A New Framework for RFID Privacy

Robert H. Deng, Yingjiu Li
SIS, Singapore Management University

Andrew C. Yao
ITCS, Tsinghua University

Moti Yung
Google Inc. and Columbia University

Yunlei Zhao*
Software School, Fudan University

Abstract

Formal RFID security and privacy frameworks are fundamental to the design and analysis of robust RFID systems. In this paper, we develop a new definitional framework for RFID privacy in a rigorous and precise manner. Our framework is based on a zero-knowledge (ZK) formulation [7, 5] and incorporates the notions of adaptive completeness and mutual authentication. We provide meticulous justification of the new framework and contrast it with existing ones in the literature. In particular, we prove that our framework is stronger than the ind-privacy model of [14], which answers an open question posed in [14] for developing stronger RFID privacy models. Along the way we also try to clarify certain confusions and rectify several defects in the existing frameworks.

Based on the protocol of [16], we propose an efficient RFID mutual authentication protocol and analyze its security and privacy. The methodology used in our analysis is of independent interest and can be applied to analyze other RFID protocols within the new framework.

1 Introduction

Radio Frequency IDentification (RFID) tags are low-cost electronic devices, from which the stored information can be collected by an RFID reader efficiently (from tens to hundreds of tags per second) at a distance (from several centimeters to several meters) without the line of sight [20]. RFID technology has been widely used in numerous applications, ranging from manufacturing, logistics, transportation, warehouse inventory control, supermarket checkout counters, to many emerging applications [1]. As a key component of future ubiquitous computing environment, however, RFID technology has triggered significant concerns on its security and privacy as a tag's information can be read or traced by malicious readers from a distance without its owner's awareness [14, 11, 22, 13, 15, 4, 12].

It is critical to investigate formal RFID security and privacy frameworks that are fundamental to the design and analysis of robust RFID systems [14, 3, 21, 19, 8, 17, 16, 18]. However, due to high system complexity, it turns out to be full of subtleties in developing rigorous and precise RFID system models. By examining the existing RFID system models, in this paper we develop a new definitional framework for RFID security and privacy in a rigorous and precise manner. Our framework is based on a zero-knowledge formulation [7, 5], and incorporates the notions of adaptive completeness and mutual authentication. Compared to existing frameworks, very briefly, our framework is more reasonable in practice than those of [8, 16], and is stronger in terms of privacy than those of [14, 3]. Along the way, we also clarify certain confusions and rectify several defects in the existing frameworks.

To show how this new framework can be applied, we design an efficient RFID mutual authentication protocol based on the RFID protocol of [16] and analyze its security and privacy. The methodology used in our analysis is of independent interest and can be applied to analyze other RFID protocols within the new framework.

2 Preliminaries

If $A(\cdot,\cdot,...)$ is a randomized algorithm, then $y \leftarrow A(x_1, x_2, ...; \rho)$ means that y is assigned with the unique output of the algorithm A on inputs $x_1, x_2, ...$ and coins ρ, while $y \leftarrow A(x_1, x_2, ...)$ is a shorthand for first picking ρ at random and then setting $y \leftarrow A(x_1, x_2, ...; \rho)$. Let $y \leftarrow A^{O_1,...,O_n}(x_1, x_2, ...)$ denote that y is assigned with the output of the algorithm A which takes $x_1, x_2, ...$ as inputs and has oracle accesses to $O_1, ..., O_n$. If S is a set, then $s \in_R S$ indicates that s is chosen uniformly at random from S. If $x_1, x_2, ...$ are strings, then $x_1||x_2||\cdots$ denotes the concatenation of them. If x is a string, then $|x|$ denotes its bit length in binary code. If S is a set, then $|S|$ denotes

*Contact author: ylzhao@fudan.edu.cn

its cardinality (i.e. the number of elements of S). Let Pr[E] denote the probability that an event E occurs, $\mathcal{N}$ denote the set of all integers, $\mathcal{R}$ denote the set of all real numbers.

A function $f : \mathcal{N} \rightarrow \mathcal{R}$ is said to be *negligible* if for every $c > 0$ there exits a number $m \in \mathcal{N}$ such that $f(n) < \frac{1}{n^c}$ holds for all $n > m$.

Given a security parameter κ, let $m(\cdot)$ and $l(\cdot)$ be two positive polynomials in κ. We say that $\{F_k : \{0,1\}^{m(\kappa)} \longrightarrow \{0,1\}^{l(\kappa)}\}_{k \in_R \{0,1\}^\kappa}$ is a pseudorandom function (PRF) ensemble according to the definition given by Goldreich, Goldwasser, and Micali [6] (see also appendix A for detail).

3 Model of RFID Systems

In this section, we first give a formal description of RFID system setting and adversary. We then define RFID systems to be "complete" in term of *adaptive completeness*, and "sound" in terms of *mutual authentication*.

3.1 RFID System Setting

Consider an RFID system comprising of a single legitimate reader[1] R and a set of ℓ tags $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_\ell\}$, where ℓ is a polynomial in a security parameter κ. The reader and the tags are probabilistic polynomial time (PPT) interactive Turing machines. The RFID system $(R, \mathcal{T})$ is setup by an procedure, denoted $\mathsf{Setup}(\kappa, \ell)$. Specifically, on (κ, ℓ), this setup procedure generates the public system parameter σ_R, the reader secret-key k_R and initial internal state s_R^1 (if needed) for the reader R. It may also setup an initial database DB^1 for the reader R to store necessary information for identifying and authenticating tags. For each i, $1 \leq i \leq \ell$, this procedure generates the public parameter $\xi_{\mathcal{T}_i}$ and the initial secret-key $k_{\mathcal{T}_i}^1$ for a tag $\mathcal{T}_i$ and sets the tag's initial internal state $s_{\mathcal{T}_i}^1$ (typically, $s_{\mathcal{T}_i}^1$ includes the public parameters $\sigma_R, \xi_{\mathcal{T}_i}$). It may also associate the tag $\mathcal{T}_i$ with its unique ID, as well as other necessary information such as tag key and/or tag state information, as a record in the initial database DB^1 of the reader R. Note that $\xi_{\mathcal{T}_i}$ or/and $s_{\mathcal{T}_i}^1$ can be empty strings.

We use $para = (\sigma_R, \xi_1, \cdots, \xi_\ell)$ to denote the public system parameters. We assume that in the RFID system, the reader is secure; in other words, the legitimate reader is a "black-box" to an adversary.

Every tag $\mathcal{T}_i$, $1 \leq i \leq \ell$, exchanges messages with the reader R through a protocol $\pi(R, \mathcal{T}_i)$. Without loss of generality, we assume the protocol run of π is always initiated by the reader R and the protocol π is of $2\gamma + 1$ rounds[2] for some $\gamma \geq 1$. Each protocol run of π is called a session. We assume each tag interacts with the reader sequentially, but multiple tags can interact with the reader "concurrently" (with some anti-collision protocols [23]). To allow and distinguish concurrent sessions (at the side of the reader R), we associate each session of protocol π with a unique session identifier sid. In practice, sid is typically generated by the reader when it is invoked to send the first-round message. We assume each message from a tag to the reader always bears the corresponding session-identifier. As each tag runs sessions subsequently, the session-identifiers are dropped from messages sent from the reader if no confusion arises.

Each tag $\mathcal{T}_i$, as well as the reader R, uses fresh and independent random coins (generated on the fly) in each session, *in case it is an randomized algorithm*. We assume that the random coins used in each session are erased once the session is completed (whether successfully finished or aborted). Also, in each session run, the tag may update its internal state and secret-key, and the reader may update its internal state and database. We assume that the update process of new internal state and secret-key by an uncorrupted tag automatically overwrites (i.e., erases) its old internal state and secret-key.

Given a security parameter κ, we assume that each tag $\mathcal{T}_i$ takes part in at most s (sequential) sessions in its life time[3] with R, and thus the reader R involves at most $s\ell$ sessions, where s is some polynomial in κ. In practice, the value s can be a fixed constant (e.g., $s = 2^{28}$ [1]).

More precisely, for the j-th session (ordered by the session initiation time) where $1 \leq j \leq s\ell$, the reader R takes the input from the system parameters $para$, its secret-key k_R, current internal state s_R^j, database DB^j, random coins ρ_R^j, and a partial transcript T, where T is either an empty string (which indicates the starting of a new session) or a sequence of messages $(sid, c_1, \alpha_1, c_2, \alpha_2, \cdots, c_u, \alpha_u)$, $1 \leq u \leq \gamma$ (which indicates the on-going of session sid). The reader R outputs the next message c_{u+1}. In the case of $T = (sid, c_1, \alpha_1, c_2, \alpha_2, \cdots, c_\gamma, \alpha_\gamma)$, besides sending back the last-round message $c_{\gamma+1}$, the reader R also updates its internal state to s_R^{j+1}, its database to DB^{j+1}, and stops the session by additionally outputting a bit, denoted by o_R^{sid}. This last bit output indicates either acceptation ($o_R^{sid} = 1$) or rejection ($o_R^{sid} = 0$) of the current session.

[1] It is straightforward to extend the model to include multiple legitimate readers. Notice that an adversary can use its own readers to interact with tags.

[2] For protocols of even 2γ rounds with the last-round message sent by the tag, we can define, by default, the $(2\gamma + 1)$-th round (from the reader to the tag) to be the output of R that indicates acceptation or rejection of the protocol run. Also, without loss of generality, we assume R and $\mathcal{T}_i$ exchange some system public parameters in the first two rounds.

[3] It is assumed that s is large enough so that any tag can never run up to s sessions in its life time; otherwise, an adversary may distinguish two tags, thus violate their privacy, by running one tag more than s times while the other less than s times [14].

Without loss of generality, we assume that the j-th session run by reader R corresponds to the v-th session run by tag $\mathcal{T}_i$, where $1 \leq v \leq s$ and $1 \leq i \leq \ell$. In this session, the tag $\mathcal{T}_i$ takes the input from the system parameters $para$, its current secret-key $k^v_{\mathcal{T}_i}$, current internal state $s^v_{\mathcal{T}_i}$, random coins $\rho^v_{\mathcal{T}_i}$, and a partial transcript $T = (sid, c_1, \alpha_1, \cdots, \alpha_{u-1}, c_u)$, where $1 \leq u \leq \gamma$. The tag $\mathcal{T}_i$ outputs the next message (sid, α_u), where sid is the session-identifier of the v-th session run by the tag. In the case of $T = (sid, c_1, \alpha_1, \cdots, c_\gamma, \alpha_\gamma, c_{\gamma+1})$ (i.e., $\mathcal{T}_i$ has received the last-round message of the session sid), $\mathcal{T}_i$ updates its internal state to $s^{v+1}_{\mathcal{T}_i}$, its secret-key to be $k^{v+1}_{\mathcal{T}_i}$, and stops the session by additionally outputting a bit, denoted by $o^{sid}_{\mathcal{T}_i}$. This last bit output indicates either acceptation ($o^{sid}_{\mathcal{T}_i} = 1$) or rejection ($o^{sid}_{\mathcal{T}_i} = 0$) of the current session run by $\mathcal{T}_i$.

Note that in the above description, it is assumed that the reader and tags update their internal states, database, or keys *at the end of each protocol run*. In reality, this can be performed at any point of each protocol run. Also, for RFID protocol π with unidirectional authentication from tag to reader, the tag may not have a session output. In this case, the session output $o^{sid}_{\mathcal{T}_i}$ is set to be "0" always.

3.2 Adversary

After an RFID system $(R, \mathcal{T})$ is setup by invoking the setup procedure $\mathsf{Setup}(\kappa, \ell)$, we model a probabilistic polynomial-time concurrent man-in-the-middle (CMIM) adversary $\mathcal{A}$ against $(R, \mathcal{T})$, with adaptive tag corruption. We use $\hat{m}$ to denote a message sent by adversary $\mathcal{A}$, and m to denote the actual message sent by reader R or an uncorrupted tag. The adversary is given access to the following oracles:

InitReader(): $\mathcal{A}$ invokes the reader R to start a session of protocol π and generate a session identifier sid as well as the first-round message c_1. Supposing that the new session is the j-th session run by R, the reader R stores (sid, c_1) into its internal state s^j_R, and returns c_1 to the adversary.

SendT($\mathcal{T}_i, \hat{m}$): Adversary $\mathcal{A}$ sends $\hat{m}$ to $\mathcal{T}_i$[4]. After receiving $\hat{m}$, $\mathcal{T}_i$ works as follows:

- If $\mathcal{T}_i$ currently does not run any existing session, $\mathcal{T}_i$ initiates a new session with the session-identifier sid set to $\hat{m}$, treats $\hat{m}$ as the first-round message of the new session, and returns back the second-round message (sid, α_1).
- If $\mathcal{T}_i$ is currently running an incomplete session with session-identifier $sid = \hat{c}$, and is waiting for the u-th message from R, where $u \geq 2$, $\mathcal{T}_i$ works as follows: If $2 \leq u \leq \gamma$, it treats $\hat{m}$ as the u-th message from the reader and returns back the next round message (sid, α_u). If $u = \gamma + 1$ (i.e., $\mathcal{T}_i$ is waiting for the last-round message of the session sid), $\mathcal{T}_i$ returns back its output $o^{sid}_{\mathcal{T}_i}$ to the adversary, and (internally) updates its internal state to be $s^{v+1}_{\mathcal{T}_i}$, assuming that the session sid is the v-th session run by $\mathcal{T}_i$, where $1 \leq v \leq s$,.

SendR($\widehat{sid}, \hat{\alpha}$): Adversary $\mathcal{A}$ sends $(\widehat{sid}, \hat{\alpha})$ to the reader R. After receiving $(\widehat{sid}, \hat{\alpha})$, R checks from its internal state whether it is running a session of session identifier $sid = \widehat{sid}$, and works as follows:

- If R is currently running an incomplete session with $sid = \widehat{sid}$ and is waiting for the u-th message from a tag, where $1 \leq u \leq \gamma$, R acts as follows: If $u < \gamma$, it treats $\hat{\alpha}$ as the u-th message from the tag, and returns back the next round message c_{u+1} to $\mathcal{A}$. If $u = \gamma$, it returns back the last-round message $c_{\gamma+1}$ and the output o^{sid}_R to $\mathcal{A}$, and internally updates its internal state to be s^{j+1}_R and the database to be DB^{j+1}, assuming that the session sid corresponds to the j-th session run by R.
- In all other cases, R returns back a special symbol $\perp$ (indicating invalid query).

Corrupt($\mathcal{T}_i$): Adversary $\mathcal{A}$ obtains the secret-key and internal state information (as well as the random coins) currently held by $\mathcal{T}_i$. Once a tag $\mathcal{T}_i$ is corrupted, all its actions are controlled and performed by the adversary $\mathcal{A}$.

Let O_1, O_2, O_3 and O_4 denote the above oracles, respectively. These oracles fully capture the capability of any PPT CMIM adversary with adaptive tag corruption.[5] For presentation simplicity, we denote by $\mathcal{O}$ 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 works in time* t *and makes oracle queries to* O_μ *without exceeding* n_μ *times, where* $1 \leq \mu \leq 4$. We treat each oracle call as a unit operation, and thus for a t-time adversary it holds that $\Sigma^4_{\mu=1} n_\mu \leq t$.

We denote by $A^{\mathcal{O}}(R, \mathcal{T}, para)$ a PPT algorithm A that, on input of some system public parameters $para$, concurrently interacts with R and the tags in $\mathcal{T}$ via the four oracles O_1, O_2, O_3, O_4 specified above, where $(R, \mathcal{T})$ is setup by the procedure $\mathsf{Setup}(\kappa, \ell)$.

[4] For simplicity, we abuse the notation $\mathcal{T}_i$ to denote any virtual identity of a tag in $\mathcal{T}$ (not the tag's real identity) labeled by $\mathcal{A}$ when $\mathcal{A}$ selects the tag from $\mathcal{T}$.

[5] Here, for simpler definitional complexity, we assume all tags are always within the attack scope of adversary. In practice, some tags may be in or out from the attack scope of adversary at different time [21].

Note that in our formulation, the output bits of protocol participants (which indicate authentication success or failure) are *publicly* accessible to the adversary. The reason is that, in reality, such outputs can be publicly observed from the behaviors of protocol participants during/after the protocol run or can be learnt by some other side-channel information.

3.3 Adaptive Completeness and Mutual Authentication

We now define an RFID system to be complete in terms of protocol adaptive completeness, and sound in terms of protocol mutual authentication. We first define the concept of adaptive completeness. Roughly speaking, the adaptive completeness property of an RFID protocol π means that, after any potential (particularly, desynchronizing) attacks by the PPT CMIM adversary $\mathcal{A}$, if the reader R completes its j-th session and an uncorrupted tag $\mathcal{T}_i$ completes its v-th session such that these two sessions are of identical session-identifier sid and identical round messages, then with overwhelming probability the reader R successfully identifies $\mathcal{T}_i$ (i.e., $o_R^{sid} = 1$), and with overwhelming probability the tag $\mathcal{T}_i$ also identifies the reader (i.e., $o_{\mathcal{T}_i}^{sid} = 1$) if the protocol π is of mutual authentication.

Definition 3.1 (adaptive completeness) *For an RFID system $(R, \mathcal{T})$ setup by $\mathsf{Setup}(\kappa, \ell)$, denote by*

$$(sid, c_1^{sid}, \alpha_1^{sid}, \cdots, \alpha_\gamma^{sid}, c_{\gamma+1}^{sid}, o_R^{sid}, o_{\mathcal{T}_i}^{sid}) \leftarrow \pi(R, \mathcal{T}_i)$$

the running of a session with identifier sid of the protocol π between R and an uncorrupted tag $\mathcal{T}_i \in \mathcal{T}$. Suppose that the session sid corresponds to the v-th session at the side of $\mathcal{T}_i$ and the j-th session at the side of R, where $1 \leq v \leq s$ and $1 \leq j \leq s\ell$. Consider the case that the two sessions are of the same round messages, and that all the exchanged messages in these two (matching) sessions are all honestly generated by R and $\mathcal{T}_i$ respectively. Denote by E the event that $o_R^{sid} = 0$ holds (or $o_{\mathcal{T}_i}^{sid} = 0$ holds if the protocol π is of mutual authentication) or R identifies a different tag $\mathcal{T}_{i'} \neq \mathcal{T}_i$ in its j-th session.

A PPT CMIM adversary $\mathcal{A}$ $(t, \epsilon, n_1, n_2, n_3, n_4)$-breaks the adaptive completeness of the RFID system against the uncorrupted $\mathcal{T}_i$, if the probability that event E occurs with probability at least ϵ and $\mathcal{A}$ is a (t, n_1, n_2, n_3, n_4)-adversary. The probability is taken over the coins used by $\mathsf{Setup}(\kappa, \ell)$, the coins of $\mathcal{A}$, the coins used by R (up to finishing the j-th session), and the coins used by $\mathcal{T}_i$ (up to finishing the v-th session).

An RFID system $(R, \mathcal{T})$ is of adaptive completeness, if for all sufficiently large κ and for any uncorrupted tag $\mathcal{T}_i$, there exists no adversary $\mathcal{A}$ that can $(t, \epsilon, n_1, n_2, n_3, n_4)$-breaks the adaptive completeness against $\mathcal{T}_i$, for any (t, ϵ), where t is polynomial in κ and ϵ is non-negligible in κ.

Note that the above definition captures any adversarial desynchronizing attacks by $\mathcal{A}$, the adaptive evolution of internal state and secret-key of $\mathcal{T}_i$, and the adaptive evolution of the internal state, secret-key, and database of R.

Next, we define mutual authentication of RFID protocols. Roughly speaking, for an RFID protocol π of the RFID system $(R, \mathcal{T})$, authentication from reader to tag (resp., from tag to reader) means that an CMIM adversary $\mathcal{A}$ cannot impersonate the reader R (resp., an uncorrupted tag $\mathcal{T}_i \in \mathcal{T}$) to an uncorrupted tag $\mathcal{T}_i \in \mathcal{T}$ (resp., reader R), unless $\mathcal{A}$ honestly relays messages actually generated and sent by R and the uncorrupted tag $\mathcal{T}_i$. Before we define mutual authentication for RFID protocols, we first clarify the notion of matching sessions.

Definition 3.2 (matching sessions) *Denote by $(sid, c_1^{sid}, \alpha_1^{sid}, \cdots, \alpha_\gamma^{sid}, c_{\gamma+1}^{sid})$ the transcript of exchanged round messages (except the session outputs) of a* successfully completed *session sid of the protocol π run by a tag $\mathcal{T}_i$, where $1 \leq i \leq \ell$. This session has a matching session at the side of the reader R, if R ever successfully completed a session of the identical session transcript.*

Denote by $(sid', c_1^{sid'}, \alpha_1^{sid'}, \cdots, \alpha_\gamma^{sid'}, c_{\gamma+1}^{sid'})$ the transcript of exchanged round messages (except the session outputs) of a successfully completed *session sid' run by R. This session has a matching session at the side of some tag $\mathcal{T}_i$, where $1 \leq i \leq \ell$, if either of the following conditions holds:*

- *$\mathcal{T}_i$ ever* completed*, whether successfully finished or aborted, a session of the identical transcript* prefix *$(sid', c_1^{sid'}, \alpha_1^{sid'}, \cdots, \alpha_\gamma^{sid'})$;*
- *Or, $\mathcal{T}_i$ is now running a session with partial transcript $(sid', c_1^{sid'}, \alpha_1^{sid'}, \cdots, \alpha_\gamma^{sid'})$ and is waiting for the last-round message of the session sid'.*

For a successfully completed session run by a tag $\mathcal{T}_i$, its matching session is defined to be the successfully completed session with the identical session transcript at the side of R. However, for a successfully completed session run by the reader R with the session transcript $(sid', c_1^{sid'}, \alpha_1^{sid'}, \cdots, \alpha_\gamma^{sid'}, c_{\gamma+1}^{sid'})$, its matching session can be any session at the side of an uncorrupted tag $\mathcal{T}_i$: (1) a successfully finished session of the identical session transcript; (2) a completed but aborted session of the session transcript $(sid', c_1^{sid'}, \alpha_1^{sid'}, \cdots, \alpha_\gamma^{sid'}, \hat{c}_{\gamma+1}^{sid'})$, where $\hat{c}_{\gamma+1}^{sid'} \neq c_{\gamma+1}^{sid'}$; (3) an incomplete ongoing session with partial transcript $(sid', c_1^{sid'}, \alpha_1^{sid'}, \cdots, \alpha_\gamma^{sid'})$. This treatment is to take into account the following "cutting-last-message" attack: a CMIM adversary $\mathcal{A}$ relays the messages being exchanged by R and an uncorrupted tag $\mathcal{T}_i$ for a protocol run of π until receiving the last-round message $c_{\gamma+1}^{sid'}$ from R; after this, $\mathcal{A}$ sends an arbitrary message $\hat{c}_{\gamma+1}^{sid'} (\neq c_{\gamma+1}^{sid'})$ to

$\mathcal{T}_i$ (which typically causes $\mathcal{T}_i$ to abort the session), or, just drops the session at the side of $\mathcal{T}_i$ without sending $\mathcal{T}_i$ the last-round message. Such "cutting-last-message" attacks are unpreventable.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{auth}[\kappa, \ell]$
1. run $\mathsf{Setup}(\kappa, \ell)$ to setup the reader R and a set of tags $\mathcal{T}$; denote by $para$ the public system parameters;
2. $trans \leftarrow \mathcal{A}^{\mathcal{O}}(R, \mathcal{T}, para)$.

Figure 1. Authentication Experiment

Figure 1 shows the authentication experiment $\mathbf{Exp}_{\mathcal{A}}^{auth}[\kappa, \ell]$. A CMIM adversary $\mathcal{A}$ interacts with R and tags in $\mathcal{T}$ via the four oracles in $\mathcal{O}$; At the end of the experiment, $\mathcal{A}$ outputs the transcript, $trans$, of a session. Denote by E_1 the event that $trans$ corresponds to the transcript of a successfully completed session run by R in which R successfully identifies an *uncorrupted* tag $\mathcal{T}_i$, but this session has no matching session at the side of the uncorrupted tag $\mathcal{T}_i$. Denote by E_2 the event that $trans$ corresponds to the transcript of a successfully completed session run by some *uncorrupted* tag $\mathcal{T}_i \in \mathcal{T}$, and this session has no matching session at the side of R.

Definition 3.3 (authentication) *On a security parameter κ, an adversary $\mathcal{A}$ $(\epsilon, t, n_1, n_2, n_3, n_4)$-breaks the authentication of an RFID system $(R, \mathcal{T})$ against the reader R (resp., an uncorrupted tag $\mathcal{T}_i \in \mathcal{T}$) if the probability that event E_1 (resp., E_2) occurs is at least ϵ and $\mathcal{A}$ is a (t, n_1, n_2, n_3, n_4)-adversary.*

The RFID system $(R, \mathcal{T})$ is of tag-to-reader authentication (resp., reader-to-tag authentication), if for all sufficiently large κ there exists no adversary $\mathcal{A}$ that can $(\epsilon, t, n_1, n_2, n_3, n_4)$-break the authentication of $(R, \mathcal{T})$ against the reader R (resp., any uncorrupted tag $\mathcal{T}_i \in \mathcal{T}$), for any (t, ϵ), where t is polynomial in κ and ϵ is non-negligible in κ.

An RFID system is of mutual authentication, if it is of both tag-to-reader authentication and reader-to-tag authentication.

4 Zero-Knowledge Based RFID Privacy

In this section, we present a zero-knowledge based definitional framework for RFID privacy. To make our definition formal, we need to clarify the notion of blind access to tags and the notion of clean tags.

Let $A^{\mathcal{O}}(R, \widehat{\mathcal{T}}, \mathcal{I}(\mathcal{T}_g), aux)$ be a PPT algorithm A that, on input $aux \in \{0,1\}^*$ (typically, aux includes the system parameters or some historical state information of A), concurrently interacts with R and a set of tags $\widehat{\mathcal{T}}$ via the four oracles $\mathcal{O} = \{O_1, O_2, O_3, O_4\}$. We say that A has *blind access* to a *challenge* tag $\mathcal{T}_g \notin \widehat{\mathcal{T}}$ if A interacts with $\mathcal{T}_g$ via a special interface $\mathcal{I}$. Specifically, $\mathcal{I}$ is a PPT algorithm that runs $\mathcal{T}_g$ internally, and interacts with A externally. To send a message $\hat{c}$ to $\mathcal{T}_g$, A sends to $\mathcal{I}$ a special O_2 oracle query of the form $\mathsf{SendT}(challenge, \hat{c})$; after receiving this special O_2 query, $\mathcal{I}$ invokes $\mathcal{T}_g$ with $\mathsf{SendT}(\mathcal{T}_g, \hat{c})$, and returns back to A the output by $\mathcal{T}_g$. From the viewpoint of A, it does not know which tag it is interacting with. It is also required that A interacts with $\mathcal{T}_g$ via O_2 queries only.

Next, we define the notion of clean tags. A tag $\mathcal{T}_i$ is called *clean*, if it is not corrupted (i.e., the adversary has not made any O_4 query to $\mathcal{T}_i$), and is not currently running an incomplete session with the reader (i.e., the last session of the tag has been either finished or aborted). In other words, a clean tag is an uncorrupted tag that is currently at the status of waiting for the first-round message from the reader to start a new session.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{zkp}[\kappa, \ell]$
1. run $\mathsf{Setup}(\kappa, \ell)$ to setup the reader R and a set of tags $\mathcal{T}$; denote by $para$ the system public parameters;
2. $\{\mathcal{C}, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}}(R, \mathcal{T}, para)$, where $\mathcal{C} = \{\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, \cdots, \mathcal{T}_{i_\delta}\} \subseteq \mathcal{T}$ is a set of *clean* tags, $0 \leq \delta \leq \ell$;
3. $g \in_R \{1, \cdots, \delta\}$, set $\mathcal{T}_g = \mathcal{T}_{i_g}$ and $\widehat{\mathcal{T}} = (\mathcal{T} - \mathcal{C})$;
4. $view_{\mathcal{A}} \leftarrow \mathcal{A}_2^{\mathcal{O}}(R, \widehat{\mathcal{T}}, \mathcal{I}(\mathcal{T}_g), st)$;
5. output $(g, view_{\mathcal{A}})$.

Figure 2. zk-privacy experiment: real world

Now, we are ready to give a formal definition of zero-knowledge based RFID privacy (zk-privacy, for short). Figure 2 illustrates the real world of the zk-privacy experiment, $\mathbf{Exp}_{\mathcal{A}}^{zkp}[\kappa,\ell]$ ($\mathbf{Exp}_{\mathcal{A}}^{zkp}$, for simplicity), in which a PPT CMIM adversary $\mathcal{A}$ is comprised of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ and runs in two stages. In the *first stage*, algorithm $\mathcal{A}_1$ is concurrently interacting with R and all the tags in $\mathcal{T}$ via the four oracles in $\mathcal{O}$, and is required to output a set $\mathcal{C}$ of *clean* tags at the end of the first stage, where $\mathcal{C} \subseteq \mathcal{T}$ consists of δ *clean* tags, denoted as $\{\mathcal{T}_{i_1}, \cdots, \mathcal{T}_{i_\delta}\}$. The algorithm $\mathcal{A}_1$ also outputs a state information st, which will be transmitted to algorithm $\mathcal{A}_2$. Between the first stage and the second stage, a challenge tag, denoted as $\mathcal{T}_g$, is taken uniformly at random from $\mathcal{C}$. Note that if $\delta = 0$, then no challenge tag is selected, and $\mathcal{A}$ is reduced to $\mathcal{A}_1$ in this experiment. In the *second stage*, on input st, $\mathcal{A}_2$ concurrently interacts with the reader R and the tags in $\widehat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$ via the four oracles in $\mathcal{O}$, and additionally has blind access to $\mathcal{T}_g$. Note that $\mathcal{A}$ cannot corrupt any tag (particularly $\mathcal{T}_g$) in $\mathcal{C}$, and $\mathcal{A}$ does not have access to tags in $\mathcal{C} - \{\mathcal{T}_g\}$ in the second stage. Finally, $\mathcal{A}_2$ outputs its view, denoted by $view_{\mathcal{A}}$, at the end of the second stage. Specifically, $view_{\mathcal{A}}$

is defined to include the system public parameters $para$, the random coins used by $\mathcal{A}$, $\rho_{\mathcal{A}}$, and the (ordered) list of all oracle answers to the queries made by $\mathcal{A}$ in the experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$. Note that $view_{\mathcal{A}}$ does not explicitly include the oracle queries made by $\mathcal{A}$ and $\mathcal{A}$'s output at the first stage, as all these values are implicitly determined by the system public parameter $para$, $\mathcal{A}$'s coins and all oracle answers to $\mathcal{A}$'s queries. The output of experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$ is defined to be $(g, view_{\mathcal{A}})$. Denote by $(g, view_{\mathcal{A}}(\kappa, \ell))$ the random variable describing the output of experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}[\kappa, \ell]$.

Experiment $\mathbf{Exp}^{zkp}_{\mathcal{S}}[\kappa, \ell]$
1. run $\mathsf{Setup}(\kappa, \ell)$ to setup the reader R and a set of tags $\mathcal{T}$; denote by $para$ the system public parameters;
2. $\{\mathcal{C}, st\} \leftarrow \mathcal{S}_1^{\mathcal{O}}(R, \mathcal{T}, para)$, where $\mathcal{C} = \{T_{i_1}, T_{i_2}, \cdots, T_{i_\delta}\} \subseteq \mathcal{T}$ is a set of *clean* tags, $0 \leq \delta \leq \ell$;
3. $g \in_R \{1, \cdots, \delta\}$, and set $\widehat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$;
4. $sview \leftarrow \mathcal{S}_2^{\mathcal{O}}(R, \widehat{\mathcal{T}}, st)$, where $sview$ particularly includes all oracle answers to queries made by $\mathcal{S}$;
5. output $(g, sview)$.

Figure 3. zk-privacy experiment: simulated world

Figure 3 illustrates the simulated world of zk-privacy experiment, $\mathbf{Exp}^{zkp}_{\mathcal{S}}[\kappa,\ell]$ ($\mathbf{Exp}^{zkp}_{\mathcal{S}}$, for simplicity), in which a PPT simulator $\mathcal{S}$ is comprised of a pair of algorithms $(\mathcal{S}_1, \mathcal{S}_2)$ and runs in two stages. In the *first stage*, algorithm $\mathcal{S}_1$ concurrently interacts with R and all the tags in $\mathcal{T}$ via the four oracles in $\mathcal{O}$, and outputs a set, denoted $\mathcal{C}$, of *clean* tags, where $|\mathcal{C}| = \delta$ and $0 \leq \delta \leq \ell$. It also outputs a state information st, which will be transmitted to algorithm $\mathcal{S}_2$. Between the two stages, a value g is taken uniformly at random from $\{1, \cdots, |\mathcal{C}|\}$ (which is unknown to $\mathcal{S}$). In the *second stage* of $\mathcal{S}$, on input st, $\mathcal{S}_2$ concurrently interacts with the reader R and the tags in $\widehat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$, and outputs a simulated view, denoted $sview$, at the end of the second stage. We require that all oracle answers to the queries made by $\mathcal{S}$ (in both the first stage and the second stage) in the experiment $\mathbf{Exp}^{zkp}_{\mathcal{S}}$ are included in $sview$. The output of the experiment $\mathbf{Exp}^{zkp}_{\mathcal{S}}$ is defined to be $(g, sview)$. Denote by $(g, sview(\kappa, \ell))$ the random variable describing the output of the experiment $\mathbf{Exp}^{zkp}_{\mathcal{S}}[\kappa,\ell]$.

Informally, an RFID protocol π is of zk-privacy, if for any PPT CMIM adversary $\mathcal{A}$ there exists a polynomial-time simulator $\mathcal{S}$ such that for all sufficiently large κ and any ℓ which is polynomial in κ, the distributions of $(g, view_{\mathcal{A}}(\kappa, \ell))$ and $(g, sview(\kappa, \ell))$ are indistinguishable. In other words, what can be derived by interacting with the challenge tag $\mathcal{T}_g$ in the second-stage of $\mathcal{A}$ can actually be derived by $\mathcal{A}$ itself *without interacting with* $\mathcal{T}_g$. In this sense, the interaction between $\mathcal{A}_2$ and $\mathcal{T}_g$ leaks "zero knowledge" to the adversary $\mathcal{A}$. For this reason, our RFID privacy notion is named zk-privacy.

Definition 4.1 (zk-privacy) *An RFID protocol π is of computational (resp., statistical) zk-privacy, if for any PPT CMIM adversary $\mathcal{A}$ there exists a polynomial-time simulator $\mathcal{S}$ such that for all sufficiently large κ and any ℓ which is polynomials in κ (i.e., $\ell = poly(\kappa)$, where $poly(\cdot)$ is some positive polynomial), the following ensembles are computationally (resp., statistically) indistinguishable:*

- $\{g, view_{\mathcal{A}}(\kappa, \ell)\}_{\kappa \in N, \ell \in poly(\kappa)}$
- $\{g, sview(\kappa, \ell)\}_{\kappa \in N, \ell \in poly(\kappa)}$

In other words, for any polynomial-time (resp., any computational power unlimited) distinguisher algorithm D, it holds that $|Pr[D(\kappa, \ell, g, view_{\mathcal{A}}(\kappa, \ell)) = 1] - Pr[D(\kappa, \ell, g, sview(\kappa, \ell)) = 1]| = \varepsilon$, where ε is negligible in k. The probability is taken over the random coins used by $\mathsf{Setup}(\kappa, \ell)$, the random coins used by $\mathcal{A}$, $\mathcal{S}$, the reader R and all (uncorrupted) tags, the choice of g, and the coins used by the distinguisher algorithm D.

We now extend our definition to forward zk-privacy and backward zk-privacy. Denote by $(k^f_{\mathcal{T}_g}, s^f_{\mathcal{T}_g})$ (resp., $(k^1_{\mathcal{T}_g}, s^1_{\mathcal{T}_g})$) the final (resp., initial) secret-key and internal state of $\mathcal{T}_g$ at the end of (resp., beginning) of the experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$. An RFID protocol π is of *forward* (resp., *backward*) *zk-privacy*, if for any PPT CMIM adversary $\mathcal{A}$ there exists a polynomial-time simulator $\mathcal{S}$ such that for all sufficiently large κ and any $\ell = poly(\kappa)$, the following distributions are indistinguishable:

$$\{k^f_{\mathcal{T}_g}, s^f_{\mathcal{T}_g}(resp., k^1_{\mathcal{T}_g}, s^1_{\mathcal{T}_g}), g, view_{\mathcal{A}}(\kappa, \ell)\}$$

$$\{k^f_{\mathcal{T}_g}, s^f_{\mathcal{T}_g}(resp., k^1_{\mathcal{T}_g}, s^1_{\mathcal{T}_g}), g, sview(\kappa, \ell)\}$$

For forward/backward zk-privacy, it is required that the challenge tag $\mathcal{T}_g$ should remain *clean* at the end of experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$. Note that the adversary is allowed to corrupt the challenge tag after the end of experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$.

4.1 Discussions

Why allow $\mathcal{A}_1$ to output an arbitrary set *$\mathcal{C}$ of tags, and limit $\mathcal{A}_2$ to blind access to a challenge tag chosen randomly from $\mathcal{C}$?* The definition of zk-privacy implies that the adversary $\mathcal{A}$ cannot distinguish any challenge tag $\mathcal{T}_g$ from any set $\mathcal{C}$ of tags; otherwise, $\mathcal{A}$ can figure out the identity of $\mathcal{T}_g$ in $\mathcal{C}$ from its view $view_{\mathcal{A}}$, while this tag's identity cannot be derived from any simulator's view $sview$ (a formal proof of this in case of $|\mathcal{C}| = 2$ is provided in Section 5.2). If $\mathcal{C}$ is removed from the definition of zk-privacy, it is possible for

the adversary to distinguish any two tags under its attack, even if each of the tags can be perfectly simulated by a simulator. A special case is that each tag has an upper-bound of sessions in its life time so that an adversary can distinguish any two tags by setting one tag to be run out of sessions in the learning stage [14]. In addition, we do not restrict $\mathcal{C}$ to two tags so as to take into account the case that any number of tags may be correlated.

Why limit $\mathcal{A}_1$ to output of clean *tags?* If $\mathcal{A}_1$ is allowed to output "unclean tags," $\mathcal{A}_2$ can trivially violate the privacy of the tags selected by $\mathcal{A}_1$. Consider that $\mathcal{A}_1$ selects two tags that are waiting for different round message (e.g., one tag is clean and the other is not), then $\mathcal{A}_2$ can trivially distinguish them by forwarding to $\mathcal{T}_g$ different round messages.

Why allow $\mathcal{S}$ to have access to oracles in $\mathcal{O}$? Suppose that $\mathcal{S}$ simulates a tag from scratch and $\mathcal{A}$ (run by $\mathcal{S}$ as a subroutine) requests to corrupt the tag in the middle of the simulation. Without oracle access, it is difficult or even impossible for $\mathcal{S}$ to continue its simulation for the tag and keep it consistent with its previous simulation for the same tag.

Why limit $sview$ to include all oracle answers to queries made by $\mathcal{S}$? This is to restrict $\mathcal{S}$ not to access the oracles in $\mathcal{O}$ more than $\mathcal{A}$ does. The indistinguishability between the simulated view $sview$ and the real view $view_{\mathcal{A}}$ of adversary $\mathcal{A}$ in zk-privacy implies that for any (t, n_1, n_2, n_3, n_4)-adversary $\mathcal{A}$, with overwhelming probability, $\mathcal{S}$ cannot query O_1, O_2, O_3, O_4 more than n_1, n_2, n_3, n_4 times, respectively.

Why require $\mathcal{T}_g$ to remain clean at the end of $\mathbf{Exp}^{zkp}_{\mathcal{A}}$ for forward/backward privacy? In general, forward/backward privacy cannot be achieved if the adversary is allowed to corrupt the challenge tag before the end of its sessions in $\mathbf{Exp}^{zkp}_{\mathcal{A}}$ (i.e., the tag is not clean at the moment of corruption); otherwise, the adversary is able to derive certain protocol messages from the tag's internal state, secret-key, random coins, and the partial session transcript.

More on backward privacy. In general, backward privacy means that even if $\mathcal{A}$ learns the internal state and secret-key of a tag for the v-th session, it still cannot distinguish the run of $(v+1)$-th session run by this tag from a simulated session run. Without loss of generality, we assume that the internal state and secret-key known to $\mathcal{A}$ are the initial ones (i.e., $k^1_{\mathcal{T}_g}$ and $s^1_{\mathcal{T}_g}$).

For most RFID protocols in practice, the internal state and the secret-key of any tag at any time t can be determined by the tag's initial state, initial secret-key, and the session transcript related to the tag up to time t. In such a case, the indistinguishability between the simulated view $sview$ of $\mathcal{S}$ and the real view $view_{\mathcal{A}}$ of $\mathcal{A}$, relies upon the random coins used by $\mathcal{T}_g$ in experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$. These random coins are not disclosed to $\mathcal{A}$ since the random coins used by an uncorrupted tag in any session are erased once the session is completed, and the challenge tag $\mathcal{T}_g$ is required to be clean at the end of $\mathbf{Exp}^{zkp}_{\mathcal{A}}$.

On some special cases in zk-privacy experiments. One special case is that in the experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$, $\mathcal{A}_1$ outputs $\mathcal{C} = \mathcal{T}$. In this case, the simulator $\mathcal{S}_2$ does not have oracle access to any tag. The zk-privacy is analogue to auxiliary-input zero-knowledge [5], where the view of $\mathcal{A}_1$/$\mathcal{S}_1$ corresponds to the auxiliary input. Another special case is that $\mathcal{A}_1$ outputs only a single tag in $\mathcal{C}$, and all other tags can be corrupted by $\mathcal{A}_1$ and $\mathcal{A}_2$. In this case, the forward/backward zk-privacy implies that both adversary $\mathcal{A}$ and simulator $\mathcal{S}$ have access to certain secret information of all tags.

5 Comparison with Existing RFID Security and Privacy Frameworks

In this section, we compare our RFID security and privacy framework, including adaptive completeness, mutual authentication, and zk-privacy, with typical existing frameworks. We argue that our framework is more reasonable in practice than some frameworks, and it is stronger in terms of privacy than at least one of the existing frameworks. We also clarify some subtleties and confusions in the existing frameworks.

5.1 Comparison with Model in [21, 19]

Vaudenay and Paise proposed a very flexible and comprehensive framework for RFID security and privacy in [21, 19]. In this framework , the adversary is categorized into the following classes:

- Weak adversary, which cannot corrupt any tags.
- Forward adversary, which can corrupt tags under the limitation that once the adversary corrupts a tag, it can do nothing subsequently except for corrupting more tags.
- Destructive adversary, which can do anything after a tag corruption, but under the limitation that the adversary cannot reuse a tag after corrupting it. Specifically, once a tag is corrupted it will be virtually destroyed. In particular, a destructive adversary cannot observe or interact with a corrupted tag nor can the adversary impersonate a corrupted tag to the reader.
- Strong adversary, which has no limitations on corrupting tags, and can do anything at its wish.

For each category of adversary defined above, it is also defined a *narrow* variant, where a narrow adversary cannot access the outputs of the players (i.e., reader and tags) for any protocol run.

Suppose that C is one of the adversary categories. Informally, an RFID protocol is called C-private if for any

adversary $\mathcal{A} \in C$, there exists a simulator $\mathcal{S}$ such that $\mathcal{A}$ cannot distinguish its interactions with the actual RFID system or with the simulator (the reader is referred to [21, 19] for formal definitions).

The major differences between our framework and this model are summarized below:

- In [21, 19], the simulator is not required to handle tag corruption queries by the adversary. In other words, the simulator works only for those adversaries which do not make tag corruption queries. It is not clear how such simulator acts upon tag corruption queries made by an adversary. Suppose that $\mathcal{S}$ simulates a tag from scratch and $\mathcal{A}$ (typically run by $\mathcal{S}$ as a subroutine) requests to corrupt the tag in the middle of simulation (possibly in the middle of a session run). Without access to tag corruption queries, it is difficult or even impossible for $\mathcal{S}$ to continue its simulation for the tag and keep it consistent with its previous simulation for the same tag.

- The adversary considered in our framework essentially corresponds to strong adversary in [21, 19], with the difference in that the adversary cannot corrupt any tag in set C before the end of zk-privacy experiment $\mathbf{Exp}^{zkp}_{\mathcal{A}}$. In comparison, the model in [21, 19] poses no restriction on tag corruption (though it is not clear how the simulator handles such adversaries), which implies that an adversary can corrupt any tag at any time (possibly in the middle of session). However, in such case, forward/backward privacy may not be achievable if the challenge tag is corrupted in the middle of session; this is the reason why we require that the challenge tag $\mathcal{T}_g$ must remain *clean* at the moment of corruption. Indeed, there are some confusions in [21, 19].

- The matching session concept defined in [21, 19] is restrict to identical session transcript, without clarifying some subtleties such as the "last-round-message attacks" for defining authentication from tag to reader.

- The notion of adaptive completeness is not defined in [21, 19]. The completeness notion in [21, 19] is defined for honest protocol execution only, with no adversarial desynchronizing attacks being taken into account.

5.2 Comparison with Model in [14]

The RFID privacy model proposed in [14] describes the indistinguishability between any two tags by an adversary. We refer to this privacy notion as "ind-privacy." We shall prove that zk-privacy is stronger than a revised version of ind-privacy after some subtleties are clarified.

Experiment $\mathbf{Exp}^{ind}_{\mathcal{A}}[\kappa, \ell, n_1, n_2, n_3, n_4]$
1. run Setup(κ, ℓ) to setup the reader R and a set of tags $\mathcal{T}$; denote by $para$ the system public parameters;
2. $\{\mathcal{T}_i, \mathcal{T}_j, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}}(R, \mathcal{T}, para)$; *//learning stage*
3. set $\widehat{\mathcal{T}} = \mathcal{T} - \{\mathcal{T}_i, \mathcal{T}_j\}$;
4. $b \in_R \{0, 1\}$;
5. if $b = 0$ then $\mathcal{T}_g = \mathcal{T}_i$, else $\mathcal{T}_g = \mathcal{T}_j$;
6. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(R, \widehat{\mathcal{T}}, \mathcal{I}(\mathcal{T}_g), st)$; *//guess stage*
7. the experiment outputs 1 if $b' = b$, 0 otherwise.

Figure 4. Ind-Privacy Experiment

Figure 4 illustrates the ind-privacy experiment $\mathbf{Exp}^{ind}_{\mathcal{A}}[\kappa, \ell, n_1, n_2, n_3, n_4]$ ($\mathbf{Exp}^{ind}_{\mathcal{A}}$, for simplicity) in terms of the notion defined in this paper. In $\mathbf{Exp}^{ind}_{\mathcal{A}}$, an adversary $\mathcal{A}$ is comprised of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ and runs in two stages. Throughout the experiment, the adversary $\mathcal{A}$ is allowed to launch O_1, O_2, O_3 and O_4 oracle queries without exceeding n_1, n_2, n_3 and n_4 overall calls, respectively. The experiment proceeds as follows. At first, the experiment runs Setup(κ, ℓ) to setup an RFID system $(R, \mathcal{T})$. Then, in the *learning stage*, algorithm $\mathcal{A}_1$ outputs a piece of state information st and a pair of uncorrupted tags $\{\mathcal{T}_i, \mathcal{T}_j\}$ to which it has not made Corrupt queries. Next, the experiment selects a random bit b and sets the challenge tag $\mathcal{T}_g = \mathcal{T}_i$ if $b = 0$, and $\mathcal{T}_g = \mathcal{T}_j$ otherwise. Finally, in the *guess stage*, algorithm $\mathcal{A}_2$ is asked to guess the random bit b by outputting a bit b'. During the second stage, $\mathcal{A}_2$ can interact with R and the tags in $\widehat{\mathcal{T}} = \mathcal{T} - \{\mathcal{T}_i, \mathcal{T}_j\}$, and can blindly access (but cannot corrupt) the challenge tag $\mathcal{T}_g$ via the interface $\mathcal{I}$[6].

Definition 5.1 (ind-privacy) *The advantage of adversary $\mathcal{A}$, denoted $Adv^{ind}_{\mathcal{A}}(\kappa, \ell, n_1, n_2, n_3, n_4)$, in the experiment $\mathbf{Exp}^{ind}_{\mathcal{A}}[\kappa, \ell, n_1, n_2, n_3, n_4]$ is defined to be :*

$$|Pr[\mathbf{Exp}^{ind}_{\mathcal{A}}[\kappa, \ell, n_1, n_2, n_3, n_4] = 1] - \frac{1}{2}|.$$

An adversary $\mathcal{A}$ $(\epsilon, t, n_1, n_2, n_3, n_4)$-breaks the ind-privacy of the RFID system $(R, \mathcal{T})$ if the advantage $Adv^{ind}_{\mathcal{A}}(k, \ell, n_1, n_2, n_3, n_4)$ of $\mathcal{A}$ in the experiment $\mathbf{Exp}^{ind}_{\mathcal{A}}$ is at least ϵ and the running time of $\mathcal{A}$ is at most t.

An RFID system $(R, \mathcal{T})$ is said to be of ind-privacy, if there exists no adversary who can $(\epsilon, t, n_1, n_2, n_3, n_4)$-break the RFID system for some non-negligible ϵ and some polynomials t, n_1, n_2, n_3, n_4 (all of them are in κ).

On some subtleties in ind-privacy. In the original definition of ind-privacy, it is not explicitly specified that the two tags

[6] In [14], it is stated that $\mathcal{A}_2$ is still allowed to access the challenge tag $\mathcal{T}_g$ but cannot corrupt $\mathcal{T}_g$, without formally formulating the interface entity $\mathcal{I}$ as within our framework.

output by $\mathcal{A}_1$ must be clean tags. In the definition of forward ind-privacy [14], it is not precisely specified the time point of tag corruption and the actions of adversary after tag corruption.

On relationship between ind-privacy and zk-privacy. It was mentioned in [14] that an important area for future research is to study stronger RFID privacy notions. Our proposed notion, zk-privacy, is indeed stronger than ind-privacy (specifically, a variant of ind-privacy with clean tags being explicitly required for the output of $\mathcal{A}_1$).

Proposition 5.1 *zk-privacy is stronger than ind-privacy.*

proof. We prove that if an RFID system $(R, \mathcal{T})$ is not of ind-privacy, then it is also not of zk-privacy. To prove this, we show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which can $(\epsilon, t, n_1, n_2, n_3, n_4)$-break the ind-privacy of the RFID system $(R, \mathcal{T})$, then we can construct another PPT adversary $\mathcal{A}'$ such that no PPT simulator exists for $\mathcal{A}'$.

In the experiment $\mathsf{Exp}^{zkp}_{\mathcal{A}'}$, let $\mathcal{A}'$ run $\mathcal{A}$ and do whatever $\mathcal{A}$ does. In particular, $\mathcal{A}'$ and $\mathcal{A}$ are of the same parameters (t, n_1, n_2, n_3, n_4). Since $\mathcal{A}$ run by $\mathcal{A}'$ always outputs a *pair* of clean tags at the end of its first stage, $\mathsf{Exp}^{zkp}_{\mathcal{A}'}$ outputs $(g, view_{\mathcal{A}'})$, where $g \in \{0, 1\}$ is a random bit, and $view_{\mathcal{A}'}$ implicitly determines the output of $\mathcal{A}$ (i.e., the guess bit b' in the experiment $\mathsf{Exp}^{ind}_{\mathcal{A}}$). In other words, the guess bit b' can be computed out from $view_{\mathcal{A}'}$ in polynomial-time. As we assume $\mathcal{A}$ $(\epsilon, t, n_1, n_2, n_3, n_4)$-breaks ind-privacy, it holds that $b' = g$ for the output of $\mathsf{Exp}^{zkp}_{\mathcal{A}'}$ with probability at least $\frac{1}{2} + \epsilon$. However, the simulated view $sview$ in the output of the experiment $\mathsf{Exp}^{zkp}_{\mathcal{S}}$ is independent of g (recall that the random value g is unknown to the simulator $\mathcal{S}$). Therefore, for the guess bit b' implied by $sview$ (which can be computed out from $sview$ in polynomial-time), it always holds that $Pr[b' = g] = \frac{1}{2}$. This shows that for the above $\mathcal{A}'$ and for any polynomial-time simulator, there exists a polynomial-time distinguisher that can distinguish the output of $\mathsf{Exp}^{zkp}_{\mathcal{A}}$ and that of $\mathsf{Exp}^{zkp}_{\mathcal{S}}$ with non-negligible probability at least ϵ. □

5.3 Comparison with Models in [8, 16]

The RFID privacy notion given in [8, 16] is formulated based on the unpredictability of protocol output. We refer to this privacy notion as "unp-privacy." The unp-privacy is formulated with respect to RFID protocols with a 3-round canonical form, denoted as $\pi = (c, r, f)$, where c, r, f stand for the first, second, and third round message, respectively. Note that our framework, as well as models in [14, 21, 19]), are not confined to this protocol structure.

The unp-privacy notion formulated in [8, 16] essentially says that the second-round message sent from a tag must be pseudorandom (i.e., indistinguishable from a truly random string). We observe that this requirement has certain limitations. First, given any unp-private RFID protocol $\pi = (c, r, f)$ between a reader and a tag, we can modify the protocol to $\pi' = (c, r||1, f)$, where "||" denotes the string concatenation operation. That is, the modified protocol π' is identical to π except that in the second-round the tag additionally concatenates a bit '1' to r. This modified RFID-protocol π' is not of unp-privacy, as the second-round message $r||1$ is clearly not pseudorandom. However, intuitively, the tags' privacy should be preserved since the same bit '1' is appended to all second-round messages for all tags. Notice that when RFID-protocols are implemented in practice, the messages being exchanged between reader and tags normally bear some non-random information such as version number of RFID standard. Another limitation is that the unp-privacy may exclude the use of public-key encryption in RFID-protocols, as public-key generated ciphertexts are typically *not* pseudorandom.

Another point is that the adversaries considered in the definition of unp-privacy [8, 16] is not allowed to access protocol outputs. Therefore, such adversaries are *narrow* ones as defined in [21, 19]. Informally, the unp-privacy experiment works as follows. Given a first-round message c (which could be generated by the adversary $\mathcal{A}$), the experiment selects a value r which could be either the actual second-round message generated by an uncorrupted tag in response to c or just a random value in a certain domain; then the experiment presents the value r to $\mathcal{A}$. The unp-privacy means that $\mathcal{A}$ cannot determine in which case the value r is. Note that if $\mathcal{A}$ have access to protocol outputs, it can simply distinguish between the two cases of r. What $\mathcal{A}$ needs to do is to forward r to the reader R as the second round message. If r is generated by an uncorrupted tag (and the value c was generated by the reader in a matching session), the reader will always output "accept." On the other hand, if r is just a random value, with overwhelming probability R will reject the message due to authentication soundness from tag to reader.

In summary, we argue that zk-privacy is more reasonable than unp-privacy in practice. It allows for more general protocol structure, more powerful adversary, and non-pseudorandom protocol messages.

5.4 Comparison with Traditional Formulation of Zero-Knowledge

The notion of zk-privacy is defined based on the traditional zero-knowledge formulation [7, 5] with the following differences. First, in zk-privacy, the simulator $\mathcal{S}$ is allowed to have access to oracles in $\mathcal{O}$ (where the actions of these oracles may depend upon some secret values such as secret-keys and internal states), while traditional zk-simulator is a

polynomial-time algorithm without oracle access to players of secret values. Second, the zk-privacy is formulated against a structured adversary $\mathcal{A}$ which is divided into two phases, while the traditional zk is formulated against any polynomial-time adversary. Third, in zk-privacy, the random challenge g is unknown to $\mathcal{A}$, but is presented to the distinguisher, which renders extra power to the distinguisher; in comparison, in the traditional zero-knowledge formulation, the distinguisher and the adversary essentially have the same power and advantage. Lastly, for forward (resp., backward) zk-privacy, the final (resp., initial) secret-key and internal state of the challenge tag $\mathcal{T}_g$ are disclosed to $\mathcal{A}$, while for the traditional zero-knowledge formulation, no secret values of the knowledge prover are assumed to be leaked to the adversary.

6 An RFID Protocol within Our Framework

In this section, we present a modified and refined version of the RFID protocol proposed in [16] and show that it is of adaptive completeness, mutual authentication, and zk-privacy within our framework. The operation of the RFID protocol between the reader R and a tag $\mathcal{T}_i$ is shown in Figure 5.

Protocol. Let F_k: $\{0,1\}^{2\kappa} \rightarrow \{0,1\}^{2\kappa}$ be a pre-specified keyed PRF and F_k^0 (resp., F_k^1) the κ-bit prefix (resp., suffix) of the output of F_k, where κ is the system security parameter. In practice, the PRF can be implemented based on some lightweight stream or block ciphers [10, 2, 9].

When a tag $\mathcal{T}_i$ with identity ID registers to the reader R, it is assigned a secret-key $k \in_R \{0,1\}^\kappa$, a counter ctr of length l_{ctr} with initial value 1. R pre-computes an initial index $I = F_k^0(1||pad_1)$ for the tag, where $pad_1 \in \{0,1\}^{2\kappa - l_{ctr}}$ is a fixed padding, and stores the tuple (I, k, ctr, ID) into its database.

At the start of a new protocol session, R sends a challenge string $c \in_R \{0,1\}^\kappa$ to $\mathcal{T}_i$, which also serves as the session identifier. To simplify the presentation, the session identifier as well as the corresponding verification of the identifier by protocol players are implicitly implied and will not be explicitly mentioned in the following.

Upon receiving c from R, $\mathcal{T}_i$ computes $I = F_k^0(ctr||pad_1)$, $(r_0, r_1) = F_k(c||I)$ (where $r_0 = F_k^0(c||I)$ and $r_1 = F_k^0(c||I)$), and $r_{\mathcal{T}} = r_0 \oplus (ctr||pad_2)$. $\mathcal{T}_i$ sends $(I, r_{\mathcal{T}})$ to R and then updates its counter $ctr = ctr + 1$, where $pad_2 \in \{0,1\}^{\kappa - l_{ctr}}$ is another predetermined padding string.

After receiving $(I, r_{\mathcal{T}})$ from $\mathcal{T}_i$, R searches its database to find a tuple indexed by I:

- If R finds such a tuple, say (I, k, ctr', ID), it computes $(r_0, r_1) = F_k(c||I)$, and checks whether $ctr'||pad_2 = r_0 \oplus r_{\mathcal{T}}$: If yes, R accepts $\mathcal{T}_i$ by outputting "1", sends $r_R = r_1$ to the tag, updates the tuple (I, k, ctr', ID) with $ctr' = ctr' + 1$ and $I = F_k^0(ctr'||pad_1)$; If not, R searches for the next tuple including I (to avoid potential collision of index I, i.e., two different tuples are of the same index I). We remark that in practice, when no adversary exists (particularly no desynchronization occurs), with overwhelming probability there exists just one tuple of index I in reader's database and R will succeed with this tuple without performing subsequent actions.

- If no tuple is found to have an index I (which indicates counter desynchronization between R and $\mathcal{T}_i$), for each tuple (I', k, ctr', ID) in its database, R computes $(r_0, r_1) = F_k(c||I)$ and $ctr||pad_2 = r_0 \oplus r_{\mathcal{T}}$, and checks whether $I = F_k^0(ctr||pad_1)$: If yes (which indicates ctr is the correct counter value at $\mathcal{T}_i$), R accepts $\mathcal{T}_i$, outputs "1", sends back $r_R = r_1$ as the third message, and updates the tuple (I', k, ctr', ID) with $ctr' = ctr + 1$ and $I' = F_k^0(ctr'||pad_1)$. In the that case R fails with all the tuples in its database, it rejects the tag and outputs "0".

Upon receiving r_R, $\mathcal{T}_i$ checks whether $r_R = r_1$: If yes, $\mathcal{T}_i$ accepts the reader and outputs "1"; otherwise it rejects the reader and outputs "0".

In comparison with the protocol proposed in [16], the above protocol adds mutual authentication (and is logically more precise), and we can formally prove that it is of adaptive completeness, mutual authentication, and zk-privacy within the new framework. Analysis of completeness and authentication was not conducted in [16], and as we shall see, the zk-privacy analysis of the protocol depicted in Figure 5 is much more complicated than the unp-privacy analysis in [16]. We suggest that the methodology used in our analysis is of independent interest, which can be applied to analyze other RFID protocols (particularly those based on PRFs) within our new framework.

Theorem 1 *Assuming F_k is a pseudorandom function, the protocol depicted in Figure 5 is of adaptive completeness, mutual authentication and zk-privacy.*

The complete proof of the theorem is given in Appendix B. Below we provide a high level analysis of its zk-privacy property.

According to the zk-privacy formulation presented in Section 4, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ works as follows: In the *first stage* of $\mathcal{S}$, $\mathcal{S}_1$ perfectly mimics $\mathcal{A}_1$ by running $\mathcal{A}_1$ as a subroutine and with oracle access to the four oracles in $\mathcal{O} = \{O_1, O_2, O_3, O_4\}$).

After the challenge tag $\mathcal{T}_g$ is specified (taken randomly from the clean tag set $\mathcal{C}$ output by $\mathcal{A}_1$), in the *second stage* of $\mathcal{S}$, $\mathcal{S}_2$ runs $\mathcal{A}_2$ as a subroutine and concurrently interacts

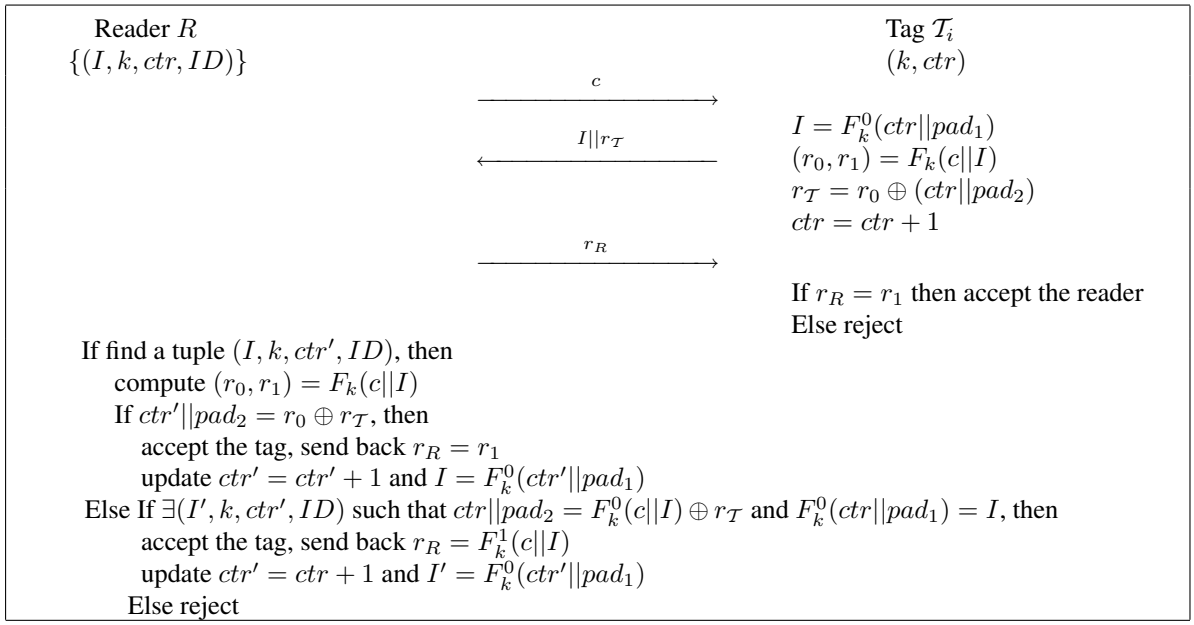

Figure 5. RFID Protocol with Mutual Authentication and zk-Privacy

with the reader R and the tags in $\widehat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$. For all oracle queries made by $\mathcal{A}_2$ directed to tags in $\widehat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$, $\mathcal{S}_2$ makes the same oracle queries, and relays back oracle answers to $\mathcal{A}_2$. But, for oracle queries made by $\mathcal{A}_2$ directed to the reader R and to the challenge tag $\mathcal{T}_g$ (blindly accessed by $\mathcal{A}_2$), $\mathcal{S}_2$ works as follows: (Recall that, the oracle queries made by $\mathcal{A}_2$ to the challenge tag is of the form $\mathsf{SendT}(challenge, \cdot)$.)

1. On oracle query $\mathsf{InitReader}()$, $\mathcal{S}_2$ makes the same oracle query to R, and gets back a random string $c \in \{0,1\}^\kappa$ from R. Then, $\mathcal{S}_2$ relays back c to $\mathcal{A}_2$.

2. On oracle query $\mathsf{SendT}(challenge, \hat{c})$, where the challenge tag $\mathcal{T}_g$ (simulated by $\mathcal{S}_2$) currently does not run any session, $\mathcal{S}_2$ opens a session for $\mathcal{T}_g$ with $\hat{c}$ as the first-round message (that also serves as the session-identifier of this new session); Then, $\mathcal{S}_2$ randomly selects $I, r_{\mathcal{T}} \in_R \{0,1\}^\kappa$, and sends back $I||r_{\mathcal{T}}$ to $\mathcal{A}_2$ as the second-round message.

3. On oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$, $\mathcal{S}_2$ works as follows:

   Case-3.1. If $\hat{I}||\hat{r}_{\mathcal{T}}$ was sent by the challenge tag $\mathcal{T}_g$ (simulated by $\mathcal{S}_2$) in a session of session-identifier $\hat{c}$, $\mathcal{S}_2$ simulates the responses of the reader R according to the following two cases.

   Case-3.1.1 If R is running an incomplete session of session-identifier $\hat{c}$ (i.e., $\hat{c}$ was sent by R upon an $\mathsf{InitReader}$ query and R is waiting for the second-round message), $\mathcal{S}_2$ just returns back a random string $r_R \in_R \{0,1\}^\kappa$ to $\mathcal{A}_2$, and outputs "1" indicating "accept".

   Case-3.1.2. Otherwise, $\mathcal{S}_2$ simply returns back a special symbol "$\perp$" indicating invalid query.

   Case-3.2. In all other cases, $\mathcal{S}_2$ makes the same oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$ to the reader R, and relays back the answer from R to $\mathcal{A}_2$.

4. On oracle query $\mathsf{SendT}(challenge, \hat{r}_R)$, where the challenge tag $\mathcal{T}_g$ (simulated by $\mathcal{S}_2$) currently runs a session of partial session-transcript $(\hat{c}, I||r_{\mathcal{T}})$ and is waiting for the third-round message, $\mathcal{S}_2$ works as follows:

   Case-4.1. If there exists a matching session of the same session transcript $(\hat{c}, I||r_{\mathcal{T}}, \hat{r}_R)$ at the side of R (where $\hat{r}_R$ may be simulated by $\mathcal{S}_2$ as in the above Case-3.1), $\mathcal{S}_2$ outputs "1" indicating "accept";

   Case-4.2. Otherwise, $\mathcal{S}_2$ simply outputs "0" indicating "reject".

5. Output of $\mathcal{S}_2$: Finally, whenever $\mathcal{A}_2$ stops, $\mathcal{S}_2$ also stops and outputs the simulated view $sview$ as specified in the zk-privacy definition, which particularly consists of all oracle answers (including ones provided by the real oracles in $\mathcal{O}$ and *ones simulated by* $\mathcal{S}_2$) to queries made by $\mathcal{A}$.

It is easy to see that $\mathcal{S}$ works in polynomial-time. We investigate the differences between the simulated view $sview$ output by $\mathcal{S}$ and the real view $view_{\mathcal{A}}$ of $\mathcal{A}$:

Difference-1: In Case-4.1 (resp., Case-4.2) $\mathcal{S}_2$ always outputs "accept" (resp., "reject"), while the actual challenge tag $\mathcal{T}_g$ may output "reject" in Case-4.1 (resp., "accept" in Case-4.2) in the experiment $\mathbf{Exp}_{\mathcal{A}}^{zkp}$.

Difference-2: On oracle query $\mathsf{SendT}(challenge, \hat{c})$ or in Case-3.1 upon the oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$, $\mathcal{S}_2$ always returns back truly random strings, while the actual players (i.e., the challenge tag $\mathcal{T}_g$ and the reader R) provide pseudorandom strings in the experiment $\mathbf{Exp}_{\mathcal{A}}^{zkp}$, by invoking the PRF F_k where k is the secret-key of the challenge tag $\mathcal{T}_g$.

Intuitively, Difference-1 can occur only with negligible probability, by the properties of adaptive completeness and mutual authentication. The subsequent analysis argues that the properties of adaptive completeness and mutual authentication indeed hold under the simulation of $\mathcal{S}$ in $\mathbf{Exp}_{\mathcal{S}}^{zkp}$.

Intuitively, Difference-2 should not constitute distinguishable gap between $sview$ and $view_{\mathcal{A}}$, due to the pseudorandomness of F_k. But, the technical difficulty and subtlety here is that: the difference between pseudorandomness and real randomness only occurs in the second stages of both $\mathbf{Exp}_{\mathcal{A}}^{zkp}$ and $\mathbf{Exp}_{\mathcal{S}}^{zkp}$, and the first stages of both $\mathbf{Exp}_{\mathcal{A}}^{zkp}$ and $\mathbf{Exp}_{\mathcal{S}}^{zkp}$ are w.r.t. the pseudorandom function F_k. In other words, to distinguish the PRF F_k from a truly random one in the second stage, the distinguisher has already accessed F_k for polynomially many times in the first stage. In general, the definition of PRF says nothing on the pseudorandomness in the second stage. To overcome this technical difficulty, we build a list of hybrid experiments.

In the first hybrid experiment, a polynomial-time algorithm $\hat{S}$ runs $\mathcal{A}$ as a subroutine and has oracle access to the PRF F_k or a truly random function H. $\hat{S}$ first randomly guesses the challenge tag $\mathcal{T}_g$ (by taking g uniformly at random from $\{1, \cdots, \ell\}$), and then setups the RFID system $(R, \mathcal{T})$ except for the challenge-tag $\mathcal{T}_g$. Note that $\hat{S}$ can perfectly handle all oracle queries made by $\mathcal{A}$ to the reader R and all tags in $\mathcal{T} - \{\mathcal{T}_g\}$. For oracle queries directed to $\mathcal{T}_g$, $\hat{S}$ mimics $\mathcal{T}_g$ with the aid of its oracle, i.e, the PRF F_k or a truly random function H. Denote by the view of $\mathcal{A}$ under the run of $\hat{S}$ with oracle access to F_k (resp., H) as $view_{\mathcal{A}}^{\hat{S}^{F_k}}$ (resp., $view_{\mathcal{A}}^{\hat{S}^{H}}$). By the pseudorandomness of F_k, we have that $view_{\mathcal{A}}^{\hat{S}^{F_k}}$ and $view_{\mathcal{A}}^{\hat{S}^{H}}$ are indistinguishable. Next, suppose $\hat{\mathcal{S}}$ successfully guesses the challenge tag $\mathcal{T}_g$ (that occurs with probability $\frac{1}{\ell}$), $view_{\mathcal{A}}^{\hat{S}^{F_k}}$ is identical to $view_{\mathcal{A}}$. In particular, in this case, the properties of adaptive completeness and mutual authentication hold in $view_{\mathcal{A}}^{\hat{S}^{F_k}}$ and thus also in $view_{\mathcal{A}}^{\hat{S}^{H}}$ (as $view_{\mathcal{A}}^{\hat{S}^{F_k}}$ and $view_{\mathcal{A}}^{\hat{S}^{H}}$ are indistinguishable). Thus, to show the indistinguishability between $view_{\mathcal{A}}$ and $sview$, it is reduced to show the indistinguishability between $view_{\mathcal{A}}^{\hat{S}^{H}}$ (in case $\hat{S}$ successfully guesses the challenge tag $\mathcal{T}_g$) and $sview$.

In the second hybrid experiment, we consider another polynomial-time algorithm S' that mimics $\hat{S}$, with oracle access to F_k or a truly random function H, but with the following modifications: in the second stage of this hybrid experiment, S' essentially mimics the original zk-privacy simulator $\mathcal{S}$. Denote by the view of $\mathcal{A}$ under the run of S' with oracle access to F_k (resp., H) as $view_{\mathcal{A}}^{S'^{F_k}}$ (resp., $view_{\mathcal{A}}^{S'^{H}}$). By the pseudorandomness of F_k, $view_{\mathcal{A}}^{S'^{F_k}}$ and $view_{\mathcal{A}}^{S'^{H}}$ are indistinguishable. We can show that $view_{\mathcal{A}}^{S'^{H}}$ and $view_{\mathcal{A}}^{\hat{S}^{H}}$ are also indistinguishable, and that $view_{\mathcal{A}}^{S'^{F_k}}$ and $sview$ are also indistinguishable (conditioned on S' successfully guesses the challenge tag $\mathcal{T}_g$), which particularly implies that the properties of adaptive completeness and mutual authentication hold also in $sview$. This establishes the indistinguishability between $sview$ and $view_{\mathcal{A}}$.

7 Conclusion and Future Work

In this paper, we proposed a new zero-knowledge based framework for RFID system privacy which incorporates the notions of adaptive completeness and mutual authentication. The framework captures various security and privacy requirements in practical RFID systems and does not suffer from the limitations of the existing RFID privacy models in the literature. We formally proved that our framework is stronger than the ind-privacy model of [14] and therefore answered an open question posed in [14] for developing stronger RFID privacy models. We further presented a modified version of the RFID protocol in [16] and showed the protocol is of adaptive completeness, mutual authentication and zk-privacy.

One of our future research directions is to analyze existing RFID protocols and design new protocols within the new framework presented in this paper.

Since our framework is formulated w.r.t. the basic scenario of a RFID system consisting of a single uncompromised reader and multiple tags, where tags identify themselves to the reader individually and independently. A future research direction is to extend our RFID privacy framework to more sophisticated and practical scenarios which allow compromising of readers, tag group authentication, and tan ownership transfer.

References

[1] C. Berbain, O. Billet, J. Etrog and H. Gilbert. An Efficient Forward Private RFID Protocol. In *Conference on Computer and Communications Security – CCS'09*.

[2] C. de Canniere and B. Preneel. Trivium. In M. Robshaw and O. Billet, editors, New Stream Cipher Designs: The eSTREAM Finalists, volume 4986 of LNCS, pages 244-266. Springer-Verlag, 2008.

[3] I. Damgård and M. Ostergaard. RFID Security: Tradeoffs between Security and Efficiency. In *Topics in Cryptology–CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 318–332, 2008.

[4] S. Garfinkel, A. Juels, and R. Pappu. RFID Privacy: An Overview of Problems and Proposed Solutions. *IEEE Security and Privacy*, 3(3):34–43, 2005.

[5] O. Goldreich. *The Foundations of Cryptography*, volume I, Basic Tools. Cambridge University Press, 2001.

[6] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[7] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems In *ACM Symposium on Theory of Computing*, pages 291-304, 1985.

[8] J. Ha, S. Moon, J. Zhou, and J. Ha. A new formal proof model for RFID location privacy. In *European Symposium on Research in Computer Security (ESORICS) 2008*, volume 5283 of *Lecture Notes in Computer Science*.

[9] M. Hell, T. Johansson, and W. Meier. The Grain Family of Stream Ciphers. In M. Robshaw and O. Billet, editors, New Stream Cipher Designs: The eSTREAM Finalists, volume 4986 of LNCS, pages 179-190. Springer-Verlag, 2008.

[10] International Standard ISO/IEC 9798 Information technology−Security techniques−Entity authentication−Part 5: Mechanisms using Zero-Knowledge Techniques.

[11] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *ASIACRYPT*, pages 52–66, 2001.

[12] A. Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006.

[13] A. Juels, R. L. Rivest, and M. Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy. In *8th ACM Conference on Computer and Communications Security – ACM CCS*, pages 103–111. ACM Press, 2003.

[14] A. Juels and S. Weis. Defining Strong Privacy for RFID. In *International Conference on Pervasive Computing and Communications – PerCom 2007*.

[15] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In *CRYPTO*, pages 293–308, 2005.

[16] C. Ma, Y. Li, R. Deng, and T. Li. RFID Privacy: Relation Between Two Notions, Minimal Condition, and Efficient Construction. In *Conference on Computer and Communications Security – ACM CCS*, 2009.

[17] C. Yu Ng, W. Susilo, Y. Mu, and R. Safavi-Naini. RFID privacy models revisited. In *European Symposium on Research in Computer Security (ESORICS) 2008*, volume 5283 of *Lecture Notes in Computer Science*.

[18] C. Yu Ng, W. Susilo, Y. Mu, and R. Safavi-Naini. New Privacy Results on Synchronized RFID Authentication Protocols against Tag Tracing. In *European Symposium on Research in Computer Security (ESORICS) 2009*, pages 321-336, volume 5786 of *Lecture Notes in Computer Science*.

[19] R.L.Paise and S. Vaudenay. Muthal Authentication in RFID: Security and Privacy. In *AsiaCCS 2008*, pages 292-299.

[20] A. Shamir. SQUASH: A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. FSE 2008, LNCS 5086, pages 144-157, 2008.

[21] S. Vaudenay. On Privacy Models for RFID. In *Advances in Cryptology - Asiacrypt 2007*.

[22] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *International Conference on Security in Pervasive Computing – SPC 2003*.

[23] 860 MHz - 930 MHz Class 1 Radio Frequency Identification Tag Radio Frequency and Logical Communication Interface Specification Candidate Recommendation Version 1.0.1, Auto-ID Center, 2002.

A Pseudorandom Functions

In this section, we review the definition of pseudorandom functions [6]. On a security parameter κ, let $m(\cdot)$ and $l(\cdot)$ be two positive polynomials in κ. We say that

$$\{F_k : \{0,1\}^{m(\kappa)} \longrightarrow \{0,1\}^{l(\kappa)}\}_{k \in_R \{0,1\}^\kappa}$$

is a PRF ensemble if the following two conditions hold:

1. Efficient evaluation: There exists a polynomial-time algorithm that on input k and $x \in \{0,1\}^{m(\kappa)}$ returns $F_k(x)$.

2. Pseudorandomness: A PPT oracle machine A (t,ϵ)-breaks the PRF ensemble, if

$$|\Pr[A^{F_\kappa}(\kappa)=1] - \Pr[A^{H_\kappa}(\kappa)=1]| \geq \epsilon$$

where F_κ is a random variable uniformly distributed over the multi-set $\{F_k\}_{k\in_R\{0,1\}^\kappa}$, H_κ is uniformly distributed among all functions mapping $m(\kappa)$-bit-long strings to $l(\kappa)$-bit-long strings, and the running time of A is at most t (here each oracle query accounts for one unit operation).

The PRF ensemble is (t,ϵ)-pseudorandom, if for all sufficiently large κ there exists no algorithm A that can (t,ϵ)-break the PRF ensemble. The PRF ensemble is pseudorandom, if for all sufficiently large κ's there exists no algorithm A that can (t,ϵ)-break the PRF ensemble, *for any t that is polynomial in κ and any ϵ that is non-negligible in κ.*

B Proof Details of Theorem 1

We provide the formal proof for Theorem 1 below.

B.1 Adaptive Completeness

Denote by $(c, I||r_\mathcal{T}, r_R, o_R^c, o_{\mathcal{T}_i}^c)$ the transcript of a session between the reader R and an uncorrupted tag $\mathcal{T}_i$. Suppose the session corresponds to the v-th session of $\mathcal{T}_i$ and the j-th session of R, where $1 \leq v \leq s$ and $1 \leq j \leq s\ell$. That is, after any potential (particularly, desynchronizing) attacks by a PPT CMIM adversary $\mathcal{A}$, finally R completes its j-th session and $\mathcal{T}_i$ completes its v-th session such that these two sessions are of identical protocol transcript. Let ID, k and ctr_i be the identity, secret-key and counter value of $\mathcal{T}_i$. Let $I = F_k^0(ctr_i||pad_1)$. We consider the probability that the event E (defined in Definition 3.1) occurs.

Firstly, note that as R sent the third-round message, we have $o_R^c = 1$. Secondly, we consider the probability that R identified a different tag $\mathcal{T}_{i'}(\neq \mathcal{T}_i)$ of identity $ID'(\neq ID)$ in its j-th session. This event occurs only in one of the following two cases:

Case-1. There exists a tuple (I, k', ctr', ID') in the database of R, where $ID' \neq ID$, such that $ctr'||pad_2 = r_0' \oplus r_\mathcal{T}$ and $(r_0', r_R) = F_{k'}(c||I)$, where $I = F_{k'}^0(ctr'||pad_1)$ was pre-computed by R.

Case-2. There exists a tuple (I', k', ctr', ID') in the database of R, where $ID' \neq ID$, such that $ctr||pad_2 = F_{k'}^0(c||I) \oplus r_\mathcal{T}$ and $I = F_{k'}^0(ctr||pad_1)$.

As $ID \neq ID'$ and the random secret keys for different tags are independent, it follows that $\Pr[k = k'] = 2^{-\kappa}$ in both two cases. Further note that both cases imply that there exists a tuple in the database of R that has a counter value, denoted $ctr_{i'}$ (i.e., ctr' or ctr), such that $I = F_k^0(ctr_i||pad_1) = F_{k'}^0(ctr_{i'}||pad_1)$. We consider the event that there *exists* an i', $1 \leq i' \neq i \leq \ell$, such that $I = F_k^0(ctr_i||pad_1) = F_{k'}^0(ctr_{i'}||pad_1)$, where k (resp., k') is the secret-key of $\mathcal{T}_i$ (resp., $\mathcal{T}_{i'}$ of identity ID'), and observe the following:

- There exists an i', $1 \leq i' \neq i \leq \ell$, such that $k' = k$, which happens with probability $(\ell-1)2^{-\kappa}$;
- Suppose for all i', $1 \leq i' \neq i \leq \ell$, $k' \neq k$ and F_k is a truly random function, we have that the probability that there *exists* an i', $1 \leq i' \neq i \leq \ell$, such that $I = F_k^0(ctr_i||pad_1) = F_{k'}^0(ctr_{i'}||pad_1)$ is also $(\ell-1)2^{-\kappa}$. Due to the pseudorandomness of the underlying PRF F_k, by straightforward calculation, we have that the probability that there *exists* an i', $1 \leq i' \neq i \leq \ell$, such that $I = F_k^0(ctr_i||pad_1) = F_{k'}^0(ctr_{i'}||pad_1)$ and $k \neq k'$, is at most $(\ell-1)2^{-\kappa}+\epsilon$, where ϵ is a negligible quantity in κ.

Putting all together, we conclude that the probability that Case-1 or Case-2 occurs is *at most* $\hat{\epsilon} = 2(\ell-1)2^{-\kappa} + \epsilon$, which is still negligible in κ.

Also note that conditioned on R correctly identifies $\mathcal{T}_i$ in its j-th session, the third-round message r_R will be $F_k^1(c||I)$ upon which $\mathcal{T}_i$ will output "accept" (i.e., $o_{\mathcal{T}_i}^c = 1$). We conclude that the event E occurs with probability at most $\hat{\epsilon}$, which is negligible in κ.

B.2 Mutual Authentication

Assuming there is a CMIM adversary $\mathcal{A}$ that breaks the tag-to-reader (resp., reader-to-tag) authentication of the RFID protocol depicted in Figure 5, we show how to construct another algorithm $\mathcal{A}'$ that breaks the pseudorandomness of the underlying PRF.

$\mathcal{A}'$ has oracle access to a PRF F_k or a truly random function $H : \{0,1\}^{2\kappa} \to \{0,1\}^{2\kappa}$, where $k \in_R \{0,1\}^\kappa$ is a random seed, and works as follows:

1. Run $\mathsf{Setup}(\kappa, \ell)$ to setup an RFID system $(R, \mathcal{T})$.
2. Randomly take $i \in_R \{1, \cdots, \ell\}$. That is, $\mathcal{A}'$ randomly guesses the target tag $\mathcal{T}_i$, with respect to which the event E_1 defined for tag-to-reader authentication (resp., E_2 defined for reader-to-tag authentication) occurs.
3. Corrupt all tags in $\mathcal{T}$ other than $\mathcal{T}_i$. Note that, by this, $\mathcal{A}'$ can perfectly mimic all the actions of the tags in

$\mathcal{T} - \{\mathcal{T}_i\}$. Moreover, $\mathcal{A}'$ gets and maintains all the records of the database DB of the reader R *except the record corresponding to* $\mathcal{T}_i$.

4. Run $\mathcal{A}$ and perfectly answer all oracle queries (made by $\mathcal{A}$) directed to tags in $\mathcal{T} - \{\mathcal{T}_i\}$, and handle oracle queries directed to R and $\mathcal{T}_i$ as follows (in particular, $\mathcal{A}'$ runs a counter ctr for simulating $\mathcal{T}_i$ that is initialized to 1):
   (a) On query $\mathsf{InitReader}()$, initiate a new session and return back a random string $c \in_R \{0,1\}^\kappa$. Here, c also serves as the session identifier.
   (b) On query $\mathsf{SendT}(\mathcal{T}_i, \hat{c})$ where $\hat{c} \in \{0,1\}^\kappa$ and $\mathcal{T}_i$ (simulated by $\mathcal{A}'$) is waiting for the first-round message of a new session, initiate a new session with $\hat{c}$ as the session identifier, query its oracle (i.e., F_k or H) with $ctr||pad_1$ to get I and then query its oracle with $\hat{c}||I$ to get (r_0, r_1), compute $r_\mathcal{T} = r_0 \oplus ctr||pad_2$, keep r_1 for this incomplete session, return $(I, r_\mathcal{T})$ (as the second-round message) and update $ctr = ctr + 1$.
   (c) On query $\mathsf{SendR}(\hat{c}, (\hat{I}, \hat{r}_\mathcal{T}))$ (where $\hat{c}$ is supposed to be a session identifier), first check whether it is keeping a session (simulated for R) of the session identifier $\hat{c}$ and is waiting for the second-round message: If not, return a special symbol "$\perp$" indicating an invalid query; if yes (which means that $\hat{c}$ is some random string ever sent by $\mathcal{A}'$ as the first-round message), mimic the actions of R as follows: Any computation involving records of tags in $\mathcal{T} - \{\mathcal{T}_i\}$ in the database DB (actually maintained by $\mathcal{A}'$) is performed by $\mathcal{A}'$ itself, and the computation involving the database record of $\mathcal{T}_i$ is performed by $\mathcal{A}'$ with the aid of its oracle. Note that $\mathcal{A}'$ actually does not necessarily know the actual identities of the tags in $\mathcal{T}$ to simulate R in this case.
   (d) On query $\mathsf{SendT}(\mathcal{T}_i, \hat{r}_R)$ where $\hat{r}_R \in \{0,1\}^\kappa$ and $\mathcal{T}_i$ (simulated by $\mathcal{A}'$) is currently running an incomplete session of session-identifier $\hat{c}$ and is waiting for the third-round message, retrieve the value r_1 (generated when generating the second-round message of the session $\hat{c}$) and check whether $\hat{r}_R = r_1$: If yes, complete the session $\hat{c}$ and return "accept"; otherwise, complete the session and return "reject".
   (e) On query $\mathsf{Corrupt}(\mathcal{T}_i)$ (i.e., $\mathcal{A}$ tries to corrupt $\mathcal{T}_i$), stop and output "failure".
5. If $\mathcal{A}'$ did not output "failure" in the above step (e) due to request by $\mathcal{A}$ to corrupt $\mathcal{T}_i$, stop whenever $\mathcal{A}$ stops, and then output "1" if the event E_1 (resp., E_2) defined in Definition 3.3 occurs.

We first note that the running time of $\mathcal{A}'$, denoted t', is polynomially related to that of $\mathcal{A}$. Furthermore, suppose the oracle accessed by $\mathcal{A}'$ is the PRF F_k and $\mathcal{A}'$ did not stop and output "failure" at step (e) due to corruption query of $\mathcal{T}_i$, the view of $\mathcal{A}$ in its real attack and the view of $\mathcal{A}$ under the simulation of $\mathcal{A}'$ are identical. As $\mathcal{A}'$ uniformly selects the uncorrupted tag $\mathcal{T}_i$ and the view of $\mathcal{A}$ is independent of the choice of i when $\mathcal{A}'$ gets oracle access to F_k, we have that: suppose $\mathcal{A}$ $(\epsilon, t, n_1, n_2, n_3, n_4)$-breaks the tag-to-reader (resp., reader-to-tag) authentication and $\mathcal{A}'$ gets oracle access to the PRF F_k, with probability $\ell^{-1}\epsilon$ the event E_1 (resp., E_2) occurs (under the simulation of $\mathcal{A}'$) *w.r.t. the uncorrupted tag* $\mathcal{T}_i$ (selected by $\mathcal{A}'$), *on which* $\mathcal{A}'$ *will output "1"*.

Recall that the event E_1 is defined as $\mathcal{A}$ successfully finishes a session with the reader R of the session transcript $(c, I||r_\mathcal{T}, r_R)$, in which R identified $\mathcal{A}$ as some uncorrupted tag $\mathcal{T}_i$, but no matching session exists at the side of the tag $\mathcal{T}_i$. That is, no session of transcript prefix $(c, I||r_\mathcal{T})$ was ever run or is now running by $\mathcal{T}_i$, where $I = F_k^0(ctr||pad_1)$ and $r_\mathcal{T} = F_k^0(c||I) \oplus (ctr||pad_2)$ for some counter value ctr. The event E_2 is defined as $\mathcal{A}$ successfully finishes a session with an uncorrupted tag $\mathcal{T}_i$ of the session transcript $(c, I||r_\mathcal{T}, r_R)$, in which $\mathcal{T}_i$ outputs "accept", but no matching session *of the identical session transcript* exists at the side of the reader R.

Now, we consider the probability that $\mathcal{A}'$ outputs "1" when having oracle access to a truly random function H, i.e., the probability that the event E_1 (resp., E_2) occurs w.r.t. $\mathcal{T}_i$ under the simulation of $\mathcal{A}'$ with oracle access to the truly random function H.

We first consider the probability the event E_1 occurs w.r.t. $\mathcal{T}_i$ under the simulation of $\mathcal{A}'$ with oracle access to H. Recall that, in this case, $I = H^0(ctr||pad_1)$ and $r_\mathcal{T} = H^0(c||I) \oplus (ctr||pad_2)$ for some ctr, where H^0 (resp., H^1) stands for the κ-bit prefix (resp., suffix) of the output of H.

- There exists no session of partial session transcript $(c, I||r'_\mathcal{T})$ at the side of $\mathcal{T}_i$ (in the simulation of $\mathcal{A}'$), where $r'_\mathcal{T} = H^0(c||I) \oplus (ctr'||pad_2)$ for any counter value ctr'. That is, $\mathcal{T}_i$ did not ever compute the value $H(c||I)$ before the event E_1 occurs. In this case, the view of $\mathcal{A}$ under the simulation of $\mathcal{A}'$ is independent of $r_\mathcal{T} = H^0(c||I) \oplus (ctr||pad_2)$ that is a truly random string in $\{0,1\}^\kappa$. Thus, the event E_1 occurs w.r.t. $\mathcal{T}_i$ with probability at most $n_3 2^{-\kappa}$, where n_3 is the upper-bound on the number of O_2 (i.e., SendR) oracle queries made by $\mathcal{A}$.
- For each session of partial session transcript $(c, I||r'_\mathcal{T})$ at the side of $\mathcal{T}_i$ (in the simulation of $\mathcal{A}'$), where $r'_\mathcal{T} = H^0(c||I) \oplus (ctr'||pad_2)$ and $I = H^0(ctr'||pad_1)$ for some counter value ctr', we observe that: (1) As

we assume no matching session exists at the side of $\mathcal{T}_i$, it must be that $r'_{\mathcal{T}} \neq r_{\mathcal{T}} = H^0(c||I) \oplus (ctr||pad_2)$ and thus $ctr' \neq ctr$; (2) But, the fact that $I = H^0(ctr||pad_1) = H^0(ctr'||pad_1)$ means that such a session exists with probability $2^{-\kappa}$. As we assume each tag involves at most s sessions, where $s \leq n_2$ and n_2 is the upper-bound of O_2 oracle queries made by $\mathcal{A}$, we conclude that the probability that there *exists* a session of partial session transcript $(c, I||r'_{\mathcal{T}})$ at the side of $\mathcal{T}_i$ (in the simulation of $\mathcal{A}'$), where $r'_{\mathcal{T}} = H^0(c||I) \oplus (ctr'||pad_2)$ and $I = H^0(ctr'||pad_1)$ for some counter value ctr', is at most $s \cdot 2^{-\kappa}$.

Putting all together, we have the probability that event E_1 occurs w.r.t. $\mathcal{T}_i$ under the simulation of $\mathcal{A}'$ when oracle accessing a truly random function, on which $\mathcal{A}'$ outputs "1", is at most $(n_3 + s)2^{-\kappa} \leq (n_3 + n_2)2^{-\kappa}$.

Now, we consider the probability that event E_2 occurs w.r.t. $\mathcal{T}_i$ under the simulation of $\mathcal{A}'$ with oracle access to H. Recall that, in this case, the event E_2 w.r.t. $\mathcal{T}_i$ is defined as $\mathcal{A}$ successfully finishes a session with the uncorrupted tag $\mathcal{T}_i$ of the session transcript $(c, I||r_{\mathcal{T}}, r_R)$, in which $\mathcal{T}_i$ outputs "accept", but no matching session *of the identical session transcript* exists at the side of the reader R, where $I = H^0(ctr||pad_1)$, $r_{\mathcal{T}} = H^0(c||I) \oplus (ctr||pad_2)$ and $r_R = H^1(c||I)$. We investigate several cases:

- R (simulated by $\mathcal{A}'$) did not send $r_R = H^1(c||I)$ (before the event E_2 occurs). In this case, the view of $\mathcal{A}$ (under the simulation of $\mathcal{A}'$) is independent of $r_R = H^1(c||I)$ that is a random string in $\{0,1\}^\kappa$. Thus, the probability that event E_2 occurs w.r.t. $\mathcal{T}_i$ is at most $s \cdot 2^{-\kappa}$, where $s(\leq n_2)$ is the upper-bound on the number of sessions involving $\mathcal{T}_i$.

- For each session run by R in which R sends $r_R = H^1(c||I)$ to $\mathcal{T}_i$, we observe that this session is of session transcript $(c, I||r'_{\mathcal{T}}, r_R))$, where $r'_{\mathcal{T}} = H^0(c||I) \oplus (ctr'||pad_2)$ and $I = H^0(ctr'||pad_1)$ for some counter value ctr'. As we assume no matching session exists at the side of R, we get $r'_{\mathcal{T}} = H^0(c||r) \oplus (ctr'||pad_2) \neq r_{\mathcal{T}} = H^0(c||r) \oplus (ctr||pad_2)$, and thus $ctr' \neq ctr$. However, as $\Pr[I = H^0(ctr||pad_1) = H^0(ctr'||pad_1) | ctr' \neq ctr] = 2^{-\kappa}$, this session happens with probability $2^{-\kappa}$. As $\mathcal{A}$ makes at most n_3 SendR (i.e., O_3) queries, the probability that there exists a session at the side of R in which R sends $H^1(c||r)$ is at most $n_3 \cdot 2^{-\kappa}$.

Putting all together, we have that the probability E_2 occurs w.r.t. $\mathcal{T}_i$ under the simulation of $\mathcal{A}'$ when oracle accessing a truly random function, on which $\mathcal{A}'$ outputs "1", is also at most $(s + n_3)2^{-\kappa} \leq (n_2 + n_3)2^{-\kappa}$.

Recall that, suppose $\mathcal{A}$ $(\epsilon, t, n_1, n_2, n_3, n_4)$-breaks the tag-to-reader authentication, we have shown that $\mathcal{A}'$ outputs "1" with probability at least $\ell^{-1}\epsilon$ when oracle accessing the PRF F_k. Denote by $\epsilon_1 = |\ell^{-1}\epsilon - (s + n_3)2^{-\kappa}|$. Suppose $\mathcal{A}$ $(\epsilon, t, n_1, n_2, n_3, n_4)$-breaks the tag-to-reader or reader-to-tag authentication, we conclude that $\mathcal{A}'$ can (t', ϵ_1)-break the pseudorandomness of the underlying PRF, where t' is polynomially related to t. As we assume the underlying PRF is secure, i.e., ϵ_1 is negligible for any t' that is polynomial in κ, we have that ϵ must also be negligible for any t-time adversary $\mathcal{A}$ where t is polynomial in κ. This establishes both the tag-to-reader and the reader-to-tag authentication of the protocol depicted in Figure 5.

B.3 ZK-Privacy

According to the zk-privacy formulation presented in Section 4, after an RFID system $(R, \mathcal{T})$ is setup by $\mathsf{Setup}(\kappa, \ell)$, for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ works as follows. In the *first stage* of $\mathcal{S}$, $\mathcal{S}_1$ runs $\mathcal{A}_1$ as a subroutine, and concurrently interacts with R and all the tags in $\mathcal{T}$ (via the four oracles in $\mathcal{O} = \{O_1, O_2, O_3, O_4\}$). For all oracle queries made by $\mathcal{A}_1$, $\mathcal{S}_1$ makes the same oracle queries, and relays back the oracle answers to $\mathcal{A}_1$. Finally, $\mathcal{S}_1$ outputs whatever $\mathcal{A}_1$ outputs at the end of the first stage, say $(\mathcal{C}, st)$, where $\mathcal{C} = \{T_{i_1}, T_{i_2}, \cdots, T_{i_\delta}\} \subseteq \mathcal{T}$ is a set of *clean* tags, $0 \leq \delta \leq \ell$, and st is some state information to be transmitted to the second stage. As for any adversary $\mathcal{A}$ who outputs an empty set $\mathcal{C}$ of clean tags (i.e., no challenge tag will be chosen), the view of $\mathcal{A}$ can be trivially perfectly simulated by the simulator with oracle access to all the tags and the reader R. In the following analysis, we focus on the case that $|\mathcal{C}| \geq 1$, i.e., $1 \leq \delta \leq \ell$.

Then, a value g is selected uniformly at random from $\{1, \cdots, \delta\}$, which specifies the challenge tag $\mathcal{T}_g = \mathcal{T}_{i_g}$. In the *second stage* of $\mathcal{S}$, $\mathcal{S}_2$ runs $\mathcal{A}_2(st)$ as a subroutine and concurrently interacts with R and the tags in $\widehat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$. For all oracle queries made by $\mathcal{A}_2$ directed to tags in $\widehat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$, $\mathcal{S}_2$ makes the same oracle queries, and relays back oracle answers to $\mathcal{A}_2$. However, for oracle queries made by $\mathcal{A}_2$ directed to R and to $\mathcal{T}_g = \mathcal{T}_{i_g}$ (blindly accessed by $\mathcal{A}_2$), $\mathcal{S}_2$ works as follows:

1. On oracle query $\mathsf{InitReader}()$ made by $\mathcal{A}_2$, make the same oracle query to R, get back a random string $c \in \{0,1\}^\kappa$ from R and relay c back to $\mathcal{A}_2$.

2. On oracle query $\mathsf{SendT}(challenge, \hat{c})$, where the challenge tag $\mathcal{T}_g$ (simulated by $\mathcal{S}_2$) currently does not run any session (which means $\hat{c}$ will be treated as the first-round message), open a session for $\mathcal{T}_g$ with $\hat{c}$ as the first-round message (that also serves as the session-identifier of this new session), randomly select $I, r_{\mathcal{T}} \in_R \{0,1\}^\kappa$, and send back $I||r_{\mathcal{T}}$ to $\mathcal{A}_2$ as the second-round message.

3. On oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$, work as follows:

   Case-3.1. If $\hat{I}||\hat{r}_{\mathcal{T}}$ was sent by $\mathcal{T}_g$ (simulated by $\mathcal{S}_2$) in a session of session-identifier $\hat{c}$ (i.e., the first-round message), $\mathcal{S}_2$ simulates the responses of R according to the following two cases. Note that $\mathcal{S}_2$ does not make the oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$ to R. As $\hat{I}$ and $\hat{r}_{\mathcal{T}}$ are truly random strings, if $\mathcal{S}_2$ makes the same oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$ to R, with overwhelming probability R will abort this session and outputs "0" indicating "reject".

   Case-3.1.1 If R is running an incomplete session of session-identifier $\hat{c}$ (i.e., $\hat{c}$ was sent by R upon an $\mathsf{InitReader}$ query and R is waiting for the second-round message), $\mathcal{S}_2$ just returns back a random string $r_R \in_R \{0,1\}^{\kappa}$ to $\mathcal{A}_2$, and outputs "1" indicating "accept".

   Case-3.1.2. Otherwise, $\mathcal{S}_2$ simply returns back a special symbol "$\perp$" indicating invalid query.

   Case-3.2. In all other cases, $\mathcal{S}_2$ makes the same oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$ to R, and relays back the answer from R to $\mathcal{A}_2$.

4. On oracle query $\mathsf{SendT}(challenge, \hat{r}_R)$, where the challenge tag $\mathcal{T}_g$ (simulated by $\mathcal{S}_2$) currently runs a session of partial session-transcript $(\hat{c}, I||r_{\mathcal{T}})$ and is waiting for the third-round message, work as follows:

   Case-4.1. If there exists a matching session of the same session transcript $(\hat{c}, I||r_{\mathcal{T}}, \hat{r}_R)$ at the side of R (where $\hat{r}_R$ may be simulated by $\mathcal{S}_2$ as in the above Case-3.1), $\mathcal{S}_2$ outputs "1" indicating "accept";

   Case-4.2. Otherwise, $\mathcal{S}_2$ simply outputs "0" indicating "reject".

5. Output of $\mathcal{S}_2$: Finally, stop whenever $\mathcal{A}_2$ stops, and output the simulated view, denoted $sview(\kappa, \ell)$, which consists of the system public parameters $para$ (output by $\mathsf{Setup}(\kappa, \ell)$), the random coins used by $\mathcal{A}$ (actually set by $\mathcal{S}$), all oracle answers (including ones provided by the real protocol participants and ones simulated by $\mathcal{S}_2$) to queries made by $\mathcal{A}$. Recall that the output of the experiment $\mathbf{Exp}_{\mathcal{S}}^{zkp}[\kappa,\ell]$ is defined as $(g, sview(\kappa, \ell))$.

It is clear that if $\mathcal{A}$ works in polynomial-time, $\mathcal{S}$ works also in polynomial-time. For all sufficiently large κ, any ℓ (that is polynomial in κ), any PPT adversary $\mathcal{A}$, and any polynomial-time distinguisher D, denote by $p_{\mathcal{S}} = \Pr[D(\kappa, \ell, g, sview(\kappa, \ell)) = 1]$ and by $p_{\mathcal{A}} = \Pr[D(\kappa, \ell, g, view_{\mathcal{A}}(\kappa, \ell)) = 1]$. Below, we show that $|p_{\mathcal{S}} - p_{\mathcal{A}}|$ is negligible, which establishes the zk-privacy property.

For any (κ, ℓ) and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we first consider a mental experiment $\mathbf{Exp}_{\hat{S}}^{prf}(\kappa, \ell)$ w.r.t. a PPT algorithm $\hat{S}$. $\hat{S}$ has oracle access to a PRF F_k or a truly random function $H : \{0,1\}^{2\kappa} \to \{0,1\}^{2\kappa}$, and works as follows:

Step-1. Select $i \in_R \{1, \cdots, \ell\}$ uniformly at random.

Step-2. Setup $\ell - 1$ tags, denoted as $\mathcal{T}' = \{\mathcal{T}_1, \cdots, \mathcal{T}_{i-1}, \mathcal{T}_{i+1}, \cdots, \mathcal{T}_{\ell}\}$, with secret-keys $k_1, \cdots, k_{i-1}, k_{i+1}, \cdots, k_{\ell}$ and counters $ctr_1, \cdots, ctr_{i-1}, ctr_{i+1}, \cdots, ctr_{\ell}$ respectively, where each secret-key is a random string of length κ and each counter is initialized to "1"; maintain a counter ctr_i to simulate $\mathcal{T}_i$ with the aid of its oracle (i.e., F_k or H); setup and maintain all the database records (of the reader R) w.r.t. the tags in $\mathcal{T}'$.

Step-3. Run $\mathcal{A}_1$ as a subroutine, and answer the oracle queries made by $\mathcal{A}_1$ as follows:

- For oracle queries directed to tags in $\mathcal{T}'$, $\hat{S}$ computes and gives the answers by itself. Note that $\hat{S}$ can perfectly simulate the actions of tags in $\mathcal{T}'$.
- For oracle queries directed to the tag $\mathcal{T}_i$, $\hat{S}$ mimics the actions of $\mathcal{T}_i$, and provides the oracle answer, with the aid of its oracle.
- For any oracle query directed to the reader R, $\hat{S}$ works as follows: If the oracle query is $\mathsf{InitReader}()$, $\hat{S}$ simply returns back a random string $c \in_R \{0,1\}^{\kappa}$. If the oracle query is $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$, $\hat{S}$ mimics the actions of R with the following modifications: the computation involving database records w.r.t. tags in $\mathcal{T}'$ is performed by $\hat{S}$ itself; the computation involving the record w.r.t. $\mathcal{T}_i$ is performed by $\hat{S}$ with the aid of its oracle.

Step-4. When $\mathcal{A}_1$ stops and outputs $(st, \mathcal{C})$, where $\mathcal{C} = \{\mathcal{T}_{i_1}, \cdots, \mathcal{T}_{i_\delta}\}$ is a set of clean tags, $1 \le \delta \le \ell$, select $g \in_R \{1, \cdots, \delta\}$ uniformly at random. If the challenge-tag $\mathcal{T}_g (= \mathcal{T}_{i_g}) \neq \mathcal{T}_i$, stop and output "$\perp$" indicating "failure"; otherwise (i.e., $\mathcal{T}_g = \mathcal{T}_i$), proceed to the next step.

Step-5. In case $\mathcal{T}_g = \mathcal{T}_i$, run $\mathcal{A}_2(st)$ as a subroutine and mimic the actions of $\mathcal{S}_2$, but with the following modifications: (Recall that $\mathcal{T}_i$ is just the challenge tag $\mathcal{T}_g$ in this case.)

- For oracle queries made by $\mathcal{A}_2$ to tags in $\hat{\mathcal{T}} = \mathcal{T} - \mathcal{C}$, $\hat{S}$ computes and gives the answers by itself. Note that $\hat{S}$ can perfectly simulate the actions of tags in $\mathcal{T}' \supseteq \hat{\mathcal{T}}$.

- For oracle queries directed to the challenge tag $\mathcal{T}_g = \mathcal{T}_i$ of the form $\mathsf{SendT}(challenge, \cdot)$, $\hat{S}$ provides the answer, mimics the actions of $\mathcal{T}_i$, with the aid of its oracle.
- For oracle queries directed to R, $\hat{S}$ works in the same way as in its first stage.
- Output of $\hat{S}$: Whenever $\mathcal{A}_2$ stops, $\hat{S}$ also stops and then gets a view, denoted $view_{\mathcal{A}}^{\hat{S}}(\kappa, \ell)$, which consists of the system public parameters $para$ (generated by $\hat{S}$ itself), the random coins used by $\mathcal{A}$ (actually set by $\hat{S}$), all oracle answers (provided by $\hat{S}$ itself to queries made by $\mathcal{A}$). Finally, $\hat{S}$ runs the assumed distinguisher D on inputs $(\kappa, \ell, g, view_{\mathcal{A}}^{\hat{S}}(\kappa, \ell))$, and outputs whatever $D(\kappa, \ell, g, view_{\mathcal{A}}^{\hat{S}}(\kappa, \ell))$ outputs.

Denote by $p_{\hat{S}}^{F_k}$ (resp., $p_{\hat{S}}^{H}$) the probability that $\hat{S}$ outputs "1", with oracle access to F_k (resp., H). Note that $\hat{S}$ outputting "1" implies $\mathcal{T}_g = \mathcal{T}_i$ (as otherwise $\hat{S}$ will output "$\perp$" indicating failure). By the pseudorandomness of F_k, we get that for some quantity $\epsilon_{\hat{S}}$ that is negligible in κ, it holds that (for sufficiently large κ):

$$|p_{\hat{S}}^{F_k} - p_{\hat{S}}^{H}| = \epsilon_{\hat{S}} \tag{1}$$

Note also that the real view of $\mathcal{A}$, i.e., $view_{\mathcal{A}}$, and the view of $\mathcal{A}$ under the simulation of $\hat{S}$ in the experiment $\mathbf{Exp}_{\hat{S}}^{prf}(\kappa, \ell)$, i.e., $view_{\mathcal{A}}^{\hat{S}}$, are identical *conditioned on $\hat{S}$ gets oracle access to the PRF F_k and $\hat{S}$ did not output "$\perp$" at Step-4 (i.e., $\mathcal{T}_g = \mathcal{T}_i$)*. In particular, the view (particularly, the output of $\mathcal{C}$) of $\mathcal{A}_1$ is independent of the choices of i in this case, and thus $\Pr[\mathcal{T}_i = \mathcal{T}_g] = \frac{1}{\ell}$, and we have:

$$p_{\hat{S}}^{F_k} = \frac{1}{\ell} p_{\mathcal{A}} \tag{2}$$

The above analysis also implies that the adaptive completeness and mutual authentication properties hold too in the experiment $\mathbf{Exp}_{\hat{S}}^{prf}(\kappa, \ell)$, whether $\hat{S}$ gets oracle access to the PRF F_k or a truly random function H; Otherwise, the pseudorandomness property of the underlying PRF F_k will be violated.

Now, we consider another experiment $\mathbf{Exp}_{S'}^{prf}(\kappa, \ell)$, in which a PPT algorithm S', with oracle access to the PRF F_k or a truly random function H, mimics the actions of $\hat{S}$ until Step-5 in the experiment $\mathbf{Exp}_{\hat{S}}^{prf}(\kappa, \ell)$, but with the following modifications at Step-5 (recall that Step-5 is for case of $\mathcal{T}_g = \mathcal{T}_i$):

D1. For any oracle query directed to the challenge tag $\mathcal{T}_g = \mathcal{T}_i$ of the form $\mathsf{SendT}(challenge, \hat{c})$, $\hat{S}$ mimics the actions of the simulator $\mathcal{S}$ and returns back $I, r_{\mathcal{T}} \in_R \{0,1\}^{\kappa}$, where $I, r_{\mathcal{T}}$ are taken randomly by S' itself.

D2. For any oracle query directed to R of the form $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$, S' works as follows:

Case-D2.1 In Case-3.1 (of the simulation of $\mathcal{S}$), S' just mimics the actions of the simulator $\mathcal{S}$.

Case-D2.2 In Case-3.2 (of the simulation of $\mathcal{S}$), S' simulates the actions of R as follows:

Case-D.2.2.1. The computation involving database records w.r.t. tags in $\mathcal{T}'$ is performed by S' itself (just as $\hat{S}$ does), which perfectly mimics the actions of the reader R in this case.

Case-D.2.2.2. But, the computation involving the record w.r.t. $\mathcal{T}_i$ is just *ignored* by S'.

D3. For any oracle query directed to $\mathcal{T}_g = \mathcal{T}_i$ of the form $\mathsf{SendT}(challenge, \hat{r}_R)$, S' just mimics the actions of the simulator $\mathcal{S}$ (in Case-4.1 and Case-4.2 of the simulation of $\mathcal{S}$).

Note that S' only queries its oracle (i.e., F_k or H) in its first-stage of simulation (in dealing with $\mathcal{A}_1$), but never queries its oracle in the second-stage of the simulation (in dealing with $\mathcal{A}_2$).

Denote by $p_{S'}^{F_k}$ (resp., $p_{S'}^{H}$) the probability that S' outputs "1", with oracle access to F_k (resp., H) in the experiment $\mathbf{Exp}_{S'}^{prf}$. By the pseudorandomness of F_k, we get that for some quantity $\epsilon_{S'}$ that is negligible in κ and for sufficiently large κ:

$$|p_{S'}^{F_k} - p_{S'}^{H}| = \epsilon_{S'} \tag{3}$$

Now, we investigate the differences between the view of $\mathcal{A}$ under the simulation of $\mathcal{S}$ and the view of $\mathcal{A}$ under the simulation of S', *conditioned on S' gets oracle access to the PRF F_k and S' does not output "$\perp$" at Step-4 (i.e., $\mathcal{T}_g = \mathcal{T}_i$)*. The only difference is that in Case-D.2.2.2 above, the computation involving the record w.r.t. $\mathcal{T}_i = \mathcal{T}_g$ is always ignored by S'; but in the simulation of $\mathcal{S}$ this computation is performed by the reader R. The observation here is Case-D.2.2 (that corresponds to Case-3.2 of the simulation of $\mathcal{S}$) means that, when $\mathcal{A}_2$ makes the oracle query $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$, $\mathcal{T}_g = \mathcal{T}_i$ did not run or is running a session of the (partial) session transcript $(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$. Just analogue to the analysis of tag-to-reader authentication (presented in Section B.2), it is straightforward to calculate that the probability the reader R successfully identifies $\mathcal{T}_i$ in Case-3.2 of the simulation of $\mathcal{S}$ is negligible. Under this observation and by noting that the view of $\mathcal{A}_1$ is independent of the choice of i when S' gets oracle access to F_k (and thus $\Pr[\mathcal{T}_g = \mathcal{T}_i] = \frac{1}{\ell}$), we have that for some quantity ϵ_S that is negligible in κ, it holds that (for sufficiently large κ):

$$p_{S'}^{F_k} = \frac{1}{\ell} p_{\mathcal{S}} + \epsilon_S \tag{4}$$

Now, we investigate the differences between the view of $\mathcal{A}$ under the simulation of $\hat{S}$ and that under the simulation of S', when both $\hat{S}$ and S' get access to a truly random function H. We have the following observations:

- In Case-D.1 (resp., Case-D2.1), S' always returns back random and independent $I, r_{\mathcal{T}}$ (resp., r_R). As $\mathcal{T}_i$ always generates I with the updated counter value, the value I returned back by $\hat{S}$ in Case-D.1, with the aid of the truly random function H, is also always uniformly distributed over $\{0,1\}^\kappa$ and is independent of what previously sent by $\hat{S}$. But the value $r_{\mathcal{T}}$ (resp., r_R) returned back by $\hat{S}$ in Case-D.1 (resp., Case-D.2.1), with the aid of the truly random function H, might not be independent of the values previously sent by $\hat{S}$, for the following reasons: (1) In case I is equal to some I-values previously sent by $\hat{S}$, the value $H(c||I) = (r_0, r_1)$ (that determines $r_{\mathcal{T}} = r_0 \oplus ctr||pad_2$ and $r_R = r_1$) can be previously defined. Note that the adversary $\mathcal{A}$ can use the same $\hat{c}$ for all sessions with the tags. This event occurs with probability at most $s \cdot 2^{-\kappa}$, where s is the upper-bound of sessions involved by $\mathcal{T}_i$; (2) For Case-D.2.1, upon the request $\mathsf{SendR}(\hat{c}, \hat{I}||\hat{r}_{\mathcal{T}})$ where $\hat{I}||\hat{r}_{\mathcal{T}}$ was sent by $\mathcal{T}_g = \mathcal{T}_i$ (simulated by $\hat{S}$) in session $\hat{c}$, the reader R (simulated by $\hat{S}$) may identify a tag $\mathcal{T}_{i'} \neq \mathcal{T}_i$, and thus the value r_R returned by $\hat{S}$ may be $F^1_{k'}(\hat{c}||I)$ (rather than an independent and random string in $\{0,1\}^\kappa$), where k' is the secret-key of $\mathcal{T}_{i'}$. But, as the adaptive completeness property holds in the experiment $\mathbf{Exp}^{prf}_{\hat{S}}(\kappa, \ell)$, this event occurs also with negligible probability.

- By the mutual authentication (specifically, the tag-to-reader authentication in this case) of the experiment $\mathbf{Exp}^{prf}_{\hat{S}}(\kappa, \ell)$, the difference caused by Case-D.2.2.2 in $\mathbf{Exp}^{prf}_{S'}(\kappa, \ell)$ occurs also with negligible probability.

- In Case-D.3 in $\mathbf{Exp}^{prf}_{S'}(\kappa, \ell)$ (that corresponds to Case-4.1 and Case-4.2 of the simulation of $\mathcal{S}$) , $\mathcal{T}_g = \mathcal{T}_i$ (simulated by S') always outputs "accept" (resp., "reject") in Case-4.1 when existing matching session at the side of R (resp., Case-4.2 when existing no matching session at R). But, in the simulation of $\hat{S}$ in $\mathbf{Exp}^{prf}_{\hat{S}}(\kappa, \ell)$, with oracle access to a truly random function H, $\mathcal{T}_g = \mathcal{T}_i$ may output "reject" in Case-4.1 (resp., "accept" in Case-4.2).

  We first observe that, by the mutual authentication (specifically, the reader-to-tag authentication in this case) of the experiment $\mathbf{Exp}^{prf}_{\hat{S}}(\kappa, \ell)$, the probability that $\mathcal{T}_g = \mathcal{T}_i$ simulated by $\hat{S}$, with oracle access to the truly random function H, outputs "accept" in Case-4.2 is negligible.

  Next, note that $\mathcal{T}_g = \mathcal{T}_i$ (simulated by $\hat{S}$ in $\mathbf{Exp}^{prf}_{\hat{S}}(\kappa, \ell)$) outputs "reject" in Case-4.1, only if the reader R (simulated by $\hat{S}$) identified a tag $\mathcal{T}_{i'} \neq \mathcal{T}_i$ upon the query $\mathsf{SendR}(\hat{c}, I||r_{\mathcal{T}})$, where $I||r_{\mathcal{T}}$ was sent by $\mathcal{T}_i$ in the session $\hat{c}$. But, by the adaptive completeness of the experiment $\mathbf{Exp}^{prf}_{\hat{S}}(\kappa, \ell)$, this event occurs also with negligible probability.

From the above analysis, we have that for some quantity ϵ_H that is negligible in κ, it holds (for sufficiently large κ):

$$|p^H_{S'} - p^H_{\hat{S}}| = \epsilon_H \tag{5}$$

Combining all the equations (1), (2), (3), (4), (5), we have $|p^{F_k}_{\hat{S}} - p^H_{\hat{S}}| = |\ell^{-1}p_{\mathcal{A}} - p^H_{\hat{S}}| = |\ell^{-1}p_{\mathcal{A}} - p^H_{S'} \pm \epsilon_H| = |\ell^{-1}p_{\mathcal{A}} - p^{F_k}_{S'} \pm \epsilon_{S'} \pm \epsilon_H| = |\ell^{-1}p_{\mathcal{A}} - \ell^{-1}p_S \pm \epsilon_S \pm \epsilon_{S'} \pm \epsilon_H| = \epsilon_{\hat{S}}$. From this equation, we conclude that $|p_{\mathcal{A}} - p_S|$ must be negligible, which then establishes the zk-privacy property of the protocol depicted in Figure 5.