \( \text{ctr} \leftarrow \text{ctr} + 1 \), computes \( f = F_k(c || \text{ctr} || r_2) \); or else if there exists \((k, \text{ctr}, ID)\) such that \( \text{ctr} = F_k(c || r_2) \) in the database, then \( R \) accepts \( T_i \) as the tag with \( ID \). In this case, \( R \) update \( \text{ctr} \leftarrow \text{ctr} + 1 \), computes \( f = F_k(c || \text{ctr} || r_2) \); or else, \( R \) rejects \( T_i \) and chooses \( f \in_R \{0,1\}^{16} \). At last, \( R \) sends \( f \) to \( T_i \).

In the counterexample, some inputs of \( F \) are not with the length \( l_d \). We do not correct these mistakes.
4) Upon receiving \( f \), if \( f = F_k(c)[ctr_1]|r_2) \), \( T_i \) sets \( st_i = 0 \) and accepts the reader \( R \); otherwise, rejects \( R \).

Now we show that the counterexample does not satisfy unp*-privacy. An adversary \( A \) can break unp*-privacy as follows. In the learning stage, \( A_1 \) triggers a valid tag \( T_i \), which is in state 0 to run a session with the reader without modifying the messages. To do so, \( A_1 \) first calls \( O_1 \) and gets \((sid,c)\), then calls \( O_2 \) with inputs \((T_i,sid,c)\) and gets \( r_1 \), where \( r_1 = F_k(c)[pad] \oplus ctr_1 \); and finally calls \( O_3 \) with inputs \((sid,c,r)\) and gets \( f \). After running the above session, \( T_i \)’s state \( st_i \) is still 0. Now \( A_1 \) submits \( T_i \) as the target tag. After the experiment tosses the coin \( b \), in the guess stage, \( A_2 \) calls \( O_2 \) with inputs \((T_i,sid,c)\) and gets \( r' \), where \( sid' \) could be any value in the session ID’s space. As known, if \( b = 1 \), the experiment sets \( r' = F_k(c)[pad] \oplus (ctr_1+1); \) else, sets \( r' \in R \). With the knowledge of \( F_k(c)[pad] \oplus ctr_1 \), \( A_2 \) can differentiate \( F_k(c)[pad] \oplus ctr_1 \) from a random string. Thus \( A_2 \) can successfully guess the value of \( b \).

Yang et al. showed a traceability attack against the counterexample. If the tag \( T_i \)’s state \( st_i \) is 0, the reader can authenticate \( T_i \) with \( r_1 \) without checking \( r_2 \)’s integrity. Thus, an adversary can infer the value of \( st_i \) by modifying \( r_2 \) and observing the reader’s protocol execution result. If the reader \( R \) accepts \( T_i \), the adversary knows that the value of \( st_i \) is 0; or else, the value of \( st_i \) is 1. This kind of traceability attack cannot be captured by unp*-privacy due to two reasons. First, an adversary cannot access protocol execution results in the unp*-privacy experiment. Second, the soundness experiment [28] for tag authentication does not require the transcripts of a session to be matching (as defined in [30]) on the reader side and the tag side.

C. Unp*-privacy

To capture the above traceability attack, Yang et al. proposed a new privacy notion, called unp*-privacy. The experiment \( \text{Exp}_{A}^{\text{unp}} \) is illustrated in Figure 2. Intuitively, unp*-privacy requires that protocol transcripts are pseudorandom, and any modification on the second/third message would result in “rejection” by the reader/tag. Compared to unp*-privacy, unp*-privacy is different in several ways. First, an adversary can access one additional oracle \( O_5 \). On input \((sid,f,T_i)\), \( O_5 \) returns \( o_T \), which is \( T_i \)’s execution result of the session \( sid \). Second, without being specified explicitly, an adversary can get the reader’s protocol execution results through querying \( O_3 \). Third, in the guess stage, if \( b = 0 \), the experiment provides not only random chosen “protocol transcripts”, but also “protocol execution results”. The experiment outputs an “execution result” as “1” when \( O_2/O_5 \) is queried with the output of its preceding oracle, \( O_2/O_3 \), without modification; or “0” otherwise.

**Definition 3.4:** The advantage of adversary \( A \) in the experiment \( \text{Exp}_{A}^{\text{unp}} \) is defined as:

\[
\text{Adv}_{A}^{\text{unp}}(\kappa,l,n_1,n_2,n_3,n_4) = |\mathbb{P}[\text{Exp}_{A}^{\text{unp}}(\kappa,l,n_1,n_2,n_3,n_4) = 1] - \frac{1}{2}|,
\]

where the probability is taken over the choice of the tag set \( T \) and the coin tosses of the adversary \( A \).

**Definition 3.5:** An adversary \( A(\epsilon,t,n_1,n_2,n_3,n_4) \)-breaks the unp*-privacy of the RFID system \((R,T)\) if the advantage \( \text{Adv}_{A}^{\text{unp}}(\kappa,l,n_1,n_2,n_3,n_4) \) of \( A \) in the experiment \( \text{Exp}_{A}^{\text{unp}} \) is at least \( \epsilon \), and the running time of \( A \) is at most \( t \).

**Definition 3.6 (Unp*-Privacy):** An RFID system \((R,T)\) is said to be \((\epsilon,t,n_1,n_2,n_3,n_4)\)-unp*-private, if for all sufficiently large \( \kappa \) there exists no adversary who can \((\epsilon,t,n_1,n_2,n_3,n_4)\)-break the unp*-privacy of \((R,T)\) for any \((\epsilon,t)\), where \( t \) is polynomial in \( \kappa \) and \( \epsilon \) is non-negligible in \( \kappa \).

**Discussions.** We notice several issues with the definition of unp*-privacy. First, \( O_5 \) is too powerful. An adversary can query \( O_5 \) with arbitrarily selected inputs \((sid,c,r)\), and get a response \( f \). This contradicts the common assumption that the reader’s internal routine cannot be interfered by an adversary. Second, to satisfy unp*-privacy, the reader is required to send a dummy third-round message \( f \) even if it has rejected the tag after receiving an invalid second round message, which is obviously not necessary. Third, no session management is defined in \( \text{Exp}_{A}^{\text{unp}} \). It is not clear what will happen when an adversary queries an oracle with the same inputs multiple times, or with different inputs that share the same \( sid \) multiple times.

IV. FORMAL FRAMEWORK FOR RFID SYSTEM WITH PoP

We propose the first formal framework for RFID tag/mutual authentication with PoP under a refined RFID system model. Our framework also addresses the above issues of unp*-privacy.

A. System model

Among the existing RFID system models for tag/mutual authentication, Deng et al.’s [30] is considered to be more
generic and elaborate than others. However, their model requires a reader/tag to continue sending a reply instead of terminating the execution upon receiving an invalid message. We refine Deng et al.’s model to allow protocol terminations during executions.

**Definition 4.1:** An RFID system RS is defined to be a tuple $(R, T, \text{Setup}, \pi)$, where

- Setup$(\kappa, l)$ initializes the whole system RS with a single legitimate reader $R$ and $l$ tags $T$ with public system parameter $\sigma$, where $l$ is polynomial in the security parameter $\kappa$. Each tag $T_i \in T$, for $1 \leq i \leq l$, is assigned with an unique identity $ID_i$, public parameters $\xi_{T_i}$, an initial secret key $k_{T_i}^1$, and an initial internal state $s_{T_i}^1 = \emptyset$. The reader $R$ is assigned with a secret key $k_R$, an initial internal state $s_R = \emptyset$, and a database $DB^1$.

- Protocol $\pi(R, T_i)$, denoted as $\pi$ for short, is a $(2\gamma + 1)$-round interactive protocol between $R$ and $T_i$ Each session of $\pi$ is initialized by the reader with a fresh session ID $sid$ randomly selected in its space $P_{sid}$. $c_\mu/\alpha_\mu$ denotes the $\mu$-th reader-to-tag/tag-to-reader message in a session, and $P_{sid}/P_{\alpha_\mu}$ denotes $c_\mu/\alpha_\mu$’s message space, for $1 \leq \mu \leq (\gamma + 1)/\gamma$. $RT_i$ terminates a session by outputting $o_{\mu\alpha_i}$, which is a one-bit value for showing its protocol execution result of the session. $o_{\mu\alpha_i}$ is set as “1” for acceptance, “0” for denial, and “null” if the execution result is not available. $o_{\mu\alpha_i}$ can be sent through a channel different from the wireless channel used by protocol messages (e.g., sound channel). The transcripts of a session are defined as $tr_{s_{\alpha_i}} = (sid, c_1, \ldots, c_\gamma, \alpha_\gamma, c_{\gamma+1}, o_{\mu\alpha_i}, T_i)$.

We assume that both of the reader and the tags process protocol sessions sequentially, and a tag can run the protocol without exceeding $s$ times in its lifetime.

**1. On reader side:** Suppose the reader $R$ is with current state $s_R^j$, current database $DB^j$, for $1 \leq j \leq l$; there are three cases.

- **Case 1:** $R$ can start its $j$-th session if $s_R^j = \emptyset$. It takes $para, k_R, s_R^j$, and $DB^j$ as inputs, and generates $s_R^{j'}$ which contains partial transcripts $tr_{s_{\alpha_i}} = (sid, c_1)$ and other temporary information, e.g. random coins for this session, where $sid$ is a fresh session ID, and $c_1$ is the first message in this session. Then $R$ updates $s_R^j$ to $s_R^{j'}$, and sends $(sid, c_1)$ to $T_i$.

- **Case 2:** The reader receives a message $(sid', \alpha)$ when $s_R^j \neq \emptyset$. It updates $s_R^j$ to $s_R^{j'}$, if $s_R^{j'} \neq \emptyset$ means that $R$ is currently running a session with partial transcripts $tr_{s_{\alpha_i}} = (sid, c_1, \ldots, c_\mu)$ in $s_R^j$, where $\mu \in [1, \gamma]$. $\pi_R$ takes para,

\[ k_R, s_R^j, DB^j \text{, and } (sid', \alpha) \text{ as inputs, and proceeds as follows:} \]

1. If $sid' = sid$, $\mu \in [1, \gamma]$, and $\alpha$ is valid, then $R$ generates $s_R^{j'}$ including a reply $c_{\mu+1}$, updates $s_R^j$ to $s_R^{j'}$, and then sends $(sid, c_{\mu+1})$ to $T_i$.

2. If $sid' = sid$, $\mu = \gamma$, and $\alpha$ is valid, then $R$ first generates $s_R^{j'}$ including the final $c_{\mu+1}$, updates $s_R^j$ to $s_R^{j'}$, and sends $(sid, c_{\mu+1})$ to $T_i$. Then $R$ outputs $o_R = 1$, updates $DB^j$ to $DB^{j+1}$, and initializes $s_R^{j+1}$ as $\emptyset$.

3. If $sid' = sid$, $\mu \in [1, \gamma]$, and $\alpha$ is invalid, then $R$ outputs $o_R = 0$, updates $DB^j$ to $DB^{j+1}$, and initializes $s_R^{j+1}$ as $\emptyset$.

4. If $sid' \neq sid$, $R$ simply ignores the message.

**Case 3:** If $R$ is running a session $sid$ with partial transcripts $tr_{s_{\alpha_i}} = (sid, c_1, \ldots, c_\mu)$ in $s_R^j$, but fails to receive a message $(sid, \alpha)$ within a pre-fixed time period, then it outputs $o_R = 0$, updates $DB^j$ to $DB^{j+1}$, and initializes $s_R^{j+1}$ as $\emptyset$.

**2. On tag side:** Suppose the tag $T_i$ is with current key $k_T^i$, and internal state $s_T^j$, for $1 \leq v \leq s$, and there is a coming message $(sid, c_i, T_i)$, $T_i$ deals with $(sid, c_i)$ as follows.

**Case 1:** If $c_i \in P_{\alpha_i}$ (it means that $c_i$ is a message for starting a new session), there are two subcases.

1. If $s_T^j = \emptyset$, then $T_i$ generates $s_T^{j'}$ including partial transcripts $tr_{s_{\alpha_i}} = (sid, c_i, \alpha_1)$ and the random coins for the session $sid$ (if any), where $\alpha_1$ is the reply for $c_i$. Then $T_i$ updates $s_T^v$ to $s_T^{v'}$, and sends $(sid, \alpha_1)$ to the reader $R$.

2. If $s_T^j \neq \emptyset$ (it means that $T_i$ is currently running a session), $T_i$ outputs $o_R = 0$. Then $T_i$ erases its internal state $s_T^j$, updates its key to $k_T^{j+1}$, generates $s_T^{j+1}$ including partial transcripts $tr_{s_{\alpha_i}} = (sid, c_i, \alpha_1)$ and random coins (if any), where $\alpha_1$ is the reply for $c_i$. Finally, $T_i$ sends $(sid, \alpha_1)$ to the reader $R$.

**Case 2:** $c_i \notin P_{\alpha_i}$ and $\exists tr_{s_{\alpha_i}} = (sid, c_1, \ldots, c_\mu)$ in $s_T^j$; there are two sub-cases.

1. If $\mu \in [2, \gamma]$: If $c_i$ is valid, $T_i$ generates $s_T^{j'}$ including a reply $\alpha_{\mu}$; updates $s_T^{j'}$ with $s_T^{j'}$, and sends $(sid, \alpha_{\mu})$ to the reader $R$; else, $T_i$ outputs $o_R = 0$. Then $T_i$ erases its internal state $s_T^j$, updates its secret-key $k_T^j$ to $k_T^{j+1}$, and sets internal state $s_T^{j+1} = \emptyset$ for its next session.

2. If $\mu = \gamma + 1$: If $c_i$ is valid, $T_i$ sets $o_R = 1$; else, sets $o_R = 0$. Then $T_i$ outputs $o_R$, erases its internal state $s_T^j$, updates $k_T^j$ to $k_T^{j+1}$, and sets $s_T^{j+1} = \emptyset$.

**Case 3:** In any other cases, $T_i$ simply ignores this message.

Based on the refined model $RS$, we define an RFID system model $RS^*$ that supports tag/mutual authentication and PoP.

**Definition 4.2:** An RFID system $RS^*$ that supports tag/mutual authentication and PoP is defined to be a tuple $(R, T, \text{Setup}^*, \pi^*, \text{CredGen}, \text{CredVeri})$. 
• Setup*(κ,l) is defined the same as in Definition [4.1]

• Protocol π*(R,T) is also defined the same as in Definition [4.1]

• CredGen(para,kR,sRj,para,cred,para,DB): Given inputs para, kR, sRj, and DB3, CredGen outputs a PoP credential cred if the reader’s j-th session’s execution result oR = 1, else outputs ⊥.

• CredVeri(para,cred): Given a PoP credential cred and the system’s public parameters para, it outputs the verification result “1” if cred is a valid one, and “0” otherwise.

B. Adversary model

A probabilistic polynomial-time concurrent man-in-the-middle (CMIM) adversary A against RS* can access five oracles, InitReader, SendT, SendR, CorruptTag, and GetCred. The first four oracles are defined the same as in [30]. The oracle GetCred is for capturing the adversary’s capabilities to gain PoP credentials. The adversary can also obtain protocol execution results through SendT and SendR.

• InitReader(): A invokes the reader R to start a new session of protocol π*, and returns (sid,c1) to the adversary.

• SendT(Ti, sid,c): A sends a message (sid,c) to the tag Ti. Ti returns a message (sid,α) (if any) and an execution result oR (if any) to the adversary.

• SendR(sid,α): A sends a message (sid,α) to the reader. The reader returns a message (sid,α) (if any) and an execution result oR (if any) to the adversary.

• CorruptTag(Ti): A gets all the contents stored on Ti, including Ti’s current secret key kR and internal state sRi.

• GetCred(sid): Given a session ID sid, if the reader R has ever been involved in a session with sid as its j-th session (without loss of generality), the oracle returns the result of CredGen(para,kR,sRj,para,cred,para,DB), or else, returns ⊥.

Let O′1, O′2, O′3, O′4, and O′5 denote the above oracles respectively. And O′ denotes the set of them. An adversary is a (t, n1, n2, n3, n4, n5)-adversary, if it makes oracle queries to O′i without exceeding ni times respectively, where 1 ≤ i ≤ 5, and the running time is at most τ.

C. Our new privacy and security notions

Our framework utilizes two existing security notions defined in [30], namely adaptive completeness and tag/mutual authentication, and two new notions, unp*-privacy and PoP credential unforgeability.

1) Unp*-privacy: Intuitively, unp*-privacy requires that: (1) protocol messages should be pseudorandom; (2) every message (except the first one which normally contains a random value) should be authenticated. Compared with unp-privacy, it is not constrained to analyse three-round protocols only, it is defined under a more reasonable adversary model, and it is defined with delicate session control when b = 0 holds in the guess stage.

The unp*-privacy experiment ExpA[unp*,κ,l,n1,n2,n3,n4,n5] is illustrated in Figure 3. To conduct delicate session control, we define two types of sessions. If a session sid is initialized through querying O′1, it is considered as a session between the reader R and the target tag Tc. Its transcripts are recorded with trs*sid. Later, when the adversary sends a message (sid,c) to Tc through O′2, the experiment checks trs*sid if the message c is the same as the last message cµ stored in trs*sid, which means that c came from the reader R without being changed, the experiment returns the next message αµ ∈ R Fαµ (if 1 ≤ µ ≤ γ) or oR = 1 (if µ = γ + 1), and updates trs*sid by adding αµ or oR ; or else, the message c is considered invalid, the experiment returns oR = 0, and updates trs*sid by adding oR. The queries to O′3 are processed similarly. The second type of sessions are the ones initialized through querying O′2. An adversary chooses (sid,c) and queries O′2 to start a session with Tc. The transcripts are recorded with trsAdv*sid. The adversary is allowed to get a response from O′2, but if it queries O′2 again with the third round protocol message (sid,c′), the experiment would output oR = 0 and terminate the session.

Definition 4.3: The advantage of adversary A in the experiment ExpA[unp*,κ,l,n1,n2,n3,n4,n5] is defined as:

\[ Adv_A[unp*,κ,l,n1,n2,n3,n4,n5] = |Pr[Exp_A[unp*,κ,l,n1,n2,n3,n4,n5] = 1] - \frac{1}{2} |, \]

where the probability is taken over the choice of the tag set T and the coin tosses of the adversary A.

Definition 4.4: An adversary A (κ,l,n1,n2,n3,n4,n5)-breaks the unp*-privacy of the RFID system (R,T) if the advantage AdvA[unp*,κ,l,n1,n2,n3,n4,n5] of A in the experiment ExpA[unp*] is at least ε, and the running time of A is at most t.

Definition 4.5 (Unp*-Privacy): An RFID system (R,T) is said to be (κ,l,n1,n2,n3,n4,n5)-unp*-private, if for all sufficiently large κ, there exists no adversary who can (κ,l,n1,n2,n3,n4,n5)-break the unp*-privacy of (R,T) for any (κ,l), where t is polynomial in κ and ε is non-negligible in κ.

2) PoP credential unforgeability: Intuitively, PoP credential unforgeability has two folds of meanings. Each credential regarding the reader and a valid tag should correspond to a completed protocol session between the reader and the tag. And an adversary should not be able to generate a valid PoP credential regarding itself and a valid tag.

PoP credential unforgeability is formalized by the experiment ExpA[cred,unfρ1][κ,l] shown in Figure 3. The adversary A is allowed to query the five oracles without exceeding n1,n2,n3,n4,n5 times respectively. Then it outputs a PoP credential cred. Denote by E1 the event under both of the following conditions: (1) the PoP credential corresponds to the reader R and an uncorrupted tag Ti; (2) for the reader R, there does not exist j ∈ [1,ls], such that CredGen(para,kR,sRj,para,cred,para,DB) = cred and CredVeri(i,para,cred) = 1. Denote by E2 the event under the condition that the PoP credential corresponds to an adversary A and an uncorrupted tag Ti and CredVeri(i,para ∪ paraA,cred) = 1, where paraA are the adversary A’s public parameters.
Experiment $\text{Exp}_A^{\text{unp-Priv}}[\kappa, n_1, n_2, n_3, n_4, n_5]$
1. run $\text{Setup}(c)$ to setup $(R, T)$.
2. $(T_*, s_t) \leftarrow A^\text{Priv}_{\kappa}[R, T]$.
3. select $b \in_R [0,1]$.
4. $b' \leftarrow A_2(O_1, O_2, O_3)(R, T, s_t)$;
   (1) if $b = 1$,
      when $A_2$ queries $O_1, O_2$, and $O_3$, return the results of
      the oracles.
   (2) if $b = 0$,
      a. when $A_2$ queries $O'_1$, return $\sigma_{id} \in R P_{sid}$ and
         $c_1 \in_R P_{c_1}$, create $tr_s = (\sigma_{id}, c_1)$;
      b. when $A_2$ queries $O'_2$ with parameters $(\sigma_{id}, c)$,
         if $\exists tr_s = (\sigma_{id}, c, \mu) \land c \in P_{c_1}$,
         choose $\alpha \in_R P_{\alpha_1}$, create $tr_{Adv} = (\sigma_{id}, c, \alpha)$,
         return $(\sigma_{id}, \alpha)$;
         if $\exists tr_s = (\sigma_{id}, c, \mu) \land \mu < \gamma + 1$, then
         if $c_1 = c$, then choose $\alpha \in_R P_{\alpha_1}$,
         update $tr_s = tr_{Adv} \cup \alpha$, return $(\sigma_{id}, \alpha)$;
         $(tr_{Adv} \cup \alpha$ means that $\alpha$ is appended to $tr_{Adv}$)
         else
         if $\mu = 1 \land c \in P_{c_1}$,
         return $\sigma_{id}$; if $c_1$ from $R$ has been modified to $c$, then the “tag”
         //responds with $\alpha$. $\alpha$ will be rejected by the
         //reader later, so $\sigma_{id}$ is set as 0 in advance.
         then choose $\alpha \in_R P_{\alpha_1}$, set $\sigma_{id} = 0$,
         update $tr_{Adv} = tr_{Adv} \cup \alpha \oplus \sigma_{id}$,
         return $(\sigma_{id}, \alpha)$;
         else set $\sigma_{id} = 0$, update $tr_{Adv} = tr_{Adv} \cup \sigma_{id}$,
         return $\sigma_{id}$;
         if $\exists tr_{Adv} = (\sigma_{id}, \alpha, \sigma_{id} = 1)$, then
         if $c_{\gamma} = c$, then set $\sigma_{id} = 1$; else, set $\sigma_{id} = 0$;
         update $tr_{Adv} = tr_{Adv} \cup \sigma_{id}$, return $\sigma_{id}$;
         if $\exists tr_{Adv} = (\sigma_{id}, \alpha_1)$,
         then $\sigma_{id} = 0$,
         update $tr_{Adv} = tr_{Adv} \cup \sigma_{id} \oplus \sigma_{id}$,
         return $\sigma_{id}$; else ignore this query;
      c. when $A_2$ queries $O'_3$ with parameters $(\sigma_{id}, \alpha)$,
         if $\exists tr_{Adv}$, then ignore this query;
         if $\exists tr_{Adv} = (\sigma_{id}, \alpha, \sigma_{id} = 0)$, then
         return $\sigma_{id} = 0$;
         if $\exists tr_{Adv} = (\sigma_{id}, \alpha_1, \sigma_{id} = 0)$, then
         if $\alpha = \alpha_1$, then choose $c \in_R P_{c_{\alpha_1}+1}$,
         if $\mu < \gamma$, then update $tr_{Adv} = tr_{Adv} \cup c$,
         return $(\sigma_{id}, c)$;
         else
         set $\sigma_{id} = 1$;
         update $tr_{Adv} = tr_{Adv} \cup c \oplus \sigma_{id}$,
         return $(\sigma_{id}, c)$;
         else set $\sigma_{id} = 0$;
         update $tr_{Adv} = tr_{Adv} \cup \sigma_{id}$,
         return $\sigma_{id}$;
         else ignore this query.
5. output $1$ if $b' = b$, and $0$ otherwise.

Fig. 3: Unp-Privacy Experiment

Experiment $\text{Exp}_A^{\text{CredUfrg}}[\kappa, l, n_1, n_2, n_3, n_4, n_5]$
1. run $\text{Setup}(\kappa, l)$ to setup $(R, T)$.
2. $\text{cred} \leftarrow A_2^\text{CredUfrg}(R, T)$.

Fig. 4: PoP Credential Unforgeability Experiment

Definition 4.6: An adversary $A$ $(\epsilon, t, n_1, n_2, n_3, n_4, n_5)$-breaks PoP credential unforgeability of an RFID system $(R, T)$ if the probability that event $E_1$ or $E_2$ occurs is at least $\epsilon$ with the running time at most $t$, and $A$ is a $(\kappa, l, n_1, n_2, n_3, n_4, n_5)$-adversary.

Definition 4.7 (Credential unforgeability): The RFID system $(R, T)$ satisfies PoP credential unforgeability, if for all sufficiently large $\kappa$ there exists no adversary $A$ that can $(\epsilon, t, n_1, n_2, n_3, n_4, n_5)$-break the credential unforgeability of $(R, T)$ for any $(\epsilon, t)$, where $t$ is polynomial in $\kappa$ and $\epsilon$ is non-negligible in $\kappa$.

V. GENERIC CONSTRUCTION

In this section, we provide a generic construction to transform an RFID tag/mutual authentication protocol to one with PoP, conduct security analysis, and discuss its practicability.

A. Cryptographic primitives

The cryptographic primitives used in our construction include pseudorandom function, cryptographic hash function, and digital signature scheme.

Pseudorandom function (PRF) [28] Let $F : \mathcal{K} \times D \to \mathcal{R}$ be a family of functions, where $\mathcal{K}$, $D$, and $\mathcal{R}$ denote the set of keys (or indexes), the domain, and the range of $F$ respectively. Let $|\mathcal{K}| = \gamma$, $|D| = m$, and $|R| = n$. Let $\text{Rand}^{D \to R}$ be the family of all functions with domain $D$ and range $\mathcal{R}$. A polynomial time predictable test (PTPT) for $F$ is an experiment, where a probabilistic polynomial time algorithm $T$, given $\gamma$, $m$, $n$ as input and with access to an oracle $O_f$ for a function $f \in_R F$ or $f \in_R \text{Rand}^{D \to R}$, outputs either $0$ or $1$. Figure 5 shows a PTPT for $F$.

Experiment $\text{Exp}_T^{\text{PTPT}}[F, \gamma, m, n]$
1. select $b \in_R [0,1]$.
2. if $b = 1$, select $k \in_R \mathcal{K}$ and set $f = F_k$;
   otherwise, $f \in_R \text{Rand}^{D \to R}$.
3. $b' \leftarrow O_f$.

Fig. 5: Polynomial Time Predictable Test

Definition 5.1: An algorithm $T$ passes the PTPT for the function family $F$ if it correctly guesses the random bit which is selected by the PTPT experiment, namely $b' = b$. The advantage of algorithm $T$ is defined as

$$\text{Adv}_T(\gamma, m, n) = |Pr[b' = b] - \frac{1}{2}|,$$

where the probability is taken over the choice of $f$ in $F$ and the coin tosses of algorithm $T$.
Definition 5.2: A function family $F : \mathcal{K} \times \mathcal{D} \rightarrow \mathbb{R}$ is said to be a pseudorandom function family if it has the following properties:

- **Indexing**: Each function in $F$ has a unique $\gamma$-bit key (index) associated with it. It is easy to select a function $f \in F$ randomly if $\gamma$ random bits are available.
- **Polynomial Time Evaluation**: There exists a polynomial time algorithm such that, given input of a key (index) $k \in \mathcal{K}$ and an argument $x \in \mathcal{D}$, it outputs $F_k(x)$.
- **Pseudorandomness**: No probabilistic polynomial time algorithm $T$ can pass the PTPT for $F$ with non-negligible advantage.

**Digital signature scheme** A digital signature scheme $\text{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ consists of three algorithms: (1) a randomized key generation algorithm $\mathcal{K}: (PK, sk) \xrightarrow{\}$ $\mathcal{K}(n)$ which returns a pair of public key and private key $(PK, sk)$ given a security parameter $\kappa$; (2) a signing algorithm $\mathcal{S}: \sigma \xrightarrow{\}$ $\mathcal{S}_k(M)$ (may be randomized or stateful) that takes a secret key $sk$ and a message $M$ as input and outputs a signature $\sigma \in \{0, 1\}^* \cup \{\bot\}$; (3) a deterministic verification algorithm $\mathcal{V}: d \xrightarrow{\} \mathcal{V}_{PK}(M, \sigma)$ that takes a public key $PK$, a message $M$, and a signature $\sigma$ as input and outputs a bit $d$, where $d = 1$ if $\sigma$ is valid and $d = 0$ if $\sigma$ is invalid. Existential unforgeability under an adaptive chosen-message attack (EU-CMA) is a widely used security notion for digital signature scheme.

**B. Generic construction**

Given an RFID system $\mathcal{RS} = (R, T, \text{Setup}, \pi)$ as defined in Definition 4.1, our construction $\mathcal{RS}^* = (R, T, \text{Setup}^*, \pi^*, \text{CredGen}, \text{CredVerify})$ is shown below.

\[ a) \text{Setup} \rightarrow \text{Setup}^*: \mathcal{RS}^* \] introduces a signature scheme $\mathcal{DS}$, a cryptographic hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^{2l}$, and a PRF family $G : \{0, 1\}^{l+x} \times \{0, 1\}^{2l} \rightarrow \{0, 1\}^{2l}$ as additional building blocks, where $l_k$ and $l_p$ are polynomial to the security parameter $\kappa$ of $\mathcal{RS}$. Each tag is embedded with an additional $l_k$-bit key $k^*_T$, with an initial value $v^*_T$. Each tag and the reader are assigned with a unique pair of public key and secret key for the signature scheme, denoted as $(PK_T, sk_T)$ for $1 \leq i \leq l$, and $(PK_R, sk_R)$ respectively. The reader's secret key, initial state, and each tag $T_i$'s secret key, initial state, public parameters are updated/copied from $\mathcal{RS}^*$'s settings accordingly: $k^*_R = (k_{R}, sk_{R})$, $s^*_R = s^*_R$, $k^*_T = (k^*_T, k^*_T, sk^*_T)$, $s^*_T = s^*_T$, $\xi^*_T = (\xi^*_T, PK_T)$, $DB^*_T$ is updated such that for each $T_i$, $\text{rcd}^*_T = \text{rcd}^*_T, k^*_T, PK_T$, where $\text{rcd}^*_T \in DB^*_T$. $\mathcal{RS}^*$'s public system parameters for $\pi^*$ are set as $\{\sigma^*, PK_R, \xi^*_T, \xi^*_T, \xi^*_T, \xi^*_T\}$, where $\sigma^*$ consists of $\sigma$, and all public parameters from the three new building blocks, $\mathcal{DS}$, $H$, and $G$.

b) $\pi \rightarrow \pi^*$: Given $\pi(R, T_i) = \{c_1, \ldots, c_{l+1}, o_R, o_{T_i}\}$, $\pi^*(R, T_i)$ is constructed as $\{c_1, \ldots, c_{l}, o_R, c_{l+1}, o_{T_i}, o_{T_i}\}$. The superscript $*$ denotes that the procedures for generating the message/result are different from $\pi$ or newly added. Note that $c_1$ could be null if $\pi$ consists of only $2\gamma$ rounds. And in the protocol $\pi$, the reader $R$ outputs $o_R$ and updates $T_i$'s secret key after sending $c_{l+1}$, while in the protocol $\pi^*$, the reader $R$ postpones these two steps and any other updates that depend on the reader’s execution result $o_R$ till receiving $\alpha_{\gamma+1}$. Similarly, the tag $T_i$ also postpones outputting $o_{T_i}$ and updating $T_i$’s secret key till it sends the last message $\alpha_{\gamma+1}$.

The protocol $\pi^*$ is shown in Fig. 6. We omit $sid$ in each message, and put $o_R^*, o_T^*, o_{T_i}^*$ in parentheses to demonstrate that it is sent at the same time with the corresponding message (if any) but may through a different channel. The protocol $\pi^*$’s procedures are the same as the protocol $\pi$ in generating and dealing with the first $2\gamma - 1$ messages. Suppose that $R$ currently has the secret key $sk_R$, internal state $s_{R}$ and database $DB$, and $T_i$ has the secret keys $k_{T_i}, k_{T_i}^*, sk_{T_i}$, and internal state $s_{T_i}$. Below we show the procedure dealing with the $2\gamma$-th round message $\alpha_{\gamma}$ and onwards.

- Upon receiving $\alpha_{\gamma}$ (sid is omitted for simplicity), the reader $R$ authenticates $T_i$ with $\alpha_{\gamma}$, and computes $c_{\gamma+1}$ according to $\pi$’s procedures first. Then $R$ computes $c_{\gamma+1} = H(S_{sk_R}(r), \sigma_{\gamma+1} = G_{sk_R}(H(c_1||c_1opi||c_{\gamma+1}||c_{\gamma+1})))$. For simplicity, we set $r \in \mathbb{R} \{0, 1\}^9$ here. In real applications, $r$ could be a string with arbitrary length, and contains detailed information about the session, for example, an authenticated timestamp. Finally, $R$ sends $c_{\gamma+1} = (c_{\gamma+1}, c_{\gamma+1}^*, c_{\gamma+1}^*)$ to $T_i$.
- Upon receiving $\alpha'_{\gamma+1} = (c_{\gamma+1}, c_{\gamma+1}^*, c_{\gamma+1}^*)$, if $c_{\gamma+1}$ is valid according to the protocol $\pi$’s procedures and $c_{\gamma+1}^* = \mathcal{G}_{k_{T_i}^*}(H(c_1||c_1opi||c_{\gamma+1}||c_{\gamma+1}^*)$, $T_i$ sets $o_{T_i}^* = 1$; else sets $o_{T_i}^* = 0$. If $o_{T_i}^* = 1$, $T_i$ computes $\alpha'_{\gamma+1} = \mathcal{G}_{k_{T_i}^*}(c_{\gamma+1}^*) \oplus S_{sk_{T_i}}(c_{\gamma+1}^*)$ and $\alpha''_{\gamma+1} = \mathcal{G}_{k_{T_i}^*}(S_{sk_{T_i}}(c_{\gamma+1}^*))$, then sends $\alpha_{\gamma+1} = (\alpha'_{\gamma+1}, \alpha''_{\gamma+1})$ to $R$. Finally, $T_i$ outputs $o_{T_i}^*$, erases $s_{T_i}$, updates $k_{T_i}$ according to the procedures in $\pi$, and initializes internal state for the next session as $\emptyset$.

Upon receiving $\alpha'_{\gamma+1} = (\alpha'_{\gamma+1}, \alpha''_{\gamma+1})$, $R$ computes $\sigma = o_{R}^* \oplus \mathcal{G}_{k_{T_i}^*}(\sigma_{\gamma+1}^*)$. Then $R$ sets $o_{R}^* = 1$ if $\mathcal{V}_{PK_{T_i}}(c_{\gamma+1}^*, \sigma) = 1$ and $\alpha''_{\gamma+1} = \mathcal{G}_{k_{T_i}^*}(\sigma)$; else, sets $o_{R}^* = 0$. Finally, $R$ outputs $o_{R}^*$, and updates its internal state and database.

**Fig. 6: An RFID Protocol with PoP**
c) CredGen(\(para^*, k_R^*, s_{R^*}^*, DBJ^*\)): Given system parameters \(para^*\), the reader’s secret key \(k_R^*\), and the \(j\)-th session’s internal state \(s_{R^*}^j\) and database \(DBJ^*\), the function outputs a PoP credential \(cred\) if the reader has completed the \(j\)-th session with \(o_R^j = 1\), where \(o_R^j \in s_{R^*}^j\); or else, outputs “⊥”. \(s_{R^*}^j\) contain the \(j\)-th session’s transcripts \(trs = \{c_1, \ldots, c_\gamma\}\), \(c_{\gamma+1} = (c_{\gamma+1, c_{\gamma+1, c_{\gamma+1}, c_{\gamma+1}}})\), \(c_{\gamma+1} = (c_{\gamma+1, c_{\gamma+1, c_{\gamma+1}}})\), \(o_R^j\), and random coins which include \(r_1\) for generating \(c_{\gamma+1}\). Suppose \(R\) has authenticated the tag as \(T_i\), \(cred\) is computed as a tuple \(\{R, T_i, cred_1, cred_2, cred_3\}\), where \(cred_1 = r^*, cred_2 = S_{sk_R}(r^*)\), and \(cred_3 = \alpha_{i+1} \oplus G_{k^{l_\gamma}}(o_{R^j+1})\).

d) CredVeri(\(para^*, cred\)): Given a PoP credential \(cred = \{R, T_i, cred_1, cred_2, cred_3\}\) and the system’s public parameters \(para^*\), it outputs the verification result “1” if \(\mathcal{V}_{PK_R}(cred_1, cred_2) \land \mathcal{V}_{PK_T}(H(cred_2), cred_3) = 1\), and “0” otherwise, where \(PK_R\) and \(PK_T\) are retrieved from \(para^*\).

C. Security analysis on the generic construction

\(\pi^*(R, T_i)\) can be considered as a combination of two protocols, including the underlying protocol \(\pi(R, T_i)\) and a new protocol \(\pi'(R, T_i)\). \(\pi'(R, T_i)\) is a two-round mutual authentication protocol. \(R\) and \(T_i\)’s share a secret key \(k_{T_i}^\prime \in R \{0, 1\}^{l_k}\) and a string \(trs \in \{0, 1\}^{l_{\gamma+1}}\), where \(l_\gamma\) is the length of \(\pi\)’s transcripts. \(R\) also has \(T_i\)’s public key \(PK_{T_i}\). \(\pi^*\) runs as the following.

1) The reader \(R\) chooses \(r^* \in \{0, 1\}^{l_{\gamma+1}}\), computes \(c' = H(S_{PK_R}(r^*))\) and \(c'' = G_{k^{l_{\gamma+1}}}(H(trs)||c')\). Then it sends \(c = (c', c'')\) to \(T_i\).

2) Upon receiving \(c = (c', c'')\), \(T_i\) computes \(\alpha' = G_{k_{T_i}^\prime}(c') \oplus S_{sk_{T_i}^\prime}(c')\) and \(\alpha'' = G_{k_{T_i}^\prime}(S_{sk_{T_i}^\prime}(c'))\), sends \(\alpha = (\alpha', \alpha'')\) to \(R\), and outputs \(o_{R^j} = 1\); else, outputs \(o_{R^j} = 0\).

3) Upon receiving \(\alpha = (\alpha', \alpha'')\), \(R\) computes \(\sigma = \alpha' \oplus G_{k_{T_i}^\prime}(c'')\) first. Then if \(\mathcal{V}_{PK_T}(c', \sigma) = 1\) and \(\alpha'' = G_{k_{T_i}^\prime}(\sigma)\), the reader \(R\) outputs \(o_{R^j} = 1\); else, outputs \(o_{R^j} = 0\).

In \(\pi^*(R, T_i)\), \(R\) and \(T_i\) first run the protocol \(\pi\)'s procedures to authenticate \(T_i\). After receiving \(\alpha_i\), if \(R\) considers \(T_i\) as a valid tag, say \(T_j\) (\(T_j = T_i\) if the authentication is correct, else \(T_j \neq T_i\)), it sets the value of \(trs\) as the concatenation of \(\pi\)'s protocol messages, and continues to run \(\pi^\prime\)'s procedures with \(T_j\) using \(T_j\)'s keys \(k_{T_j}^\prime\) and \(PK_{T_j}\).

We conduct security analysis on the protocol \(\pi'(R, T_i)\) and the combined protocol \(\pi^*(R, T_i)\), and have the following theorems (proof sketches are provided in Appendix).

**Theorem 1:** If the function family \(G : \{0, 1\}^{l_k} \times \{0, 1\}^{2l_{\gamma}} \rightarrow \{0, 1\}^{2l_{\gamma}}\) is a PRF family, and the underlying signature is complete, then \(\pi'(R, T_i)\) satisfies completeness, mutual authentication, and unp\#-privacy under the random oracle model.

**Theorem 2:** If \(\pi(R, T_i)\) satisfies tag/mutual authentication, \(\pi'(R, T_i)\) satisfies mutual authentication, both of \(\pi(R, T_i)\) and \(\pi'(R, T_i)\) satisfy adaptive completeness and unp\#-privacy, and the underlying signature is EU-CMA secure, then \(RS^* = (R, T, Setup^*, \pi^*, CredGen, CredVeri)\) satisfies adaptive completeness, mutual authentication, PoP credential unforgeability, and unp\#-privacy.

D. On the practicability of the generic construction

Unavoidably, our generic construction incurs additional requirements and runtime overheads. Among them, computational requirements and runtime overheads on the tag side are more critical.

In terms of computational capability, our generic construction requires tags to support hash, PRF, and signature scheme. Li et al. [28] pointed out that, to achieve unp\#-privacy, RFID tags at least should have the ability to compute a PRF or its equivalents such as symmetric block ciphers or cryptographic hash functions. This minimum condition applies to unp\#-privacy as well. A compact hash function with 128-bit output requires about 4,000 gate equivalents [52], and a compact AES 128-bit implementation requires 3,400 gate equivalents [53]. There exist passive RFID tags (e.g., [54], [55]) that support AES on the market.

Asymmetric cryptographic primitives normally are considered too heavy for low-cost RFID tags. Fortunately, ECC has been demonstrated to be usable in passive UHF RFID systems [56], [57]. Compared to other types of signature schemes with the same security level, ECC-based ones enjoy small key sizes and signature sizes, and better efficiency. It is feasible to adopt an ECC-based signature scheme in our construction due to the following reasons. First, efficient ECC processors have been proposed for passive ultra-high frequency (UHF) RFID tags. For example, Tan et al. [57] designed a 163-bit ECC processor (with 80-bit security) which requires around 12,000 gate equivalents. Second, in some ECC-based signature schemes, the expensive ECC point multiplication operations can be eliminated by pre-computing them. Third, there exist ECC-based signature schemes specifically designed for resource-constrained devices, such as Yavuz and Ozmen’s K-time ECC-based signature scheme SEMECS [58].

The modularized design of our generic construction makes the best use of existing tag/mutual authentication protocols, and simplifies the security analysis. In practice, it may not be necessary to generate a PoP credential in each and every run of the protocol. In such case, our construction can be easily implemented to support two running modes for better efficiency. In one mode, it only runs the underlying protocol \(\pi\)'s procedure for tag/mutual authentication if no PoP credential is required; in the other mode, it runs the full procedure if PoP credential is needed.

VI. AN RFID SYSTEM WITH MUTUAL AUTHENTICATION AND PoP

In this section, we refine the RFID mutual authentication protocol proposed by Li et al. in [28], and prove that the refined protocol is secure and satisfies unp\#-privacy. Then we construct a protocol to support PoP based on the refined protocol, and prove that the constructed protocol is secure and of unp\#-privacy under our framework.
A. A refined RFID mutual authentication protocol

Compared with the original RFID mutual authentication protocol proposed by Li et al. [28], our refined protocol is different in two aspects: (1) the reader R does not need to send a dummy third round message if the second round message from the tag is invalid; (2) both the reader and the tag explicitly output an execution result. The refined system $MA = (R, T, \text{Setup}, \pi)$ is defined below.

1) Setup$(\kappa, l)$: It initializes the whole system RS with a single legitimate reader R and l tags $T_1, l, l_0, l_d, l_p, l_r, u_0$ are all polynomial in the security parameter $\kappa$, where $l_r + l_p = l_d$ and $u_0 + l_r + l_d = l_d$. pad is a pre-fixed $l_p$-bit padding. Let $F : \{0, 1\}^k \times \{0, 1\}^u \rightarrow \{0, 1\}^l$ be a PRF family, $ctr \in \{0, 1\}^{l_1}$ be a counter. Each tag $T_i \in T$, $1 \leq i \leq l$, is assigned with an unique identity $ID_i$, a secret-key $k_i \in R \{0, 1\}^{l_1}$. A counter $ctr_i$ with initial value 1. For each tag $T_i \in T$, the reader R pre-computes an initial index $I_i = F_k(ctr_i||pad)$, and stores $rcd_i = (I_i, k_i, ctr_i, ID_i)$ in its database $DB$. The whole system’s public parameters are $para = \{\kappa, l, l_0, l_d, l_p, l_r, u_0, F, pad\}$.

2) Protocol $\pi(R, T_i)$: The protocol runs as below.

- R chooses $r_R \in R \{0, 1\}^{l_1}$, and sends $c_1 = r_R$ to $T_i$.
- Upon receiving $c_1$ from $T_i$, $R$ chooses $r_T \in R \{0, 1\}^{l_1}$, computes $c_{11} = F_k(ctr_i||pad)$ and $c_{12} = F_k(c_1||\alpha_{11}||\alpha_{12}||ctr_i)$, and sets $α_{12} = r_T$. Then $T_i$ sends $\alpha_1 = \{\alpha_{11}, \alpha_{12}, \alpha_{13}\}$ to $R$, and updates its counter $ctr_i = ctr_i + 1$.
- After receiving $\alpha_1 = \{\alpha_{11}, \alpha_{12}, \alpha_{13}\}$, $R$ authenticates $T_i$ through the following steps.
  - Step 1: $R$ searches its database to find a record $rcd = (I, k, ctr, ID, PK_T)$ such that $I = c_{11}$ and $c_{12} = F_k(c_1||\alpha_{11}||\alpha_{12}||ctr)$. If it discovers one, it updates $ctr = ctr + 1$ and $I = F_k(ctr||pad)$, and proceeds to Step 3; else, proceeds to Step 2.
  - Step 2: $R$ searches its database to find a record $rcd = (I', k, ctr, ID, PK_T)$ such that $I' = c_{11}$ and $c_{12} = F_k(c_1||\alpha_{11}||\alpha_{12}||ctr)$. If it discovers one, it updates $ctr = ctr + 1$ and $I' = F_k(ctr||pad)$, and proceeds to Step 3; else, outputs $\sigma_R = 0$.
  - Step 3: $R$ computes $c_2 = F_k(c_1||ctr_i||\alpha_{12})$, sends $c_2$ to $T_i$, and outputs $\sigma_T = 1$; else, outputs $\sigma_T = 0$.
- Upon receiving $c_2$, if $c_2 = F_k(c_1||ctr_i||\alpha_{12})$, $T_i$ outputs $\sigma_T = 1$; else, outputs $\sigma_T = 0$.

We summarize the security analysis results on the protocol $\pi$ with Theorem 3 while omitting the proofs, as they are similar to the ones provided in [28, 32].

Theorem 3: If the function family $F : \{0, 1\}^k \times \{0, 1\}^u \rightarrow \{0, 1\}^l$ is a PRF family, then the RFID system $MA = (R, T, \text{Setup}, \pi)$ defined above satisfies adaptive completeness, mutual authentication, and unp$^\#$-privacy.

B. An RFID mutual authentication protocol with PoP

We now construct a system $MAPP = (R, T, \text{Setup}^*, \pi^*, \text{CredGen}, \text{CredVeri})$ with PoP based on the system $MA = (R, T, \text{Setup}, \pi)$. Below we show the additions to RS.

1) Setup$^*(\kappa, l)$: We add a cryptographic hash function $H : \{0, 1\}^k \rightarrow \{0, 1\}^{l_r}$, another PRF family $G : \{0, 1\}^{l_k} \times \{0, 1\}^{2l_r} \rightarrow \{0, 1\}^{2l_r}$, and a signature scheme $DS$. Setup$^*(\kappa, l)$ is the same as defined in Section V-B.

2) Protocol $\pi^*(R, T_i)$: The protocol is depicted in Figure 7 and runs as follows.

- R chooses $r_{R1} \in R \{0, 1\}^{l_k}$, $r_{R2} \in R \{0, 1\}^{l_r}$ and sends $c_1 = r_{R1}$ to $T_i$.
- Upon receiving $c_1$, the tag $T_i$ chooses $r_T \in R \{0, 1\}^{l_1}$, computes $\alpha_{11} = F_k(ctr_i||pad)$, $\alpha_{12} = r_T$, and $\alpha_{13} = F_k(c_1||\alpha_{11}||\alpha_{12}||ctr)$. Then $T_i$ sends $\alpha_1 = \{\alpha_{11}, \alpha_{12}, \alpha_{13}\}$ to the reader $R$, and updates its counter $ctr_i = ctr_i + 1$.
- After receiving $\alpha_1 = \{\alpha_{11}, \alpha_{12}, \alpha_{13}\}$, the reader $R$ authenticates the tag through the following steps.
  - Step 1: R searches its database to find a record $rcd = (I, k, ctr, ID, PK_T)$ such that $I = \alpha_{11}$ and $\alpha_{12} = c_{12} = F_k(ctr||pad)$, and discovers one, it updates $ctr = ctr + 1$ and $I = F_k(ctr||pad)$, and proceeds to Step 3; else, proceeds to Step 2.
  - Step 2: R searches its database to find a record $rcd = (I', k, ctr, ID, PK_T)$ such that $I' = \alpha_{11}$ and $\alpha_{12} = c_{12} = F_k(ctr||pad)$.
  - Step 3: R computes $c_2 = F_k(c_1||ctr_i||\alpha_{12})$, and $c_{22} = G_k(H(c_1||\alpha_{12}||\alpha_{22}))$. Then $R$ sends $c_2 = (c_21, c_22, c_23)$ to $T_i$.
  - Upon receiving $c_2$, if $c_21 = F_k(c_1||ctr_i||\alpha_{12})$ and $c_{22} = G_k(H(c_1||\alpha_{12}||\alpha_{22}))$, $T_i$ computes $\alpha_{21} = G_k^{c_2}(c_{22})$ and $\alpha_{22} = G_k^{c_2}(S_k^{\alpha_{12}}(\alpha_{22}))$. Then $T_i$ sends $\alpha_2 = (\alpha_{21}, \alpha_{22})$ to the reader $R$, and outputs $\sigma_{R1} = 1$; otherwise it outputs $\sigma_{R1} = 0$.
  - Upon receiving $c_2$, if $\sigma_{R1} = 1$, $\alpha_{21} = G_k^{c_2}(c_{22}) = 1$ and $\alpha_{22} = c_{22}$, $R$ outputs $\sigma_R = 1$; else, outputs $\sigma_R = 0$.

CredGen(para, skR, sR, DB): Suppose the reader R has finished its j-th session of the protocol $\pi^*$, and identified the tag as $T_i$. CredGen computes a PoP credential $cred = (R, T_i, r_{R2}, S_k^{\alpha_{12}}(r_{R2}), \alpha_{21} + G_k^{\alpha_{12}}(c_{22}))$, where $r_{R2}$, $\alpha_{12}$, and $\alpha_{2}$ are stored in $s_R^i$, and $k_i$ comes from $T_i$’s record $rcd_i \in DB$.

CredVeri(para, cred): Given a PoP credential $cred = (R, T_i, c_{12}, c_{13}, c_{14}, c_{23}, c_{24})$ and the system’s public parameters $para$, it outputs the verification result “1” if $\forall_{PK_T}(c_{12}, c_{13}) = 1 \land V_{PK_T}(H(c_{23}, c_{24})) = 1$, and outputs “0” in any other cases.

We summarize the security analysis results on the RFID system $MAPoP$ with Theorem 4(proof sketches are provided in Appendix).

Theorem 4: If both the function family $F : \{0, 1\}^k \times \{0, 1\}^u \rightarrow \{0, 1\}^l$ and the function family $G : \{0, 1\}^k \times \{0, 1\}^{2l_r} \rightarrow \{0, 1\}^{2l_r}$ are PRF families, and the signature scheme $DS$ is complete and satisfies EU-CMA, then $MAPoP$ satisfies adaptive completeness, mutual authentication, PoP credential unforgeability, and unp$^\#$-privacy under the random oracle model.
We consider the implementation for MAPoP with 128-bit security. SchnorrQ [59] is a digital signature scheme that is based on the well-known Schnorr signature scheme [49] combined with the use of the elliptic curve FourQ [60]. SchnorrQ offers extremely fast, high-security digital signatures targeting the 128-bit security level. We also consider SEMECS [58], a $K$-time ECC-based signature scheme that is more efficient compared to its counterparts. It achieves optimal signature and private key sizes, and requires constant-size signatures at most for each secret key, and the public key's size is linear to $K$.

$K$-time signature scheme satisfies completeness and $K$-time EU-CMA defined in [58].

We denote SEMECS using FourQ as SEMECSQ. We select SchnorrQ and SEMECSQ as candidate signature schemes for MAPoP. We select BLAKE3 [61] to instantiate the PRF families $F$ and $G$ and serve as the underlying hash function for MAPoP due to its high efficiency and high security. BLAKE3 is a 128-bit secure cryptographic hash which can operate in three modes with a single algorithm. We use hash mode and keyed_hash mode in our implementation.

We consider three different kinds of instantiations for MAPoP, denoted as IMP1, IMP2, and IMP3 respectively. They are all based on BLAKE3, but differ in the selection/implementation of the signature scheme. In IMP1, both the reader and the tags adopt SchnorrQ; and the tags need to conduct ECC point multiplication operations when generating signatures. In IMP2, both the reader and the tags adopt SchnorrQ; and the tags use pre-computed $(r,rP)$ pairs for generating signatures. In IMP3, the reader adopts SchnorrQ, while the tags adopt SEMECSQ which does not require any ECC operation for signing a message. We provide performance analysis of IMP1, IMP2, IMP3, and $MA$ (without PoP). The performances of $MA$ based on BLAKE3 serve as a baseline for the performances of IMP1, IMP2, and IMP3.

1) Storage requirement: For the reader, we only list the size of each tag’s record in the database. In MAPoP, the reader stores $rcd = \{I_i,k_T,k'_T,ctr_i,ID_i,PK_T\}$ for each tag $T_i$; and $T_i$ stores $\{T'_i,k'_T,ctr_i,sk_T\}$. In MA, each reader stores $rcd = \{I_i,k_T,ctr_i,ID_i\}$ for each tag $T_i$; and $T_i$ stores $\{k_T,ctr_i\}$. SEMECSQ’s secret key size is 32 bytes, public key size is 32 bytes, and signature size is 64 bytes. The $K$-time signature scheme SEMECSQ’s secret key size is 32 bytes, signature size is 32 bytes, and public key size is 64 + $64K + |K|$ bytes. BLAKE3 takes an input of any byte length in $[0,2^{64}]$ (and together with a 32-byte key in keyed_hash mode only), and by default generates a 32-byte output. The output can be extended to any byte length in $[0,2^{64}]$. According to BLAKE3’s parameter settings, $k_T$, $k'_T$, $I_i$, and $ctr_i$ are all 32 bytes. We set $ID_i$ as 32 bytes, as it is the maximum length of an electronic product code (EPC) [59]. In IMP2, we assume that a tag stores $K$ pairs of pre-computed values $(r,rP)$. The storage requirements are shown in Table II. For IMP2 and IMP3, we set $K = 2^{17}$, which allows a tag to run a full protocol session in every 20 minutes for 5 years without updating pre-computed values/secret key. Note that, both the reader and the tags need to store the value of $K$ and maintain a counter for the signatures which require $2|K|$ bytes, while we omit to count them, as $|K|$ only takes three bytes.

2) Communication requirement: The communication overheads are also shown in Table II. IMP3’s fourth round message is 32 bytes shorter than IMP1’s and IMP2’s due to the

4 $P$ is a point on FourQ with order $t$, and $r \in R [1,t]$. 

---

![Fig. 7: An RFID Mutual Authentication Protocol With PoP](image-url)
The following reason. The forth round message of $\pi^*$ consists of two parts, $\alpha_{21}$ and $\alpha_{22}$, where $\alpha_{21} = G_{K_1'}(e_{23}) \oplus S_{\text{sk}_T}(e_{23})$. In IMP1 and IMP2, because Schnorr’s signature size is 64 bytes, then $G_{K_1'}$ is instantiated using BLAKE3 with an extended 64-byte output and $\alpha_{21}$ is 64 bytes. While in IMP3, because SEMECS’s signature size is 32 bytes, then $G_{K_1'}$ is instantiated using BLAKE3 with a default 32-byte output and $\alpha_{21}$ is 32 bytes.

3) Computational requirement: Let $\text{eMul}$, $\text{eAdd}$, $\text{mMul}$, and $H$ denote an ECC point multiplication operation, an ECC point addition operation, a modular multiplication operation, and a hash function operation respectively; and $T_{eMul}$, $T_{eAdd}$, $T_{mMul}$, and $T_H$ denote their running time respectively. Generally, an ECC point multiplication operation requires much more time than other operations. For example, under the settings of [62], $T_{eMul} \approx 3,333T_H$, $T_{eAdd} \approx 14T_H$, and $T_{mMul} \approx 2.7T_H$.

Sync denotes that the reader is synchronized with a tag, and can identify the tag using the index $I$. Desync denotes that the reader is de-synchronized with a tag, so that it needs to search the whole database with $l$ records for identifying the tag, where $l$ is the number of tags. We omit cheap operations if there exists at least one expensive one. In Schnorr’s, generating a signature incurs one $H$ and one $\text{mMul}$, and verifying a signature incurs two $\text{eMul}$. In SEMECS, generating a signature incurs three $H$ and one $\text{mMul}$, and verifying a signature incurs two $\text{eMul}$. The computational overheads of IMP1, IMP2, IMP3, and $MA$ are shown in Table II.

| | IMP1 | IMP2 | IMP3 | $MA$ |
|---|---|---|---|---|
| **storage requirement in terms of bytes** | | | | |
| $R$ | 192 | 192 | $\approx$ 8 MB | 96 |
| $T_r$ | 128 | $\approx$ 8 MB | 128 | 64 |
| **message size in terms of bytes** | | | | |
| 1st round | 32 | 32 | 32 | 32 |
| 2nd round | 96 | 96 | 96 | 96 |
| 3rd round | 96 | 96 | 96 | 32 |
| 4th round | 96 | 96 | 64 | - |

4) Discussions: None of IMP1, IMP2, and IMP3 outperforms the other two in all aspects. IMP1 has the highest requirements on tag computational capability. Although Tan et al. proposed a 163-bit ECC processor [57] suitable for UHF RFID tags, it only supports 80-bit security. There are efficient hardware implementations for 256-bit ECC (e.g., [63], [64]), however, it may still require further development to make them applicable for RFID tags. IMP1 also has the highest computational overheads on the tag side due to the expensive $\text{eMul}$. Tan et al. showed that their ECC processor supports the running of their authentication protocol up to 1,500 sessions per minute. As both their authentication protocol and IMP1 require one $\text{eMul}$ on the tag side in each session, we estimate that our generic construction (with 80-bit security) can be executed with comparable overhead if such ECC processor is implemented on the tag side.

Among IMP1, IMP2, and IMP3, IMP2 has the highest requirement on tag storage. Each pair of $(r, rP)$ takes 64 bytes. Besides leaving 96 bytes for storing keys, a tag with 2048-byte user memory (e.g., [55]) can only store 30 pairs of $(r, rP)$. As each session of $\pi^*$ consumes one pair of $(r, rP)$, 30 pairs may not be always sufficient in practice.

Among IMP1, IMP2, and IMP3, IMP3 has the highest requirement on reader storage. If $K = 2^{17}$, it takes 8 TB for the reader to store one million tags’ public keys. It is not difficult to equip the reader’s back-end server to match this storage requirement. Since IMP3 does not involve any expensive operation on the tag side, we consider IMP3 as the most applicable instantiation of $MAPoP$ among the three options.

VII. SUMMARY AND FUTURE WORKS

Proof of possession (PoP) is a property critical for many RFID-enabled applications, especially for RFID data-driven applications. We are the first to propose a formal framework for RFID tag/mutual authentication with PoP. We provided a secure and privacy-preserving generic construction in our framework based on any secure and unpoP-privacy-preserving tag/mutual authentication protocol, an EU-CMA secure signature scheme, a cryptographic hash function, and a PRF. We also proposed a concrete construction for RFID mutual authentication protocol with PoP, conducted security analysis, and provided a practical implementation solution. Potential future research directions include exploring other (stronger or weaker) formal frameworks for RFID tag/mutual authentication with PoP and designing RFID tag/mutual authentication protocols that support PoP with better practicality.

TABLE II: Comparison on Computational Overheads

| | IMP1 | IMP2 | IMP3 | $MA$ |
|---|---|---|---|---|
| $R$ (Sync) | 3eMul | 3eMul | 3eMul | 3H |
| $R$ (Desync) | $O(I) : H$ | $O(I) : H$ | $O(I) : H$ | $O(I) : H$ |
| $T_e$ | $\text{eMul}$ | $8H$ | $10H$ | $3H$ |

TABLE III: Comparison on Computational Overheads

REFERENCES

[1] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-channel(s). In *CHES*, pages 29–45, Redwood Shores, USA, 2002.
[2] Y. Li, R. H. Deng, and E. Bertino. RFID Security and Privacy. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool Publishers, 2013.
[3] A. Juels, D. Molner, and D. Wagner. Security and Privacy Issues in E-passports. In *SecureComm*, pages 74–88, Athens, Greece, 2005.
[4] H. Chien and C. Chen. Mutual Authentication Protocol for RFID Conforming to EPC Class 1 Generation 2 Standards. Comput. Stand. Interfaces, vol. 29, no. 2, pages 254–259, 2007.
[5] T. Yeh, Y. Wanga, T. Kuo, and S. Wang. Securing RFID Communications in Supply Chains. In *CHES*, pages 7678–7683, 2010.
[6] O. Molnar and D. Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In *CCS*, pages 210–219, New York, USA, 2004.
[7] Y. Li and X. Ding. Protecting RFID Communications in Supply Chains. In *ASIACCS*, pages 234–241, New York, USA, 2007.
[8] B. Song and C. J. Mitchell. RFID Authentication Protocol for Low-Cost Tags. In *WiSec*, pages 140–147, Alexandria, USA, 2008.
[9] S. Cai, Y. Li, T. Li, and R. H. Deng. Attacks and Improvements to an RFID Mutual Authentication Protocol and its Extensions. In *WiSec*, pages 51–58, Zurich, Switzerland, 2009.
