PPS: Privacy-Preserving Statistics using RFID Tags

Erik-Oliver Blass[1,2] Kaoutar Elkhiyaoui[1] Refik Molva[1]
[1]EURECOM, Sophia Antipolis, France
[2]College of Computer and Information Science, Northeastern University, Boston, MA 02115

***Abstract*—As RFID applications are entering our daily life, many new security and privacy challenges arise. However, current research in RFID security focuses mainly on simple authentication and privacy-preserving identification. In this paper, we discuss the possibility of widening the scope of RFID security and privacy by introducing a new application scenario. The suggested application consists of computing statistics on private properties of individuals stored in RFID tags. The main requirement is to compute global statistics while preserving the privacy of individual readings. *PPS* assures the privacy of properties stored in each tag through the combination of homomorphic encryption and aggregation at the readers. Re-encryption is used to prevent tracking of users. The readers scan tags and forward the aggregate of their encrypted readings to the back-end server. The back-end server then decrypts the aggregates it receives and updates the global statistics accordingly. PPS is provably privacy-preserving. Moreover, tags can be very simple as they are not required to perform any computation, but only to store data.**

I. Introduction

Radio Frequency Identification tags are low cost wireless devices that were aimed to identify products along the different steps of the supply chain without the need for line of sight. Due to its cost and time effectiveness, RFID tags have gained more and more popularity in many applications, such as access control, product tracking, item identification, and counterfeiting detection. Most of the work on RFID security and privacy has focused on lightweight authentication, identification, and formal security and privacy models in RFID settings [2, 4, 5, 8, 18, 21–23].

The paper at hand focuses on a new RFID application scenario which raises new requirements beyond the authentication and identification issues. We focus on the problem of collecting statistics over private *properties* of a population of individuals while assuring the privacy of these individuals with respect to their properties. A typical real-world application example of this problem is the computation of customer statistics using special RFID-based membership or loyalty cards. A main requirement is the protection of the customer's identity and privacy during computation.

To tackle this problem, we present a new protocol called *PPS* ("Privacy-Preserving Statistics"). In PPS, each RFID tag stores the properties of its holder in an encrypted form. Intermediate parties called *readers* collect encrypted properties from *tags*, compute aggregates over encrypted readings without decrypting them, and periodically forward the result of such aggregation operations to the *back-end server*. The server is then able to derive the global statistics by decrypting the aggregates it receives from the readers. PPS involves the following entities:

- **Issuer** I: the issuer initializes each tag by writing into the tag's memory an encrypted representation of the properties of the tag holder.
- **Tags** $\{T_i\}$: each tag stores an encrypted representation of different properties of the tag holder. More precisely, the encrypted representation of p different properties $P_i \in \{\text{true}, \text{false}\}$, $1 \leq i \leq p$, is stored in the tag, where P_i is set to "true", if the tag holder possesses the property P_i.
- **Readers** $\{R_i\}$: readers are in charge of collecting properties stored on tags. They read the data stored on each tag and forward the result of these readings to the back-end server.
- **A back-end server** S: S processes the aggregate data received from readers and derives some global statistics such as distribution of attendance rate with respect to event types and population characteristics.

The main requirement for S is to count the number of tag holders satisfying each property P_i for all the properties. The main concern is to gather statistics while preserving the *privacy* of tag holders. Neither readers nor the back-end server should be able to disclose the values of a tag holder's properties. The main intuition to ensure privacy in PPS is to combine both encryption and aggregation. The list ω_i of the tag holder properties are encrypted as $E(\omega_i)$ and stored on the tag. Through subsequent readings of tags in its range, the reader computes the aggregate of the ciphertexts received from the tags, $\sum E(\omega_i)$, and periodically forwards the encrypted aggregate value to the back-end server.

Since the back-end server S can decrypt, readers must aggregate the ciphertexts received from the tags in their range before forwarding the encrypted data to S. Note that forwarding each individual reading to the server would strongly overload typically embedded, low capacity readers.

Even though the privacy of properties is assured through encryption, *unlinkability* of tags, as defined by Chatmon et al. [7], has to be assured, too. Unlinkability prevents the readers or eavesdropping adversaries from tracking tags over different sessions; an adversary should never be able to link two responses of the same tag over different sessions. Therefore, data sent by the tags should be different at each reading. Re-encryption is used to that effect. In Section II-B, we formally define the notion of unlinkability.

The **major contributions** of PPS are:

1.) contrary to related work on RFID tag identification, PPS provides an RFID-based mechanism to collect statistics over a set of properties in a privacy-preserving manner.

2.) formal proofs of *privacy* and *unlinkability* against eavesdroppers, malicious readers, and curious back-end servers.

3.) minimal hardware requirements resulting in cheap tags: PPS does not require tags to do *any cryptographic computation*, tags are passive, i.e., battery-less and only require data storage functions. Contrary to related work, PPS' storage-only requirements enable implementations on today's available EPC class1 Gen2 tags.

4.) data integrity: PPS can detect tag tampering.

II. Adversary & Privacy Models

In this section, we introduce the adversary model and define formally the notions of privacy and unlinkability.

A. Adversary model

PPS protects against two different categories of adversaries $\mathcal{ADV}_1$ represents external adversaries and malicious readers. We assume an active adversary who can not only eavesdrop messages, but also intercept, modify, and even initiate communication. He might even replace a tag's content by re-writing it. (Re-writing tags has some special implications on security, and we discuss this issue separately in Section V.)

$\mathcal{ADV}_2$ represents a malicious back-end server. A back-end server is passive in the sense that it only receives aggregates from readers. It cannot initiate communication with tags.

$\mathcal{ADV}_1$ does not collude with $\mathcal{ADV}_2$. Note that in scenarios where the readers and the back-end server collude, PPS will not provide privacy.

As motivated in the introduction, the primary goal of $\mathcal{ADV}_1$ or $\mathcal{ADV}_2$ is to gain some knowledge about sensitive information, in this case individual tag holders' properties. We formalize this below.

B. Privacy Models

PPS borrows privacy notions for storage-only tags as originally proposed by Ateniese et al. [1], Golle et al. [11], and experiment-based definitions by Juels and Weis [13].

At the end of a protocol execution, PPS is said to be privacy-preserving, if $\mathcal{ADV}_1$ and $\mathcal{ADV}_2$ can neither decide which properties a given tag satisfies nor link tags to previous protocol executions. In conclusion, an adversary should not have a higher chance in breaking privacy or unlinkability than simple guessing. The following oracle-like constructions exist:

$\mathcal{O}_{\text{pick}}$ is an oracle that randomly selects a tag T_i from all the n tags in the system.

$\mathcal{O}_{\text{select}}$ is an oracle that randomly returns a tag T_i from all the n tags in the system along the list S_i of properties P_j T_i is satisfying.

$\mathcal{O}_{\text{flip}}$ is an oracle that, provided with two tags T_0, T_1, randomly chooses $b \in \{0, 1\}$ and returns T_b.

$\mathcal{O}_{\text{aggregate}}$ computes a total of s aggregates Agg_1, Agg_2, $\ldots$, Agg_s, each time by randomly choosing a set of γ tags: Agg_1 is computed using tags $(T_1^1, T_1^2, \ldots, T_1^\gamma)$, Agg_2 is computed using $(T_2^1, T_2^2, \ldots, T_2^\gamma)$, $\ldots$, Agg_s is computed using $(T_s^1, T_s^2, \ldots, T_s^\gamma)$. The sets of tags are chosen randomly, but there is at least one tag that is an element of two different sets, i.e., used in the computation of two different aggregates. Finally, $\mathcal{O}_{\text{aggregate}}$ returns Agg_1, Agg_2, $\ldots$, Agg_s.

1) Privacy against $\mathcal{ADV}_1$: An adversary breaks the privacy of PPS, if given a tag T and a property P_i, he can decide if a tag T satisfies the property P_i or not.

To that effect, an adversary $\mathcal{ADV}_1$ has access to tags in two phases. In a learning phase (Algorithm 1), $\mathcal{ADV}_1$ is provided with a challenge tag T_c from the oracle $\mathcal{O}_{\text{pick}}$. He can read from T_c for a maximum of t times. $\mathcal{O}_{\text{select}}$ gives $r-1$ tags to $\mathcal{ADV}_1$ along with the list S_i of properties P_j that each tag T_i satisfies. $\mathcal{ADV}_1$ can read and write into T_i for a maximum of s times. After each read or write access to a tag in the learning phase, the tag is allowed to interact with a legitimate reader by a normal PPS protocol run ("EXECUTE").

```
Tc ← O_pick;
for i := 1 to t do
    READ(Tc);
    EXECUTE(Tc);
end
for i := 1 to r − 1 do
    (Ti, Si) ← O_select;
    for j := 1 to s do
        READ(Ti);
        WRITE(Ti);
        EXECUTE(Ti);
    end
end
```
Algorithm 1: Learning

```
Pi ← PICKPROPERTY;
OUPUT b;
```
Algorithm 2: Challenge

In the second phase, a challenge phase (Algorithm 2), $\mathcal{ADV}_1$ picks a property P_i by calling a function PICKPROPERTY. Given the results of the different readings and T_c, $\mathcal{ADV}_1$ outputs a bit b, such that $b = 1$ if he guesses that T_c satisfies P_i, and $b = 0$ otherwise.

$\mathcal{ADV}_1$ *succeeds*, if his guess is right.

Definition 1. *PPS is said to be privacy-preserving with respect to $\mathcal{ADV}_1$, if for all adversaries of category $\mathcal{ADV}_1$, $\Pr[\mathcal{ADV}_1 \text{ succeeds}] \leq \frac{1}{2} + \epsilon$, such that ϵ is negligible.*

2) Privacy against $\mathcal{ADV}_2$: As assumed above, $\mathcal{ADV}_2$, i.e., a malicious back-end server, only receives aggregates from readers. In any case, there is no relation between tags, and therewith tag holders, and $\mathcal{ADV}_2$. In conclusion, $\mathcal{ADV}_2$ simply cannot learn anything about properties of tags.

While we do not target a formal proof, privacy against $\mathcal{ADV}_2$ is furthermore discussed and additional reasoning is given in the according security analysis section IV-A2.

3) Unlinkability against $\mathcal{ADV}_1$: The tags targeted in this paper only feature storage capabilities. Hence, tags cannot update the content of their memory themselves after a read and, therefore, the content of a tag's memory does not change between two protocol executions. In the face of an overwhelmingly powerful adversary who can eavesdrop all communications between tags and readers, tags would be trivially linkable. However, we conjecture that it is fair to assume that an adversary in the real world cannot continuously monitor tags and that there is at least one protocol execution that is "un-observed" by the adversary. Once a tag T is re-written outside the range of the adversary, the adversary should not be able to link the previous interactions he has seen to tag T. In accordance with notions of related work such as: *insubvertible encryption* by Ateniese et al. [1], *backward security* by Dimitrou [8], and *privacy against anonymizers* by Sadeghi et al. [19], we assume that there is at least one protocol execution that takes place outside the range of the adversary. Under this assumption, neither external adversaries nor readers are be able to link two responses from the same tag once it is re-written outside their range.

More formally, in a learning phase (Algorithm 3), $\mathcal{ADV}_1$ is provided with r random tags from $\mathcal{O}_{\text{pick}}$. $\mathcal{ADV}_1$ can *read* from and *write* into the r tags for a maximum of s times. After each

for $i := 1$ **to** r **do**
  $T_i \leftarrow \mathcal{O}_{\text{pick}}$;
  for $j := 1$ **to** s **do**
    READ(T_i);
    WRITE(T_i);
    EXECUTE(T_i);
  end
end

Algorithm 3: Learning

$T_0 \leftarrow \mathcal{O}_{\text{pick}}$;
$T_1 \leftarrow \mathcal{O}_{\text{pick}}$;
for $i := 0$ **to** 1 **do**
  for $j := 1$ **to** t **do**
    READ(T_i);
    WRITE(T_i);
    EXECUTE(T_i);
  end
end
$(T_0, T_1) \rightarrow \mathcal{O}_{\text{flip}}$;
$T_b \leftarrow \mathcal{O}_{\text{flip}}$;
OUTPUTb;

Algorithm 4: Challenge

read or write access, the tags interact with legitimate readers by a PPS execution ("EXECUTE").

In the challenge phase (Algorithm 4), $\mathcal{ADV}_1$ is provided with two challenge tags T_0, T_1 that he is allowed to *write* into and *read* from for a maximum of t times. After each access to T_0 and T_1 by $\mathcal{ADV}_1$, T_0 and T_1 interacts with a legitimate reader (EXECUTE).

Then, $\mathcal{O}_{\text{flip}}$ is queried with T_0 and T_1, $\mathcal{O}_{\text{flip}}$ provides $\mathcal{ADV}_1$ with T_b. Given the results of the readings and T_b, the adversary $\mathcal{ADV}_1$ guesses the value of $b \in \{0, 1\}$. He succeeds, if his guess is right.

Definition 2. *PPS is said to provide unlinkability with respect to $\mathcal{ADV}_1$, if for all adversaries of category $\mathcal{ADV}_1$, $\Pr[\mathcal{ADV}_1 \text{ succeeds}] \leq \frac{1}{2} + \epsilon$, such that ϵ is negligible.*

4) Unlinkability against $\mathcal{ADV}_2$: An adversary $\mathcal{ADV}_2$ should not be able to link aggregates to aggregates it has received before. More precisely, a malicious back-end server should not tell, whether a received aggregate involves a tag that was involved in another aggregate received earlier. $\mathcal{ADV}_2$ has access to the system in two phases. During learning (Algorithm 5), $\mathcal{O}_{\text{aggregate}}$ provides $\mathcal{ADV}_2$ with s aggregates $\text{Agg}_1, \ldots, \text{Agg}_s$ that he could decrypt.

In the challenge phase (Algorithm 6), $\mathcal{ADV}_2$ outputs a pair $b, b' \in \{1, \ldots, s\}$ and therewith Agg_b and $\text{Agg}_{b'}$.

for $i := 1$ **to** s **do**
  $Agg_i \leftarrow \mathcal{O}_{\text{aggregate}}$;
end

Algorithm 5: Learning

OUTPUT (b, b')

Algorithm 6: Challenge

$\mathcal{ADV}_2$ succeeds, if Agg_b and $\text{Agg}_{b'}$ have been computed by $\mathcal{O}_{\text{aggregate}}$ with at least one tag in both aggregates.

Definition 3. *PPS is said to provide unlinkability with respect to $\mathcal{ADV}_2$, if for all adversaries of category $\mathcal{ADV}_2$, $\Pr[\mathcal{ADV}_2 \text{ succeeds}] \leq \frac{1}{s(s-1)} + \epsilon$, such that ϵ is negligible.*

III. PPS

To encrypt the properties in PPS, we use Elgamal [9]. Elgamal is a multiplicatively homomorphic encryption and therefore allows ciphertexts aggregation at the readers. Being probabilistic, Elgamal allows readers to re-encrypt the data sent from the tags and hence counters linking attacks. However, the target scenario of our application calls for an additive homomorphism, and thus Elgamal alone falls short of suiting the target application. Therefore, we use Gödel encoding [10] to encode the tag properties into one message and to adapt Elgamal to our application[1].

A. Elgamal Cryptosystem

1) Setup: the system outputs two large prime P and Q. Let $\mathcal{G}$ be a subgroup of $\mathbb{Z}_{\text{P}}^*$ of order Q, and g be a generator of $\mathcal{G}$. All arithmetic operations will be performed mod P.

2) Key generation: the secret key sk is $x \in \mathbb{Z}_{\text{Q}}$. The public key pk is $y = g^x$.

3) Encryption: to encrypt a message $m \in \mathbb{Z}_{\text{P}}^*$, select $r \in \mathbb{Z}_{\text{Q}}$ and compute $c = (u, v) = (g^r, y^r m)$.

4) Decryption: to decrypt $c = (u, v)$, compute $m = \frac{v}{u^x}$.

To adapt Elgamal to our scheme, we encode the properties using Gödel encoding before encryption as follows.

B. Gödel Property Encoding

To efficiently encode the tag holder properties, we assign to each property P_i a prime number p_i. Both, properties P_i and primes p_i are publicly known.

• **Setup:** let P_i, $1 \leq i \leq p$, be the p properties the back-end server is interested in, and p_i are p different primes. Each property P_i is mapped to one p_i.

• **Encoding:** let m be the vector $(\nu_1, ..., \nu_p)$ such that $\nu_i = 1$, if the tag T fulfills the property P_i, otherwise $\nu_i = 0$. The encoding of the properties of the tag T is defined as $\Omega(m) = \prod_{i=1}^{p} p_i^{\nu_i}$. Note that this encoding is homomorphic: $\forall m_1, m_2 \in \{0,1\}^p$, $\Omega(m_1 + m_2) = \Omega(m_1)\Omega(m_2)$.

• **Decoding:** factorization of $\Omega(m)$ yields the p different factors $p_i^{\nu_i}$ and therewith properties P_i.

C. Protocol

Overview: In PPS, the tags are initialized once by the issuer. Whenever a tag T is read by a reader R, the reader aggregates the ciphertext $c = (u, v)$ it receives from T, then it re-encrypts the ciphertext c and writes the new ciphertext into T. Periodically, readers in the system forward their aggregates to the back-end server. The latter decrypts and decodes the aggregates and computes the statistics it is interested in.

We assume that the system comprises, for ease of understanding, a single reader, and it has γ tags in its range.

• **System setup:** the output of the setup operation is a pair of keys (pk, sk): $(y = g^x, x)$, $x \in \mathbb{Z}_{\text{Q}}$, and p primes p_i such that the property P_i corresponds to prime number p_i. Elgamal secret key $sk = x$ is known by both the issuer and the back-end server. Generator g, the public key $pk = y$ and the p primes are made public.

• **Tag initialization:** the input comprises vector $m = (\nu_1, ..., \nu_p)$, public key y, p primes p_i, and random number $r \in \mathbb{Z}_{\text{Q}}$. Issuer I encodes the vector m following the Gödel encoding and computes $\omega = \Omega(m)$. The output of the initialization operation is a ciphertext $(u, v) = (g^r, y^r \omega)$.

• **Aggregation:** provided with a set of γ ciphertexts (u_i, v_i), $1 \leq i \leq \gamma$ received from tags in its range. The reader outputs the *aggregate* $(U, V) = (\prod_{i=1}^{\gamma} u_i, \prod_{i=1}^{\gamma} v_i)$.

[1]Note that additively homomorphic encryptions such as Paillier [17] or Naccache-Stern [16] may appear to be suitable. However, these schemes do not support an efficient and compact encoding of *multiple* tag properties, rendering them impractical.

TABLE I
SAMPLE PROPERTIES AND THEIR ENCODING

Properties	Gödel encoding
Male	2
under 25	3
Student	5
Employee	7
European union citizen	11
Disabled	13
Aggregate size γ	68

• **Re-encryption:** upon receiving a ciphertext $(u, v) = (g^r, y^r\omega)$ from a tag T, the reader picks a random number $r' \in \mathbb{Z}_Q$ and computes a new ciphertext $(u', v') = (g^{(r+r')}, y^{(r+r')}\omega)$. Note that a value $y^{(r+r')}\omega = 0 \bmod P$ is considered "forbidden". When a reader reads a tag that stores 0, it discards the tag. This means that the reader does not aggregate or re-encrypt the tag.

• **Decryption and decoding:** upon receiving the ciphertext $(U, V) = (\prod_{i=1}^{\gamma} u_i, \prod_{i=1}^{\gamma} v_i)$ from the reader. The back-end server computes $W = \frac{V}{U^x}$ and factorizes $W = \prod_{i=1}^{p} p_i^{\nu_i}$. This factorization is easily feasible, as the back-end server knows the primes p_i. Given this factorization, the back-end server gets $\Omega^{-1}(W) = (\nu_1, ..., \nu_p)$. The respective ν_i corresponds to the number of tags satisfying the property P_i that have been read by the reader.

To get the total number of tags satisfying a property P_i in the case of multiple readers, the back-end server sums the ν_i for all the readers in the system.

Aggregation under restrictions: To ensure the correctness of statistics obtained by the back-end server, we cannot allow the readers to aggregate an infinite number of ciphertexts. They can only aggregate up to a threshold γ of ciphertexts $c_i = E(\omega_i)$ at a time, such that $\prod_{i=1}^{\gamma} \omega_i < P$.

Evaluation: Given p properties P_i and p primes p_i, the threshold γ is defined as $\frac{|P|}{\log_2(\prod_{i=1}^{p} p_i)}$, typically $|P| = 1024$ bits. Furthermore, if readers send to the back-end server the number of tags they read, we can reduce the number of primes used in the Gödel encoding to represent the different properties. For example, this applies in the case with complementary properties, such as $(P_1, P_2) = (male, female)$. A sample Gödel encoding of a card holder's private properties for an imaginary loyalty card is presented in Table I. Given the total number of tags read and the number of tag holders satisfying P_1, we deduce the number of tag holders satisfying P_2. This leads to a more efficient property encoding and thus a larger aggregate size γ which improves the privacy of PPS against $\mathcal{ADV}_2$ as discussed in Section IV-B2.

IV. PRIVACY ANALYSIS

This section provides *formal proofs* for PPS's privacy and unlinkability as defined in the models of Section II-B.

In this section, we use two additional oracles:

$\mathcal{O}_{\text{semantic}}$ is provided with two plaintexts ω_0, ω_1, randomly chooses $b \in \{0, 1\}$, encrypts ω_b using Elgamal and public key pk, and returns the resulting ciphertext c_b.

$\mathcal{O}_{\text{semantic}-\text{re}}$ is provided with two Elgamal ciphertexts c_0, c_1, randomly chooses $b \in \{0, 1\}$, re-encrypts c_b using public key pk, and returns the resulting ciphertext c'_b.

A. Privacy

1) Privacy against $\mathcal{ADV}_1$:

Theorem 1. *PPS is privacy-preserving with respect to $\mathcal{ADV}_1$ under the DDH assumption over $\mathcal{G}$.*

Proof: Assume we have an adversary $\mathcal{A} \in \mathcal{ADV}_1$ who breaks the privacy experiment. We build an adversary $\mathcal{A}'$ that executes $\mathcal{A}$ as a subroutine and breaks the semantic security of Elgamal which leads to a contradiction under DDH. In this proof, we make use of the fact that a tag T satisfies a property P_i, **iff** the corresponding prime number p_i divides the plaintext underlying the ciphertext stored on T.

– $\mathcal{A}'$ picks p properties P_i that he maps to p distinct primes p_i. Then, $\mathcal{A}'$ computes n Gödel encodings ω_j using the primes p_i. Finally, he encrypts ω_j using Elgamal and gets n ciphertexts that he stores on the tags.

– $\mathcal{A}'$ specifies two plaintexts $\omega_0 = \prod p_i^{\nu_{0,i}} \leq P - 1$ and $\omega_1 = \prod p_i^{\nu_{1,i}} \leq P - 1$, such that $\forall i$, $1 \leq i \leq p$, and $b' \in \{0, 1\}$: $\nu_{b',i} \in \{0, 1\}$ and $\nu_{0,i} + \nu_{1,i} = 1$. In terms of properties P_i, this means that tag T_0, storing plaintext ω_0, and tag T_1, storing ω_1, do not have a property in common.

The adversary $\mathcal{A}'$ should specify ω_0 and ω_1 such that $\nu_{0,i} + \nu_{1,i} = 1$. Otherwise, $\mathcal{A}$ could choose a challenge property P_i that both ω_0 and ω_1 encode. In this case, the output of $\mathcal{A}$ about P_i will not provide the necessary information to $\mathcal{A}'$ to break the semantic security of Elgamal. The same holds if $\mathcal{A}$ chooses a property P_i that neither ω_0 nor ω_1 encode.

– $\mathcal{A}'$ transmits $\{\omega_0, \omega_1\}$ to the oracle $\mathcal{O}_{\text{semantic}}$.

– $\mathcal{O}_{\text{semantic}}$ returns the encryption c_b of one of the plaintexts ω_0, ω_1 to $\mathcal{A}'$.

– $\mathcal{A}'$ writes c_b into a challenge tag T_c. Then, $\mathcal{A}'$ calls the adversary $\mathcal{A}$ that enters the learning phase. Simulating $\mathcal{O}_{\text{select}}$, $\mathcal{A}'$ provides $\mathcal{A}$ with $r - 1$ tags along with the list of properties they are satisfying. $\mathcal{A}$ is allowed to read and write into these tags for a maximum of s times. $\mathcal{A}'$ provides $\mathcal{A}$ as well with the challenge tag T_c. $\mathcal{A}$ has only read access to T_c and he is allowed to read it for a maximum of t times. Tags are required to interact with a legitimate reader through the function EXECUTE after being read or written into. As pk is public, $\mathcal{A}'$ can simulate successfully EXECUTE.

– $\mathcal{A}$ selects a property P_i and outputs 1, if T_c satisfies P_i and 0 otherwise.

If $\mathcal{A}$ outputs 1, this implies that the prime number p_i corresponding to P_i divides ω_b. By construction, ω_0 and ω_1 do not have any prime divisor in common, and therefore, ω_b is the plaintext dividable by p_i.

If $\mathcal{A}$ outputs 0, this implies that p_i does not divide ω_b and by construction p_i divides ω_{1-b}. Therefore, ω_b is the plaintext that is not dividable by p_i.

$\mathcal{A}'$ can tell which plaintext ω_b corresponds to c_b. This breaks the semantic security of Elgamal ensured under the DDH assumption [20], which leads to a contradiction. ∎

2) Privacy against $\mathcal{ADV}_2$: As stated in Section II-A, $\mathcal{ADV}_2$ receives only aggregated ciphertexts. Still, given the aggregates, $\mathcal{ADV}_2$ can learn some information about the properties of tags read by readers, but is never able to tell *which* tag, and therewith *which* holder satisfies *which* property.

For instance, if $\mathcal{ADV}_2$ receives an encrypted aggregate from a reader R, and decrypts it to $\text{Agg} = \prod_{i=1}^{p} p_i^{\nu_i}$, and $\exists j$ such that $\nu_j = 0$ after factorization, $\mathcal{ADV}_2$ can learn that all the tags that were read by R do not satisfy the property P_j.

However, as $\mathcal{ADV}_1$ and $\mathcal{ADV}_2$ do not collude, $\mathcal{ADV}_2$ cannot tell *which* tag satisfies or does not satisfy a certain property P_i.

B. Unlinkability

1) Unlinkability against $\mathcal{ADV}_1$:

Theorem 2. *PPS provides tag unlinkability against $\mathcal{ADV}_1$ under the DDH assumption over $\mathcal{G}$.*

Proof: The semantic security property of Elgamal encryption can be extended to the semantic security of Elgamal *under re-encryption* [11]. Let $\mathcal{A}'$ be an adversary that chooses two ciphertexts c_0 and c_1, $\mathcal{A}'$ then sends $\{c_0, c_1\}$ to $\mathcal{O}_{\text{semantic-re}}$. $\mathcal{O}_{\text{semantic-re}}$ flips a coin b, re-encrypts c_b to c'_b and returns c'_b to $\mathcal{A}'$. The semantic security of Elgamal under re-encryption entails that guessing the value of b is as difficult as DDH, see Golle et al. [11].

Now, assume we have an adversary $\mathcal{A} \in \mathcal{ADV}_1$ whose advantage to break the unlinkability experiment is not negligible. We construct a new adversary $\mathcal{A}'$ that executes $\mathcal{A}$ and breaks Elgamal's semantic security under re-encryption.

– $\mathcal{A}'$ picks p properties $p_i, 1 \leq i \leq p$ that he maps to p distinct primes $p_i, 1 \leq i \leq p$. Then, he initializes n tags.

– $\mathcal{A}'$ calls the adversary $\mathcal{A}$ that enters the learning phase. $\mathcal{A}'$ simulates $\mathcal{O}_{\text{pick}}$ and provides $\mathcal{A}$ with r tags. $\mathcal{A}$ is allowed to read and write into these tags for a maximum of s times. After each reading, $\mathcal{A}'$ simulates EXECUTE and re-encrypts the ciphertexts, as pk is public.

– $\mathcal{A}$ enters the challenge phase: $\mathcal{A}'$ simulates $\mathcal{O}_{\text{pick}}$ and submits tags T_0 and T_1 to the adversary $\mathcal{A}$. $\mathcal{A}$ writes into and reads from T_0 and T_1 for a maximum of t times. $\mathcal{A}'$ can simulate successfully the function EXECUTE as pk is public.

– $\mathcal{A}'$ reads the data stored on T_0 and T_1. Without loss of generality, let c_0 (c_1 resp.) denotes the ciphertext stored on T_0 (T_1 resp.). Then, $\mathcal{A}'$ transmits c_0 and c_1 to the oracle $\mathcal{O}_{\text{semantic-re}}$.

– $\mathcal{O}_{\text{semantic-re}}$ returns the result c'_b of re-encrypting one of the two ciphertexts to $\mathcal{A}'$. $\mathcal{A}'$ writes c'_b into a tag T.

– $\mathcal{A}'$ calls $\mathcal{A}$ and provides him with T, simulating $\mathcal{O}_{\text{flip}}$. Then, $\mathcal{A}$ outputs his guess for the value of b.

Since $\mathcal{A}$'s advantage in the unlinkability experiment is not negligible, $\mathcal{A}$ can tell which tag corresponds to the new ciphertext c'_b. If $\mathcal{A}$ outputs 0, this means that c'_b is re-encryption of c_0, otherwise c'_b is a re-encryption of c_1. Therefore, $\mathcal{A}'$ can break the semantic security under re-encryption of Elgamal that is ensured under the DDH assumption [11], again leading to a contradiction. ■

2) Unlinkability against $\mathcal{ADV}_2$:

Theorem 3. *PPS provides unlinkability of tags against $\mathcal{ADV}_2$ for large γ.*

Sketch: An aggregate $\text{Agg} = \prod_{i=1}^{p} p_i^{\nu_i}$ is called *completely blinded*, **iff** $\forall i,\ 1 \leq i \leq p : \nu_i > 0$. Now, given a sufficiently large γ, the aggregates received by the back-end server will be completely blinded with high probability.

Therefore, the back-end server cannot distinguish between the tags involved in the aggregates. Moreover, using a large s in the learning phase would not give the adversary $\mathcal{ADV}_2$ a greater advantage in guessing (b, b').

In the following, we compute an upper bound of the advantage ϵ of $\mathcal{ADV}_2$ in the unlinkability experiment.
Let E be the event that aggregate Agg is completely blinded, so $\forall i,\ 1 \leq i \leq p : \nu_i > 0$. Let γ be the number of ciphertexts participating in the aggregate, and π_i is the probability that a tag holder satisfies property P_i. Without loss of generality, we assume $\pi_1 \leq \pi_2 \leq \ldots \leq \pi_p$. Then, the probability that $\nu_i = 0$ is $Pr(\nu_i = 0) = (1 - \pi_i)^\gamma \leq (1 - \pi_1)^\gamma$.

Let $\overline{E}$ be the complementary event of E. Therefore,
$Pr(\overline{E}) = Pr(\nu_1 = 0 \vee \nu_2 = 0 \ldots \vee \nu_p = 0)$
$Pr(\overline{E}) \leq \sum_{i=1}^{p} Pr(\nu_i = 0) \leq p(1 - \pi_1)^\gamma$.

$\epsilon = Pr(\overline{E})$ is the advantage of $\mathcal{ADV}_2$ in the unlinkability experiment which is **negligible** in γ. Therefore, we say that PPS is ϵ-unlinkable against $\mathcal{ADV}_2$, such that $\epsilon \leq p(1 - \pi_1)^\gamma$.

Note that the advantage of $\mathcal{ADV}_2$ heavily depends on the probability π_1. If π_1 is very small, i.e., representing a *rare* property such as being disabled, PPS cannot provide unlinkability against $\mathcal{ADV}_2$. In such a case, the back-end server can link tags to aggregates. For instance, if the back-end server sees two aggregates where the property "disabled" is satisfied, it can guess with a non negligible probability that these two aggregates have one tag in common. ■

V. Security Analysis

Tags in our scheme only feature (re-)writable memory. As there is no access control on tags to check the authenticity of readers re-writing their memory, such a setup is vulnerable to "malicious writing". Malicious writing affects the correctness of the results obtained at the back-end server. Given that access control is not feasible in our read-write only tags, this attack *cannot* be prevented. We can divide malicious writing attacks into two categories:

- **Writing an invalid ciphertext ("garbage") into the tag:** this attack can be detected at the back-end server, as decryption and Gödel decoding will not succeed. Moreover, if the adversary writes the value 0 into the tag, this will be detected at the next honest reader to read the tag.
- **Writing a valid ciphertext into the tag:** a malicious reader could try to alter statistics. The simplest way to implement such an attack is by copying the content of a tag into another one ("cloning"). Since the ciphertext written into the tag is a valid one, this type of attack cannot directly be detected at decryption, and we will tackle it in the following.

Instead of one ciphertext, each tag stores two ciphertexts (c, c_{ID}). The first ciphertext c encrypts the properties of the tag holder as described in the previous section. The second ciphertext c_{ID} encrypts a unique ID of the tag using standard Elgamal encryption. After a tag is scanned by a reader, the reader re-encrypts both ciphertexts c and c_{ID} and writes the new ciphertexts into the tag. The reader aggregates c and keeps a record of c_{ID}. During decryption at the back-end server, if the back-end server suspects that a received aggregate is not correct, he contacts a "trusted third party". This trusted third party (TTP) checks the records c_{ID} stored at the readers. TTP decrypts these ciphertexts and gets the IDs of the tags that were scanned along with the corresponding properties of their holders. In this manner, the TTP detects tag cloning as the ID of the cloned tag will be repeated several times.

Furthermore, in order to detect tag tampering, the tag issuer should keep a database of the tag IDs and their corresponding properties and reveal it to the TTP. Therewith, the TTP can compare the decrypted properties and the actual properties stored in the issuer database. If there is a discrepancy between the properties corresponding to the same tag ID, the TTP reports a fraud. Meanwhile, the TTP does not reveal the records of the IDs stored on the readers either to the back-end server or to the readers.

VI. Related work

Juels et al. [14] utilize re-encryption to protect privacy of RFID-enabled banknotes. Each time a banknote is spent, the readers in shops or banks re-encrypt the encrypted serial number of the banknote stored on the tag. The main drawback of this scheme is that the authorized readers have access to the plaintext underlying the ciphertext spoiling unlinkability. Similarly, Golle et al. [11] introduce *universal re-encryption* allowing special re-encryption without knowing the public key initially used to encrypt the plaintexts. While this protocol provides key privacy, it fails at providing unlinkability after *malicious writing*. An adversary can write his own message m into a tag and encrypt it under its public key. Therewith, the adversary can always link the tag.

Ateniese et al. [1] tackle the above problems by proposing *insubvertible encryption*, i.e., universal re-encryption and randomized certificates. If the certificate is valid, the ciphertext stored on the tag will be re-encrypted. Otherwise, it will be replaced by a *dummy encryption*. Ateniese et al. [1] aim at privacy preserving *identification*, but not privacy preserving statistics collection which is the focus of PPS. Also, Ateniese et al. [1], as well as the results presented by Blundo et al. [3], require special message encodings to map messages to points on elliptic curves. However, currently known efficient encoding schemes fail at preserving the homomorphic properties that are the essential prerequisite for PPS.

Camenisch and Groß [6] propose an attribute encoding for anonymous credentials. The scheme allows users to prove the possession of an attribute with a given value while preserving the privacy of the users. While such an approach could be used to "emulate" privacy-preserving computation of statistics, the main drawback is the requirement for complex *interactive* proofs between tags and readers– infeasible in our setting with storage-only tags.

Han et al. [12] present a protocol to estimate the total number of tags in the vicinity of a reader. The main idea is to infer this number by examining the number of empty and collision slots in the framed slotted Aloha protocol used for communication. Although [12] enables estimating the total number of tags anonymously, it does not lend itself to collect statistics on tag properties as targeted in the paper at hand.

Kerschbaum et al. [15] propose to privately compute performance properties of an RFID supply chain using data stored on tags. However, this work focuses on computing these metrics without leaking sensitive information of the supply-chain's parties. Kerschbaum et al. [15] use additive homomorphic encryption that does not support collecting statistics on multiple properties and consequently cannot be as efficient as PPS.

References

[1] G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable rfid tags via insubvertible encryption. In *12th ACM conference on Computer and communications security*, pages 92–101, New York, USA, 2005.

[2] E.-O. Blass, A. Kurmus, R. Molva, G. Noubir, and A. Shikfa. The F_f-family of protocols for rfid-privacy and authentication. *IEEE Transactions on Dependable and Secure Computing*, 8(3):466–480, 2011.

[3] C. Blundo, A. De Caro, and G. Persiano. Untraceable tags based on mild assumptions. Cryptology ePrint Archive, Report 2009/380, 2009. http://eprint.iacr.org/.

[4] J. Bringer and H. Chabanne. Trusted-HB: A low-cost version of HB^{+} secure against man-in-the-middle attacks. *IEEE Transactions on Information Theory*, 54(9):4339–4342, 2008.

[5] J. Bringer, H. Chabanne, and E. Dottax. HB^{++}: a lightweight authentication protocol secure against some attacks. In *SecPerU*, pages 28–33, 2006.

[6] J. Camenisch and T. Groß. Efficient attributes for anonymous credentials. In *15th ACM conference on Computer and communications security*, pages 345–356, New York, USA, 2008.

[7] C. Chatmon, T. van Le, and M. Burmester. Secure anonymous RFID authentication protcols. Technical report, Florida State University, Department of Computer Science, Tallahassee, Florida, USA, 2006. http://www.cs.fsu.edu/~burmeste/TR-060112.pdf.

[8] T. Dimitrou. rfidDOT: RFID delegation and ownership transfer made simple. In *International Conference on Security and privacy in Communication Networks*, Istanbul, Turkey, 2008.

[9] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, USA, 1985. Springer New York, Inc.

[10] K. Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatsheft für Mathematik und Physik*, 38: 173–198, 1931.

[11] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In *RSA Conference, Cryptographer's track*, pages 163–178. Springer, 2004.

[12] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu. Counting RFID tags efficiently and anonymously. In *IEEE INFOCOM*, San Diego, USA, 2010.

[13] A. Juels and S. A. Weis. Defining strong privacy for RFID. In *5th IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 342–347, Washington D.C., USA, 2007.

[14] A. Juels, R. Pappu, and Thingmagic Llc. Squealing euros: Privacy protection in rfid-enabled banknotes. In *Financial Cryptography'03*, pages 103–121. Springer, 2002.

[15] F. Kerschbaum, N. Oertel, and L. Weiss Ferreira Chaves. Privacy-preserving computation of benchmarks on item-level data using rfid. In *3rd ACM conference on Wireless network security*, pages 105–110, New York, USA, 2010.

[16] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *ACM conference on Computer and communications security*, pages 59–66, New York, 1998.

[17] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT'99*, pages 223–238. Springer, 1999.

[18] R. Di Pietro and R. Molva. Information confinement, privacy, and security in RFID systems. In *Lecture Notes in Computer Science, Volume 4734*, pages 187–202, 2007.

[19] A. R. Sadeghi, I. Visconti, and C. Wachsmann. Anonymizer-Enabled Security and Privacy for RFID. In *8th International Conference on Cryptology And Network Security*, Kanazawa, Japan, December 2009.

[20] Yiannis T. and Moti Y. On the security of elgamal based encryption. In *PKC'98, LNCS 1431*, pages 117–134. Springer, 1998.

[21] G. Tsudik. YA-TRAP: yet another trivial RFID authentication protocol. In *International Conference on Pervasive Computing and Communications Workshops*, Pisa, Italy, 2006.

[22] S. Vaudenay. On privacy models for RFID. In *Advances in Cryptology*, pages 68–87, 2007.

[23] S.A. Weis, S.E. Sarma, R.L. Rivest, and D.W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *Security in Pervasive Computing*, pages 201–212, Boppard, Germany, 2003.