Untraceable Tags based on Mild Assumptions[*]

Carlo Blundo, Angelo De Caro, and Giuseppe Persiano
Dipartimento di Informatica ed Applicazioni, Università di Salerno, Italy

Monday 3rd August, 2009

Abstract

Radio frequency identification (RFID) chips have been widely deployed in large-scale systems such as inventory control and supply chain management. While RFID technology has much advantage, however it may create new problems to privacy. Tag untraceability is a significant concern that needs to be addressed in deploying RFID-based system.

In this paper we propose a new construction for untraceable tags. Our construction is the first construction in the symmetric bilinear setting based on a *mild* assumption. That is our assumption is tautological in the generic group model and is "efficiently falsifiable" in the sense that its problem instances are stated non-interactively and concisely (i.e., independently of the number of adversarial queries and other large quantities).

1 Introduction

RFID [4] and NFC [9] are the *de-facto* technology for storing small amount of data on devices that can be read without physical contact. It is expected that everyday objects will be tagged with small components which are used to carry information to identify the object. For example, the garnment industry plans to use RFID tags for the management of post-sale services. Obviously, it is expected that encryption is used for storing information on the tag so that only legitimate users can access the stored data. Encryption though does not solve all problems and we are interested in privacy issues associated with RFID tags. Specifically, RFID tags can be read by anyone

[*]The work of the authors has been supported in part by the European Commission through the FP7 Information Communication Technologies programme, under Contract FET-215270 FRONTS (Foundations of Adaptive Networked Societies of Tiny Artefacts).

and the string stored on a tag, even though it is a ciphertext, can be used to trace the tag and, in the case the tag is attached to a personal object, to trace the owner of the tag.

We thus envision a system in which the environment helps in alleviating this problem: as tags move in the environment they are read by special devices called the *randomizers* which provide the following service: everytime a randomizer reads a tag carrying a ciphertext, the ciphertext is re-randomized; that is, a new ciphertext carrying the same cleartext is computed. This can be easily achieved if the randomizers are trusted with the secret keys: just decrypt the ciphertext to obtain the cleartext and then encrypt the cleartext using fresh randomness. In some applications though this is a very strong trust assumption: even if one of the randomizers is corrupted then all privacy is lost. We thus look at the problem of designing special encryption schemes that support re-randomization; that is, given a ciphertext Ct carrying cleartext M, it is possible to produce a new ciphertext Ct' carrying the same cleartext M, even if the decryption key is not available.

The El-Gamal encryption scheme. A simple variation of the El-Gamal encryption scheme is known to be re-randomizable [5], but it is of limited applicability. Let us review the re-randomizable version of the ElGamal encryption scheme.

Let p be a large prime and let g be a generator of $\mathbb{Z}_p^\star$. The public key for the ElGamal encryption scheme consists simply of an element $y \in \mathbb{Z}_p^\star$ and the associated secret key is $x \in \mathbb{Z}_p^\star$ such that $y = g^x$ (all operations are in $\mathbb{Z}_p^\star$). In the encryption scheme $\mathsf{rElGamal}$ (the re-randomizable version of the ElGamal encryption scheme), to encrypt message $M \in \mathbb{Z}_p^\star$, one selects $r, s \in Z_p^\star$ at random and computes the pair (g^r, My^r, g^s, y^s). The plaintext associated to ciphertext $\mathsf{Ct} = (C_0, C_1, U_0, U_1)$ is recovered by computing C_1/C_0^x, where x is the secret key. The re-randomization procedure takes a ciphertext $\mathsf{Ct} = (C_0, C_1, U_0, U_1)$, selects $t, t' \in Z_p^\star$ at random and returns $\widehat{\mathsf{Ct}} = (C_0 \cdot U_0^t, C_1 \cdot U_1^t, U_0^{t'}, U_1^{t'})$. It is easy to see that if Ct is a ciphertext for cleartext M then $\widehat{\mathsf{Ct}}$ is a uniformly distributed ciphertext for the same cleartext M. Also notice that the re-randomization procedure does not need to know neither the public key nor the secret key associated with the ciphertext Ct.

Suppose now that we want to store message M on a tag and suppose we use $\mathsf{rElGamal}$ to encrypt M before actually storing on the tag. Unfortunately, an adversary $\mathcal{A}$ that wants to trace a tag has a very simple and successful strategy. $\mathcal{A}$ simply generates a pair of public/secret key $(y_{\mathcal{A}}, x_{\mathcal{A}})$ for $\mathsf{rElGamal}$ and writes a random message $M_{\mathcal{A}}$ on the tag $T_{\mathcal{A}}$ that he wants to trace by computing ciphertext $\mathsf{Ct}_{\mathcal{A}}$ for public key $y_{\mathcal{A}}$. Notice that everytime $T_{\mathcal{A}}$ is re-randomized by the randomizers, message $M_{\mathcal{A}}$ is not affected. Thus to check that a given tag T is actually $T_{\mathcal{A}}$, $\mathcal{A}$ can simply try to decrypt the stored ciphertext and if the decryption gives back $M_{\mathcal{A}}$ then with very high probability $\mathcal{A}$ can conclude that he is in presence of $T_{\mathcal{A}}$.

We notice that $\mathsf{rElGamal}$ can still be used in the scenario in which writing on the tag can be selectively disabled by the owner. That is, the owner of the tag enables writing on the tag when in presence of trusted randomizers and disables writing if he is in an untrusted environment.

The scenario. In this paper, we consider the more challenging scenario in which writing on a tag cannot be selectively disabled. Obviously, in this scenario, an adversary $\mathcal{A}$ can destroy the content of a tag T by overwriting its content. We will guarantee though that $\mathcal{A}$ cannot trace tag T even in this case.

We have three types of honest players:

1. The Central Authority CA that publishes some public information Pub and issues a pair of private and secret keys to each authorized player.

2. The players that receive a public and secret key from the CA and use the keys to encrypt and decrypt messages that are stored on tags.

3. The randomizers that receive tags and randomize the ciphertexts stored on the tags. The randomization procedure changes the ciphertext but not the cleartext stored on the tag.

Notice that the role of the CA is necessary: if users could generate keys by themselves then the it would not be possible to prevent attacks similar to the one we have discussed for the $\mathsf{rElGamal}$ encryption scheme.

In this paper we give a construction for untraceable tags. We split the presentation in two parts. In Section 4 we present a tag system that is secure against adversaries that can only read tags. Building on the construction of Section 4, in Section 5 we present out main result, a tag system that is secure against adversaries that can write on tags.

Previous work. In [1], a construction for an untraceable tag system was proposed. The security of the construction of [1] is based on a stronger version of the LRSW assumption introduced by Lysyanskaya et al. [6]. The strong LRSW assumption does not hold for symmetric bilinear mapping. Specifically, the construction of [1] requires the existence of three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ such that no morphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ exists and of a bilinear mapping $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. This is called the *asymmetric* bilinear setting. If one tries to use the construction of [1] in the symmetric bilinear setting then, as it is easily seen, tags become traceable. Our construction instead is in the *symmetric* bilinear setting. In [1] the authors state that in the full version of their paper they will show a construction of the symmetric bilinear setting. To the best of our knowledge such a full version was never published.

Moreover, our construction is based on a *mild* assumption in the sense of [2]. That is our assumption is tautological in the generic group model [11] and is "efficiently falsifiable" [8] in the sense that its problem instances are stated non-interactively and concisely (i.e., independently of the number of adversarial queries and other large quantities). In contrast, the assumption used to prove the security of the construction in [1] is stated in an interactive way.

2 The model

We start by defining the notion of a tag system and then define its security properties. We consider quintuples of algorithms (GenPub, GenKey, rEnc, rDec, $\mathsf{Randomize}$) with the following intended meaning.

1. $\mathsf{GenPub}(1^k)$ is executed by the CA. It takes as input the security parameter k and returns the public information Pub, the randomizing information rPub, and the master secret key Msk.

2. $\mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$ is executed by the CA to generate the secret key of a player. It takes as input the public information Pub and the master secret key Msk and returns the public key Pk and the secret key Sk.

3. $\mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ is executed by a player to encrypt a message M to be written on a tag. It takes as input the public information Pub, the public key Pk of the user for which the message is encrypted, and the message M and returns the ciphertext Ct.

4. $\mathsf{rDec}(\mathsf{Pub}, \mathsf{Sk}, \mathsf{Ct})$ is executed by a player to decrypt a ciphertext Ct. It takes as input the public information Pub, the secret key Sk of the user, and the cipheretext Ct and returns the cleartext M.

5. $\mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Ct})$ is executed by the randomizers to randomize ciphertexts. It takes as input the public information Pub, the randomizing information rPub, and a ciphertext Ct that encrypts a message M for public key Pk and returns a new ciphertext $\mathsf{Ct}^\star$ that encrypts message M for Pk. We stress that M, Pk, and the secret key Sk are not given as input to $\mathsf{Randomize}$.

In a typical scenario, the players are manufacturers that attach tags to consumer goods. They obtain their pair of private and secret key from the CA and use the encryption algorithm rEnc to store information regarding the good on the tag. We envision randomizers being present in the physical environment were the end user lives. Finally, the decryption algorithm rDec is used by the manufacturer to recover the information written on the tag when the end user requires assistance (or maintenance) from the manufacturer.

Definition 2.1 *A* tag system *is a quintuple of algorithms (*GenPub*,* GenKey*,* rEnc*,* rDec*,* $\mathsf{Randomize}$*) such that for any* $\ell = \mathsf{poly}(k)$*,*

$$\begin{aligned}\text{Prob}[&(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k); (\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk});\\ &\mathsf{Ct}_0 \leftarrow \mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M});\\ &\mathsf{Ct}_1 \leftarrow \mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Ct}_0);\\ &\ldots;\\ &\mathsf{Ct}_\ell \leftarrow \mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Ct}_{\ell-1}); \mathsf{M}' \leftarrow \mathsf{rDec}(\mathsf{Pub}, \mathsf{Sk}, \mathsf{Ct}_l) : \mathsf{M} = \mathsf{M}'] = 1\end{aligned}$$

We next define the security properties of a tag system. We start from semantic security.

2.1 Semantic security

Consider the following experiment with an adversary $\mathcal{A}$.

$\mathsf{SSExp}_\mathcal{A}(1^k)$
1. $(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k)$.
2. $(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$.
3. Run $\mathcal{A}$ on input Pub and Pk and obtain messages $\mathsf{M}_0, \mathsf{M}_1$.
4. Toss a random coin $\eta \in \{0, 1\}$ and compute $\mathsf{Ct} = \mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M}_\eta)$.
5. Run $\mathcal{A}$ on input Ct and let η' be its output.
6. If $\eta = \eta'$ then return 1 else return 0.

In $\mathsf{SSExp}_\mathcal{A}$ the adversary $\mathcal{A}$ selects two strings of his choice, M_0 and M_1. Then, one of the strings is picked at random, it is encrypted and given to the adversary. We require that the adversary is not able to guess which of the two string has been encrypted.

Definition 2.2 *A tag system (*GenPub*,* GenKey*,* rEnc*,* rDec*,* $\mathsf{Randomize}$*) is* semantically secure *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *we have that*

$$\left|\text{Prob}[\,\mathsf{SSExp}_{\mathcal{A}}(1^k) = 1\,] - \frac{1}{2}\right|$$

is negligible in k*.*

2.2 Weak untraceability

Next we define the notions of untraceability for a tag systems. We start with the notion of weak untraceability and then present our notion of strong untraceability.

For defining the notion of weak untraceability we use the following experiment.

```
$\mathsf{WUExp}_{\mathcal{A}}(1^k)$
  1. $(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k)$.
  2. $(\mathsf{Pk}_0, \mathsf{Sk}_0) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$ and $(\mathsf{Pk}_1, \mathsf{Sk}_1) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$.
  3. Run $\mathcal{A}$ on input $\mathsf{Pub}, \mathsf{Pk}_0$ and $\mathsf{Pk}_1$ and obtain messages $\mathsf{M}_0, \mathsf{M}_1$.
  4. Compute $\mathsf{Ct}_0 = \mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}_0, \mathsf{M}_0)$, $\mathsf{Ct}_1 = \mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}_1, \mathsf{M}_1)$.
  5. Toss a random coin $\eta \in \{0,1\}$ and compute
         $\mathsf{Ct}^\star = \mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Ct}_\eta)$.
  6. Run $\mathcal{A}$ on input $\mathsf{Ct}_0, \mathsf{Ct}_1, \mathsf{Ct}^\star$ and let $\eta'$ be its output.
  7. If $\eta = \eta'$ then return 1 else return 0.
```

In $\mathsf{WUExp}_{\mathcal{A}}$ the adversary $\mathcal{A}$ selects two strings of his choice, M_0 and M_1. Both strings are encryped using different public keys (namely, Pk_0 and Pk_1) obtaining the ciphertexts Ct_0 and Ct_1, respectively. Then, one of the ciphertexts is picked at random, it is re-randomized and given to the adversary along with Ct_0 and Ct_1. We require that the adversary is not able to guess which of the two ciphertexts (i.e, tags) has been re-randomized.

Definition 2.3 *A tag system (*GenPub*,* GenKey*,* rEnc*,* rDec*,* $\mathsf{Randomize}$*) is* weakly untraceable *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *we have that*

$$\left|\text{Prob}[\,\mathsf{WUExp}_{\mathcal{A}}(1^k) = 1\,] - \frac{1}{2}\right|$$

is negligible in k*.*

We remark that weak untraceability protects against adversaries that can only read tags and not write on tags. Thus it is a very weak notion and cannot be applied to our scenario of interest. In Section 4 we will give a construction of a weakly untraceable tag system which constitutes the basis for our construction of a strongly untraceable tag system.

2.3 Strong untraceability

Next we define the notion of a strongly untraceable tag system and for this we need the following experiment.

```
$\mathsf{SUExp}_{\mathcal{A}}(1^k)$
  1. $(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k)$.
  2. $(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$.
  3. Run $\mathcal{A}$ on input $\mathsf{Pub}$ and $\mathsf{Pk}$ and obtain strings $\mathsf{Ct}_0$ and $\mathsf{Ct}_1$.
  4. Set $\mathsf{Ct}_0^\star \leftarrow \mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Ct}_0)$ and
         $\mathsf{Ct}_1^\star \leftarrow \mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Ct}_1)$.
  5. If $\mathsf{Ct}_0^\star = \perp$ or $\mathsf{Ct}_1^\star = \perp$ then return 0.
  6. Toss a random coin $\eta \in \{0, 1\}$.
  7. Run $\mathcal{A}$ on input $\mathsf{Ct}_\eta^\star$ and let $\eta'$ be its output.
  8. If $\eta = \eta'$ then return 1 else return 0.
```

Essentially in $\mathsf{SUExp}_{\mathcal{A}}$ the adversary $\mathcal{A}$ selects two strings of his choice, Ct_0 and Ct_1. Then both strings are re-randomized and, if the procedure is successful on both of them, then one is picked at random and given to the adversary. We require that the adversary is not able to guess which of the two tags has been re-randomized. Notice that if the adversary selects the two strings so that the randomization procedure fails (that is, it outputs the special failure symbol $\perp$) on exactly one of them, then traceability is unavoidable. We disallow this case by having the experiment return 0 (meaning that the adversary failed).

Observe also that the two strings Ct_0 and Ct_1 need not to be well-formed ciphertexts with respect to Pk but still the randomization procedure could be successful. However that if they both are well-formed ciphertexts then we are actually executing experiment $\mathsf{WUExp}_{\mathcal{A}}$. This implies that strong untraceability is stronger than weak untraceability (as one would expect).

Definition 2.4 *A tag system (*GenPub, GenKey, rEnc, rDec, $\mathsf{Randomize}$*) is* strongly untraceable *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *we have that*

$$\left|\mathrm{Prob}[\,\mathsf{SUExp}_{\mathcal{A}}(1^k) = 1\,] - \frac{1}{2}\right|$$

is negligible in k.

2.4 Strong semantic security

We observe that the notion of semantic security does not make any security guarantee with respect to randomizers. In other words, randomizers are assumed to be trusted. If this is the case, then we have a very simple and direct construction of strongly untraceable tag systems. Roughly speaking, the randomizer decrypts the ciphertext and re-encrypts it using fresh randomness. If instead randomizers cannot be assumed to be trustful, then we require semantic security to hold also with respect to randomizers.

```
$\mathsf{SSSExp}_{\mathcal{A}}(1^k)$
  1. $(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k)$.
  2. $(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$.
  3. Run $\mathcal{A}$ on input $\mathsf{Pub}, \mathsf{Pk}$ and $\mathsf{rPub}$ and obtain messages $\mathsf{M}_0, \mathsf{M}_1$.
  4. Toss a random coin $\eta \in \{0,1\}$ and compute $\mathsf{Ct} = \mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M}_\eta)$.
  5. Run $\mathcal{A}$ on input $\mathsf{Ct}$ and $\mathsf{rPub}$ and let $\eta'$ be its output.
  6. If $\eta = \eta'$ then return 1 else return 0.
```

Experiment $\mathsf{SSSExp}_{\mathcal{A}}$ differs from $\mathsf{SSExp}_{\mathcal{A}}$ in that in the former the adversary is given access to the re-randomizing information rPub and so it correctly models security against randomizers.

Definition 2.5 *A tag system (*GenPub*,* GenKey*,* rEnc*,* rDec*,* $\mathsf{Randomize}$*) is* strongly semantic secure *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *we have that*

$$\left|\mathrm{Prob}[\,\mathsf{SSSExp}_{\mathcal{A}}(1^k) = 1\,] - \frac{1}{2}\right|$$

is negligible in k.

Finally, we have

Definition 2.6 *A quintuple of algorithms (*GenPub*,* GenKey*,* rEnc*,* rDec*,* $\mathsf{Randomize}$*) is an* untraceable tag system *if it is strongly untraceable and strongly semantic secure.*

3 Background on bilinear groups

The symmetric bilinear setting. We have multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p and a non-degenerate pairing function $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. That is, for all $g \in \mathbb{G}$, $\mathbf{e}(g,g) \neq 1$ and $\mathbf{e}(g^a, g^b) = \mathbf{e}(g,g)^{ab}$. We denote by g and $\mathbf{e}(g,g)$ generators of $\mathbb{G}$ and $\mathbb{G}_T$, respectively. We call a *symmetric bilinear* instance a tuple $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathbf{e}]$ and assume that there exists an efficient generation procedure $\mathcal{G}$ that, on input 1^k, outputs an instance with $|p| = \Theta(k)$.

In our constructions we make the following hardness assumptions.

Bilinear Decision Diffie-Hellman Given a tuple $[\mathcal{I}, g^{z_1}, g^{z_2}, g^{z_3}, Z]$ for random exponents $z_1, z_2, z_3 \in \mathbb{Z}_p$ it is hard to distinguish between $Z = \mathbf{e}(g,g)^{z_1 z_2 z_3}$ and a random Z from $\mathbb{G}_T$. More specifically, for an algorithm $\mathcal{A}$ we define experiment $\mathsf{BDDHExp}_{\mathcal{A}}$ as follows.

```
BDDHExp^A(1^k)
01. Choose instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathbf{e}]$ running $\mathcal{G}$ with security parameter $1^k$;
02. Choose $z_1, z_2, z_3 \in \mathbb{Z}_p$ at random;
03. Choose $\eta \in \{0,1\}$ at random;
04. if $\eta = 1$ then choose $z \in \mathbb{Z}_p$ at random
05.           else set $z = z_1 z_2 z_3$;
06. Set $Z_1 = g^{z_1}, Z_2 = g^{z_2}, Z_3 = g^{z_3}$ and $Z = \mathbf{e}(g,g)^z$;
07. Let $\eta' = \mathcal{A}(\mathcal{I}, Z_1, Z_2, Z_3, Z)$;
08. if $\eta = \eta'$ then return 1 else return 0;
```

Assumption 3.1 (Bilinear Decision Diffie-Hellman (BDDH)) *For all probabilistic polynomial-time algorithms $\mathcal{A}$,*

$$\left|\text{Prob}[\,\mathsf{BDDHExp}^{\mathcal{A}}(1^k) = 1\,] - 1/2\right|$$

is negligible in k.

Decision Linear. Given a tuple $[g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, Z]$ for random random exponents $z_1, z_2, z_3, z_4 \in \mathbb{Z}_p$ it is hard to distinguish between $Z = g^{z_3+z_4}$ and a random Z from $\mathbb{G}$. More specifically, for an algorithm $\mathcal{A}$ we define experiment $\mathsf{DLExp}_{\mathcal{A}}$ as follows.

```
DLExp^A(1^k)
01. Choose instance $\mathcal{I} = [p, \mathbb{G}, g, \mathbf{e}]$ running $\mathcal{G}$ with with security parameter $1^k$;
02. Choose $z_1, z_2, z_3, z_4 \in \mathbb{Z}_p$ at random;
03. Choose $\eta \in \{0,1\}$ at random;
04. if $\eta = 1$ then choose $z \in \mathbb{Z}_p$ at random
05.           else set $z = z_3 + z_4$;
06. Set $Z_1 = g^{z_1}, Z_2 = g^{z_2}, Z_{13} = g^{z_1 z_3}, Z_{24} = g^{z_2 z_4}$, and $Z = g^z$;
07. Let $\eta' = \mathcal{A}(\mathcal{I}, Z_1, Z_2, Z_{13}, Z_{24}, Z)$;
08. if $\eta = \eta'$ then return 1 else return 0;
```

Assumption 3.2 (Decision Linear (DL)) *For all probabilistic polynomial-time algorithms $\mathcal{A}$,*

$$\left|\text{Prob}[\,\mathsf{DLExp}^{\mathcal{A}}(1^k) = 1\,] - 1/2\right|$$

is negligible in k.

Note that Symmetric Decision Linear implies Symmetric Decision BDDH and the Symmentric Decision Linear assumption has been used in [2].

4 A first construction

In this section we present our construction of a tag system

$$\mathsf{Tag} = (\mathsf{GenPub}, \mathsf{GenKey}, \mathsf{rEnc}, \mathsf{rDec}, \mathsf{Randomize})$$

and then we show that it is semantically secure and weakly untraceable.

4.1 The construction

Procedure $\mathsf{GenPub}(1^k)$. We now describe the procedure GenPub used by CA to generate the public information Pub, the re-randomizing information rPub and the master secret key Msk.

1. Run $\mathcal{G}(1^k)$ to select a random bilinear instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathbf{e}]$ with $|p| = \Theta(k)$.
2. Pick $t_1, t_2, t_3, \omega, \in \mathbb{Z}_p$ and $g_0, g_1 \in \mathbb{G}$ at random.
3. Set
$$\Omega = \mathbf{e}(g, g)^{\omega t_1 t_2 t_3}, \quad T_1 = g^{t_1}, \quad T_2 = g^{t_2}, \quad T_3 = g^{t_3} \ .$$
4. Set
$$\begin{aligned}&\mathsf{Pub} = [\ \mathcal{I}, \quad g_0, \quad g_1, \quad \Omega, \quad T_1, \quad T_2, \quad T_3\ ],\\ &\mathsf{rPub} = \emptyset, \text{ and}\\ &\mathsf{Msk} = (t_1, t_2, t_3, w).\end{aligned}$$
5. Return $[\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}]$.

Procedure $\mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$. We now describe the procedure used by CA to generate the pair of public and secret key.

1. Pick $r \in \mathbb{Z}_p$ at random.
2. Set $\mathsf{Pk} = g_0 g_1^r$.

3. Set
$$D_0 = g^{rt_1t_2t_3}, \qquad D_1 = g^{-wt_1t_3}\mathsf{Pk}^{-rt_1t_3},$$
$$D_2 = g^{-wt_1t_2}\mathsf{Pk}^{-rt_1t_2}, \quad D_3 = g^{-wt_2t_3}\mathsf{Pk}^{-rt_2t_3} \quad .$$

4. Set
$$\mathsf{Sk} = [D_0, D_1, D_2, D_3].$$

5. Return $[\mathsf{Pk}, \mathsf{Sk}]$.

Procedure $\mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$**.** We first describe the *basic* encryption procedure $\mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ that takes as input the public parameters Pub, the public key Pk, and a cleartext $\mathsf{M} \in \mathbb{G}_T$. Then, we describe the *randomizable* encryption procedure rEnc in terms of E.

$\mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ is computed by picking $s, s_1, s_2 \in \mathbb{Z}_p$ at random and setting

$$C' = \Omega^s \cdot \mathsf{M}, \quad C_0 = \mathsf{Pk}^s, \quad C_1 = T_2^{s_2}, \quad C_2 = T_3^{s-s_1-s_2}, \quad C_3 = T_1^{s_1}$$

$\mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ returns $[C', C_0, C_1, C_2, C_3]$.

We will use the writing $C = \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M}; s, s_1, s_2)$ to denote the ciphertext computed using s, s_1 and s_2 as random choices. $\mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ simply computes

$$C = \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M}) \quad \text{and} \quad U = \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, 1)$$

and returns $[C, U]$.

Procedure $\mathsf{rDec}(\mathsf{Pub}, \mathsf{Sk}, \mathsf{Ct})$**.** As for the encryption procedure we first describe the *basic* decryption procedure $\mathsf{D}(\mathsf{Pub}, \mathsf{Sk}, C)$. Let $C = [C', C_0, C_1, C_2, C_3]$ be a ciphertext. Then $\mathsf{D}(\mathsf{Pub}, \mathsf{Sk}, C)$ returns

$$C' \cdot \mathbf{e}(C_0, D_0) \cdot \mathbf{e}(C_1, D_1) \cdot \mathbf{e}(C_2, D_2) \cdot \mathbf{e}(C_3, D_3).$$

Simple algebra shows that if, $(\mathsf{Pk}, \mathsf{Sk})$ are a pair of public and secret keys output by $\mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$ and $C = \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ then $\mathsf{D}(\mathsf{Pub}, \mathsf{Sk}, C) =$

M. Indeed, we notice that

$$\begin{array}{rcll}
\mathbf{e}(C_0, D_0) & = & \mathbf{e}(\mathsf{Pk}^s, g^{rt_1t_2t_3}) & \\
 & = & & \mathbf{e}(g, \mathsf{Pk})^{rt_1t_2t_3s} \\
\mathbf{e}(C_1, D_1) & = & \mathbf{e}(g^{t_2s_2}, g^{-wt_1t_3}\mathsf{Pk}^{-rt_1t_3}) & \\
 & = & \mathbf{e}(g, g)^{-wt_1t_2t_3s_2} & \cdot \quad \mathbf{e}(g, \mathsf{Pk})^{-rt_1t_2t_3s_2} \\
\mathbf{e}(C_2, D_2) & = & \mathbf{e}(g^{t_3(s-s_1-s_2)}, g^{-wt_1t_2}\mathsf{Pk}^{-rt_1t_2}) & \\
 & = & \mathbf{e}(g, g)^{-wt_1t_2t_3(s-s_1-s_2)} & \cdot \quad \mathbf{e}(g, \mathsf{Pk})^{-rt_1t_2t_3(s-s_1-s_2)} \\
\mathbf{e}(C_3, D_3) & = & \mathbf{e}(g^{t_1s_1}, g^{-wt_2t_3}\mathsf{Pk}^{-rt_2t_3}) & \\
 & = & \mathbf{e}(g, g)^{-wt_1t_2t_3s_1} & \cdot \quad \mathbf{e}(g, \mathsf{Pk})^{-rt_1t_2t_3s_1}
\end{array}$$

and thus

$$\mathbf{e}(C_0, D_0) \cdot \mathbf{e}(C_1, D_1) \cdot \mathbf{e}(C_2, D_2) \cdot \mathbf{e}(C_3, D_3) = \mathbf{e}(g, g)^{-wt_1t_2t_3s} = \Omega^{-s}.$$

Hence,

$$C' \cdot \mathbf{e}(C_0, D_0) \cdot \mathbf{e}(C_1, D_1) \cdot \mathbf{e}(C_2, D_2) \cdot \mathbf{e}(C_3, D_3) = \Omega^s \cdot \mathsf{M} \cdot \Omega^{-s} = \mathsf{M}.$$

The randomizable decryption algorithm $\mathsf{rDec}(\mathsf{Pub}, \mathsf{Sk}, \mathsf{Ct})$ with $\mathsf{Ct} = [C, U]$ simply returns $\mathsf{D}(\mathsf{Pub}, \mathsf{Sk}, C)$.

Procedure $\mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Ct})$**.** We now describe procedure $\mathsf{Randomize}$ used to randomize a ciphertext.

A ciphertext $\mathsf{Ct} = [C, U]$ for key Pk is composed of a basic encryption C of $\mathsf{M} \in \mathbb{G}_T$ and of a basic encryption U of $1 \in \mathbb{G}_T$. Notice that $C \cdot U$ (component-wise multiplication) is a new valid basic encryption of M w.r.t. key Pk. Moreover let $U = [U', U_0, U_1, U_2, U_3]$ be a basic encryption of 1 w.r.t. key Pk. Then, for random $r, r_3, r_2 \in \mathbb{Z}_p$,

$$U^\star = [U'^r, U_0^r, U_1^r T_2^{r_2}, U_2^r T_3^{r_3}, U_3^r T_1^{-r_2-r_3}]$$

is a randomly distributed encryption of 1 w.r.t. the same key. Therefore, to randomize $\mathsf{Ct} = (C, U)$ we compute $(\widehat{C}, U^{\star\star})$ where $\widehat{C} = C \cdot U^\star$ and $U^{\star\star} = (U^\star)^\star$; that is, we apply the randomization of U twice and use the intermediate result $U^\star$ to randomize C. Notice that we do not need to know the public key for which C is intended.

4.2 Semantic security

In this section we prove the following lemma.

Lemma 4.1 *Assume the BDDH assumption. Then tag system* Tag *is semantically secure.*

Proof. Suppose that there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ for which

$$\mathrm{Prob}[\,\mathsf{SSExp}_{\mathcal{A}}(1^k)\,] \geq 1/2 + 1/\mathsf{poly}(k),$$

for some polynomial poly. Then, the following probabilistic polynomial-time algorithm $\mathcal{B}$ breaks BDDH.

1. $\mathcal{B}$ receives the tuple $[\mathcal{I}, g^{z_1}, g^{z_2}, g^{z_3}, Z]$, for random z_1, z_2, z_3, and has to decide whether $Z = \mathbf{e}(g,g)^{z_1 z_2 z_3}$ or Z is random in $\mathbb{G}_T$.
2. $\mathcal{B}$ picks $x, y, t_1, t_2, t_3 \in \mathbb{Z}_p$ at random and sets

   $$\begin{array}{lll} g_0 = Z_1^{-x} g^y, & g_1 = Z_1, & \Omega = \mathbf{e}(Z_1, Z_2)^{t_1 t_2 t_3}, \\ T_1 = g^{t_1}, & T_2 = g^{t_2}, & T_3 = g^{t_3} \end{array}$$

   and

   $$\mathsf{Pk} = g_0 g_1^x.$$

   After this step Pub and Msk are implicitly defined. Indeed, $\Omega = \mathbf{e}(Z_1, Z_2)^{t_1 t_2 t_3}$ implictly sets $\omega = z_1 z_2$. Notice that Pub has the same distribution as the public information given as output by $\mathsf{GenPub}(1^k)$. Moreover, setting $\mathsf{Pk} = g_0 g_1^x$ implies that Pk is a random element of $\mathbb{G}$ as $g_0 g_1^x = g^y$.
3. $\mathcal{B}$ runs $\mathcal{A}$ on input Pub and Pk and receives $\mathsf{M}_0, \mathsf{M}_1$. Then, $\mathcal{B}$ picks $b \in \{0,1\}$ and $s_1, s_2 \in \mathbb{Z}_p$ at random and computes

   $$\begin{array}{lll} C' = Z^{t_1 t_2 t_3} \cdot \mathsf{M}_b, & C_0 = Z_3^y, & C_1 = T_2^{s_2}, \\ & C_2 = Z_3^{t_3} T_3^{-s_1 - s_2}, & C_3 = T_1^{s_1}. \end{array}$$

   Finally, $\mathcal{B}$ sets $C = [C', C_0, C_1, C_2, C_3]$ and runs $\mathcal{A}$ on input $\mathsf{Ct} = [C, U]$ where U is a random encryption of 1 for Pk and receives b' as output.
4. If $b = b'$ then $\mathcal{B}$ guess that $Z = \mathbf{e}(g,g)^{z_1 z_2 z_3}$ otherwise it guess that Z is random.

Observe that if $Z = \mathbf{e}(g,g)^{z_1 z_2 z_3}$ we have that $C = \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M}_b; z_3, s_1, s_2)$. Indeed, we have

$$C' = Z^{t_1 t_2 t_3} \cdot \mathsf{M}_b = \mathbf{e}(g,g)^{z_1 z_2 z_3 t_1 t_2 t_3} \cdot \mathsf{M}_b = \mathbf{e}(Z_1, Z_2)^{t_1 t_2 t_3 z_3} \cdot \mathsf{M}_b = \Omega^{z_3} \cdot \mathsf{M}_b,$$

and

$$C_0 = Z_3^y = g^{yz_3} = \mathsf{Pk}^{z_3} \quad \text{and} \quad C_2 = Z_3^{t_3} T_3^{-s_1-s_2} = T_3^{z_3-s_1-s_2}.$$

Therefore $\mathcal{A}$ will guess b correctly with probability at least $1/2 + 1/\mathsf{poly}(k)$,

On the other hand, if Z is random then C' is independent of b and thus $\mathcal{A}$ will guess b correctly with probability at most $1/2$. This implies that $\mathcal{B}$ breaks the BDDH assumption. □

4.3 Weak untraceability

To prove weak untraceability we show that under the Decision Linear assumption, if we apply the randomization procedure to any ciphertext $\mathsf{Ct} = [C, U]$ we obtain a tuple that is indistinguishable from a random tuple chosen from $(\mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G})^2$. We observe that it is actually enough to prove that for any basic encryption $U = [U', U_0, U_1, U_2, U_3]$ of 1 the tuple $U^{\star\star}$ is indistinguishable from a tuple chosen at random from $\mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$.

We proceed in two steps.

The first step. We prove that the following two distributions are indistinguishable under the BDDH. For any $\mathsf{M} \in \mathbb{G}_T$, define distribution $\mathsf{Dist}_0(1^k)$ as follows:

$$\begin{aligned}\mathsf{Dist}_0(1^k) = \{&(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k);\\ &(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk});\\ &U \leftarrow \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, 1);\\ &C \leftarrow \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M});\\ &[\widehat{C}, U^{\star\star}] \leftarrow \mathsf{Randomize}(\mathsf{Pub}, \mathsf{rPub}, [C, U]) : (\mathsf{Pub}, U, U^{\star\star})\}\end{aligned}$$

while, distribution $\mathsf{Dist}_1(1^k)$ is defined as follows:

$$\begin{aligned}\mathsf{Dist}_1(1^k) = \{&(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k);\\ &(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk});\\ &U \leftarrow \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, 1);\\ &r, s', r_2, r_3 \leftarrow \mathbb{Z}_p;\\ &U^\star = [\Omega^{rs'}, U_0^r, U_1^r T_2^{r_2}, U_2^r T_3^{r_3}, U_3^r T_1^{-r_2-r_3}] : (\mathsf{Pub}, U, U^\star)\}\end{aligned}$$

In the definition of Dist_1 we have denoted by Ω, T_1, T_2, T_3 the components of Pub and by U', U_0, U_1, U_2, U_3 the components of U. Notice that if we write U as $U = \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, 1; s, s_1, s_2)$ then we have

$$U^\star = [\Omega^{rs'}, \mathsf{Pk}^{rs}, T_2^{rs_2+r_2}, T_3^{r(s-s_1-s_2)+r_3}, T_1^{rs_1-r_2-r_3}].$$

That is, $U^\star$ is a ciphertext for a random element of $\mathbb{G}_T$ for public key Pk (specifically, $U^\star$ is an encryption of $\Omega^{rs'-rs}$). Thus, indistinguishability of Dist_0 and Dist_1 can be argued by a reasoning similar to the one employed to prove semantic security.

The second step. We prove that, under the Decision Linear assumption, distributions

$$\begin{aligned}\mathsf{Dist}_2(1^k) = \{&(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k);\\ &(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk});\\ &U \leftarrow \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, 1);\\ &s, s', r_2, r_3 \leftarrow \mathbb{Z}_p;\\ &U^\star = [\Omega^{s'}, \mathsf{Pk}^s, T_2^{r_2}, T_3^{r_3}, T_1^{s-r_2-r_3}] : (\mathsf{Pub}, U, U^\star)\}\end{aligned}$$

and

$$\begin{aligned}\mathsf{Dist}_3(1^k) = \{&(\mathsf{Pub}, \mathsf{rPub}, \mathsf{Msk}) \leftarrow \mathsf{GenPub}(1^k);\\ &(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk});\\ &U \leftarrow \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, 1);\\ &s, s', r_1, r_2, r_3 \leftarrow \mathbb{Z}_p;\\ &U^\star = [\Omega^{s'}, \mathsf{Pk}^s, T_2^{r_2}, T_3^{r_3}, T_1^{r_1}] : (\mathsf{Pub}, U, U^\star)\}\end{aligned}$$

are indistinguishable. Notice that Dist_2 is just a re-writing of Dist_1 and that Dist_3 is the random distribution on $\mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$. Thus, the second step completes the proof that $U^{\star\star}$ is indistinguishable from a random quintuple.
Suppose for sake of contradiction that there exists a probabilistic polynomial-time adversary $\mathcal{A}$ that can distinguish Dist_2 from Dist_3. That is, denoted with $p_2^{\mathcal{A}}(k)$ and $p_3^{\mathcal{A}}(k)$ the probabilities that $\mathcal{A}$ outputs 1 on input a random tuple from $\mathsf{Dist}_2(k)$ and $\mathsf{Dist}_3(k)$ respectively, we have that

$$|p_2^{\mathcal{A}}(k) - p_3^{\mathcal{A}}(k)| \geq 1/\mathsf{poly}(k)$$

for some polynomial poly. Then the following probabilistic polynomial-time algorithm $\mathcal{B}$ breaks the Decision Linear assumption.

1. $\mathcal{B}$ receives the tuple $[\mathcal{I}, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, Z]$, for random exponents $z_1, z_2, z_3 \in \mathbb{Z}_p$, and has to decide whether $Z = g^{z_3+z_4}$ or Z is random in $\mathbb{G}$.

2. $\mathcal{B}$ constructs Pub by picking g_0, g_1 at random from $\mathbb{G}$, Ω at random from $\mathbb{G}_T$, and setting $T_1 = g, T_2 = g^{z_1}, T_3 = g^{z_2}$.

3. $\mathcal{B}$ picks Pk at random from $\mathbb{G}$, and s, s', r_2, r_3 at random from $\mathbb{Z}_p$. $\mathcal{B}$ sets

$$U = [U', U_0, U_1, U_2, U_3] = [\Omega^{s'}, \mathsf{Pk}^s, T_2^{r_2}, T_3^{r_3}, T_1^{s-r_2-r_3}],$$

4. $\mathcal{B}$ picks at random r from $\mathbb{Z}_p$ and computes the tuple

$$\begin{aligned} U^\star &= [U'^r, U_0^r, U_1^r \cdot g^{z_1 z_3}, U_2^r \cdot g^{z_2 z_4}, U_3^r \cdot Z^{-1}] \\ &= [U'^r, U_0^r, U_1^r T_2^{z_3}, U_2^r T_3^{z_4}, U_3^r Z^{-1}]. \end{aligned}$$

5. $\mathcal{B}$ runs $\mathcal{A}$ on input $(\mathsf{Pub}, U, U^\star)$ and returns its output.

We observe that, if $Z = g^{z_3+z_4}$ then we have that

$$\begin{aligned} \hat{U} &= [U'^r, U_0^r, U_1^r T_2^{z_3}, U_2^r T_3^{z_4}, U_3^r T_1^{-z_3-z_4}] \\ &= [\Omega^{rs'}, \mathsf{Pk}^{rs}, T_2^{rr_2+z_3}, T_3^{rr_3+z_4}, T_1^{r(s-r_2-r_3)-z_3-z_4}] \end{aligned}$$

and thus $\hat{U}$ is distributed as in Dist_2. If instead Z is chosen at random from $\mathbb{G}$, then $\hat{U}$ is randomly distributed over $\mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$. We have thus proved the following lemma.

Lemma 4.2 *Assume the Decision Linear assumption. Then tag system* Tag *is weakly untraceable.*

Why strong untraceability is not guaranteed. The scheme described in this section is only weakly untraceable. Let us see where our proof breaks for strong untraceability. The first step of the proof essentially says that distribution $(\Omega^s, \mathsf{Pk}^s)$ with random $s \in \mathbb{Z}_p$ is indistinguishable from distribution $(\Omega^{s'}, \mathsf{Pk}^s)$ with random $s, s' \in \mathbb{Z}_p$. The analogous statement for the case of strong untraceability would have been, for any $A \in \mathbb{G}_T$ and $B \in \mathbb{G}$ ((A, B) is one of the strings given in output by the adversary $\mathcal{A}$ at step 3 of $\mathsf{SUExp}_{\mathcal{A}}$), the distribution (A^s, B^s) with random $s \in \mathbb{Z}_p$ (this is the distribution of the output of the randomizer on input (A, B)) is indistiguishable from the distribution (A^s, B^r) with random $r, s \in \mathbb{Z}_p$ (this is the random distribution on $\mathbb{G}_T \times \mathbb{G}$).

It is easy to see that if A and B are adversarially chosen the assumption is false. In fact, the adversary may choose $A = \mathbf{e}(a, a)$ for random $a \in \mathbb{G}$ and $B = a^b$ for random $b \in \mathbb{Z}_p$. Then, for any $s \in \mathbb{Z}_p$ and for $(C, D) = (A^s, B^s)$, we have $\mathbf{e}(a, D) = C^b$. On the other hand, for random $r, s \in \mathbb{Z}_p$ if $(C, D) = (A^s, B^r)$ then $\mathbf{e}(a, D) = C^b$ with negligible probability.

5 Strong untraceability

In this section we present a transformation that takes the weakly intraceable tag system $\mathsf{Tag} = (\mathsf{GenPub}, \mathsf{GenKey}, \mathsf{rEnc}, \mathsf{rDec}, \mathsf{Randomize})$ of the previous section and tranforms it into a strongly untraceable tag system

$$\mathsf{STag} = (\mathsf{SGenPub}, \mathsf{SGenKey}, \mathsf{SrEnc}, \mathsf{SrDec}, \mathsf{SRandomize}).$$

5.1 The transformation

Our transformation employs a regular semantically-secure encryption scheme $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

Procedure $\mathsf{SGenPub}(1^k)$**.** Execute procedure $\mathsf{GenPub}(1^k)$ and obtain $[\mathsf{Pub}, \emptyset, \mathsf{Msk}]$. Then, execute the key-generation procedure KG of the secure encryption scheme $\mathcal{E}$ and obtain $(\mathsf{rpk}, \mathsf{rsk})$. The output of the procedure is the triple $[\mathsf{SPub}, \mathsf{SrPub}, \mathsf{SMsk}]$ where

$$\mathsf{SPub} = (\mathsf{Pub}, \mathsf{rpk}), \quad \mathsf{SrPub} = \mathsf{rsk}, \quad \text{and} \quad \mathsf{SMsk} = \mathsf{Msk}.$$

Procedure $\mathsf{SGenKey}(\mathsf{SPub}, \mathsf{SMsk})$**.** The key generation procedure takes as input the public information $\mathsf{SPub} = (\mathsf{Pub}, \mathsf{rpk})$ and the master secret key $\mathsf{SMsk} = \mathsf{Msk}$, invokes $\mathsf{GenKey}(\mathsf{Pub}, \mathsf{Msk})$ to obtain $[\mathsf{Pk}, \mathsf{Sk}]$, and returns $[\mathsf{Pk}, \mathsf{Sk}]$.

Procedure $\mathsf{SrEnc}(\mathsf{SPub}, \mathsf{Pk}, \mathsf{M})$**.** The encryption procedure SrEnc takes as input the public information $\mathsf{SPub} = (\mathsf{Pub}, \mathsf{rpk})$, the public key Pk, and a cleartext M, invokes[1] $\mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ to obtain C, and returns the ciphertext $\mathsf{Ct} = [C, \mathsf{Enc}(\mathsf{Pk}, \mathsf{rpk})]$.

Procedure $\mathsf{SrDec}(\mathsf{Pub}, \mathsf{Sk}, \mathsf{Ct})$**.** The decryption procedure SrDec takes as input the public information $\mathsf{SPub} = (\mathsf{Pub}, \mathsf{rpk})$, the private key Sk, and the ciphertext $\mathsf{Ct} = [C_0, C_1]$ and returns[2] $\mathsf{D}(\mathsf{Pub}, \mathsf{Sk}, C_0)$.

Procedure $\mathsf{SRandomize}(\mathsf{SPub}, \mathsf{SrPub}, \mathsf{Ct})$**.** The randomization procedure $\mathsf{SRandomize}$ takes as input the public information $\mathsf{SPub} = (\mathsf{Pub}, \mathsf{rpk})$, the randomizing information $\mathsf{SrPub} = \mathsf{rsk}$, and the ciphertext $\mathsf{Ct} = [C_0, C_1]$ and proceeds as follows.

[1]Recall that $\mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$ is the *basic* encryption procedure used in $\mathsf{rEnc}(\mathsf{Pub}, \mathsf{Pk}, \mathsf{M})$.
[2]Recall that $\mathsf{D}(\mathsf{Pub}, \mathsf{Sk}, C_0)$ is the *basic* decryption procedure used in $\mathsf{rDec}(\mathsf{Pub}, \mathsf{Sk}, \mathsf{Ct})$.

1. Let $\mathsf{Pk} = \mathsf{Dec}(C_1, \mathsf{rsk})$. If decryption fails then return $\perp$ and halt.
2. If $C_0 \notin \mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$ then return $\perp$ and halt.
3. Compute $U = \mathsf{E}(\mathsf{Pub}, \mathsf{Pk}, 1)$.
4. Set $\widehat{C}_0$ equal to the component-wise product of C_0 and U.
5. Set $\widehat{C}_1 = \mathsf{Enc}(\mathsf{Pk}, \mathsf{rpk})$.
6. Return $(\widehat{C}_0, \widehat{C}_1)$.

We next briefly argue the security properties of the tag system STag. Strong Semantic security follows directly from the proof of semantic security of the tag system Tag (see Section 4.2).

Let $\mathsf{Ct} = [C_0, C_1]$ be an adversarially chosen pair. We assume that $C_0 \in \mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$ and that C_1 encrypts public key Pk. If this is not the case then the $\mathsf{SRandomize}$ fails and returns $\perp$. Notice that if $\mathsf{SRandomize}$ does not return $\perp$ then C_0 is a valid encryption of a message M with respect to public information Pub and some public key Pk' (notice that we do not necessarily have that $\mathsf{Pk} = \mathsf{Pk}'$). Let $\widehat{\mathsf{Ct}} = [\widehat{C}_0, \widehat{C}_1]$ be the output of $\mathsf{SRandomize}$. Observe that by the semantic security of Enc, $\widehat{C}_1$ is indistinguishable from an encryption of a random string (of the same length as Pk). In addition, $\widehat{C}_0$ is the encryption of message M' with respect to public key Pk'. We distinguish two case. If $\mathsf{Pk} = \mathsf{Pk}'$ then $\mathsf{M}' = \mathsf{M}$ and, by the weak untraceability of tag system Tag (see Lemma 4.2), $\widehat{C}_0$ is indistinguishable from a random element of $\mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$.

If $\mathsf{Pk} \neq \mathsf{Pk}'$ then $\widehat{C}_0$ is the encryption of a random element M' of $\mathbb{G}_T$ which is indistinguishable from a random element of $\mathbb{G}_T \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G}$. This follows from arguments similar to the ones used to prove the semantic security of Tag (see Lemma 4.1).

We thus have the following theorem.

Theorem 5.1 *Assume the Decision Linear assumption. The tag system* STag *is an untraceable tag system.*

6 Performances Analysis

In this section, we present the results of some experiments that we ran to evaluate the real applicability and the lightness of our schemes for untraceable tags. We also compare them with the scheme presented in [1]. For our experiments, we set up the following small test-bed:

- PC: Intel Core 2 Quad Q6600 2.40 GHz, 3 GB RAM.
- OS: Ubuntu 9.04 - kernel 2.6.28-11-generic - 64 bit.

- PBC Library ver. 0.4.18 [10].
- dcrypt Library ver. 0.3 [3].

In the following tables we summarize the results of our experiments. The second column (e.g., `Weak`) corresponds to the scheme presented in Section 4.1; the third column (e.g., `Strong`) presents the results for the scheme satisfying the strong untraceability property (the scheme is described in Section 5.1); while, the last column (e.g., `InsEnc`) describes the results attained by the Insubvertible Encryption scheme proposed in [1].

	`Weak`	`Strong`	`InsEnc`
Public Information Generation	53.3819	85.0649	103.6559
Key Generation	54.2332	81.1983	24.9269
Encryption	67.1600	49.7901	24.9460
Decryption	46.8101	47.0070	93.4520
Randomization	147.7423	53.9974	116.4845

Table 1: Execution times in milliseconds of the schemes' procedures

For the tests, we set the security parameter to $k = 1206$. Tests were repeated 5000 times. We took the time needed to execute each procedure of an untraceable tag system. In Table 1 we report the average time (expressed in milliseconds) taken by the tests we ran. Considering the randomization procedure, in spite of relying on weaker assumptions, our strong scheme has a better performance, in terms of computational requirements, than the scheme presented in [1]. Our randomization procedure (as well as the decryption one) runs twice faster as the one of [1]. This is very important, as the randomization procedure is invoked quite often (e.g., each time a tag is in proximity of a randomizer); while, all other procedures are invoked just once. Moreover, the randomization procedure is run by special devices (i.e., randomizers) which have low computing power; while, the other procedures are executed by more powerful devices.

	`Weak`	`Strong`	`InsEnc`
Bytes written on tag	1520	1281	364

Table 2: Size in bytes of the encryption

As one can see from Table 2, both our schemes generate an encrypted message (to be written on the tag) of size greater than the one generated by the scheme in [1]. This is not a big concern, as our encrypted messages easily fit in the user memory of currently produced passive RFID tags. For instance, Maxell provides RFID tags whose memory capacity ranges from 128 bytes up to 4K bytes [7]. Moreover, there exists passive RFID having user memory of 32K bytes [12].

7 Extensions and an open problem

Our construction of STag is a special case of a general construction that starts from a *randomizable* anonymous identity-based encryption scheme that enjoys a weak form of security (specifically, security against randomly chosen identities) and turns into an untraceable tag system. Unfortunately, no randomizable anonymous identity-based encryption was known prior to our work, and thus we had to construct our own.

The strong untraceability property defined in this paper does not give any guarantee against randomizers as in experiment SUExp adversary $\mathcal{A}$ has not access to rPub. It would be nice to give a construction which guarantees untraceability against randomizers and whose security is based on mild assumptions.

Nonetheless, as it is not difficult to see, if we use tag system STag, randomizers cannot distinguish between tags carrying encryptions computed with respect to the same public key. This is a very important property since in many applications the public key corresponds to the manufacturer of the object to which the tag is attached. An adversary thus does not need to look at the tag to distinguish objects from different manufacturers and the applicability of tag system STag is not limited.

References

[1] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable rfid tags via insubvertible encryption. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 92–101. ACM, 2005.

[2] Xavier Boyen and Brent Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In *Advances in Cryptology – CRYPTO 2006*, Lecture Notes in Computer Science, pages 290–307. Springer-Verlag, August 2006.

[3] The dcrypt Library. `http://www.scs.cs.nyu.edu/css/lab/dcrypt_fns.html`, Ver. 0.3.

[4] Simson Garfinkel and Beth Rosenberg. *RFID: Applications, Security, and Privacy.* Addison-Wesley Professional, 2005.

[5] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul F. Syverson. Universal re-encryption for mixnets. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2004.

[6] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 1999.

[7] Maxell. Coil-on-Chip RFID. `http://www.maxei.co.jp/products/coc/eng-smal_chip.html`, 2009.

[8] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer-Verlag, August 2003.

[9] Near field communication forum. `http://www.nfc-forum.org/`.

[10] PBC: The Pairing-Based Cryptography Library. `http://crypto.stanford.edu/pbc/`, Ver. 0.4.18.

[11] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, May 1997.

[12] TegoTag. `http://www.tegoinc.com/`, 2009.