

# Untraceable RFID Tags via Insubvertible Encryption

Giuseppe Ateniese  
The Johns Hopkins University  
ateniese@cs.jhu.edu

Jan Camenisch  
IBM Research  
jca@zurich.ibm.com

Breno de Medeiros  
Florida State University  
breno@cs.fsu.edu

## ABSTRACT

We introduce a new cryptographic primitive, called *insubvertible encryption*, that produces ciphertexts which can be randomized without the need of any key material. Unlike plain universal re-encryption schemes, insubvertible encryption prevents against adversarial exploitation of hidden channels, by including certificates proving that the ciphertext can only be decrypted by authorized parties.

The scheme can be applied to RFID tags, providing strong protection against tracing. This enables post-sale applications of manufacturer-issued RFID tags while preserving the privacy of consumers. The functionality required of the RFID tags is minimal, namely that they be re-writable (many-writable). No cryptographic capabilities are required of the tags themselves, as the readers perform all necessary computations.

**Categories and Subject Descriptors:** K.6.5 [Security and Protection]

**General Terms:** Algorithms, Security.

**Keywords:** Universal re-encryption, bilinear maps, RFID privacy.

## 1. INTRODUCTION

“Smart tags” are radio-frequency powered/emitting identification (RFID) that are being phased in as product labels, in substitution of, or in addition to, barcode tags. However, unlike barcodes and other consumer labeling techniques, RFID tags record a sufficiently long bitstring to uniquely identify specific product units, such as a the bottle of medicine sold to a particular patient. Another important feature of RFID tags is that they can be read efficiently at a distance of a few inches to several feet (depending on the technology).

Both of the above-mentioned features present potential benefits to consumers, retailers, and manufacturers. For instance, RFID tags can be read faster than barcode tags

since unlike the latter, they do not require precise reader-tag alignment, making inventory management more efficient. Similarly, the unit-specific feature of RFID tags allows for finer inventory control by, for instance, supporting automated verification of expiration dates.

On the other hand, RFID tags may pose a substantial issue to privacy by enabling distance tracking of specific product units beyond the point-of-sale. For instance, an RFID tag attached to a bottle of medicine or to food packaging would likely not be removed by the user—in fact, there may exist incentives for users to hold on to the RFIDs in the products they buy: One such instance, is that RFID tags could be used to enable more efficient product safety recalls. Over time, consumers might carry enough RFIDs on their body or belongings, and for long enough time, that the movements of each person could be tracked at a distance.

A separate reason for concern is that the “forward channel” of RFID readers can broadcast the value of tag contents several hundred feet away during the reading process. This exacerbates both concerns of individual privacy—a possible scenario would be an eavesdropper situated outside the customs hall in an airport—as well as corporate confidentiality against industrial espionage—for instance, a competitor’s spy sitting in a car in the parking lot of a warehousing facility, holding a powerful directional antenna.

Therefore, it is reasonable to consider the possibility of storing *randomizable contents* into RFID tags. The scheme we propose does not require any modification of the basic functionality of RFID tags, which still would normally broadcast their contents when read. In our protocol, however, these values would be intelligible only to the issuing entity. Moreover, readers would be able to replace the contents of tags with randomized versions at each read operation, protecting the tag from being tracked by unscrupulous parties. In our model, we assume that honest readers are capable of correctly randomizing the tag contents, whether or not they hold the encryption key of the issuing party. Indeed, not even knowledge of the identity of the issuing party is required by our protocol, and accordingly it is not deducible from the contents of the tag.

In this paper, we assume that RFID tags are read/write (many-write), but support no other operations—this is justified by current features implemented in many RFIDs. While perhaps a somewhat pessimistic assumption for future capabilities, it is reasonable to expect that cost pressures will lead manufacturers to implement very few features in the smallest and cheapest devices even in the not-so-close future. Moreover, our technique is not limited to RFID tags

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS’05, November 7–11, 2005, Alexandria, Virginia, USA.  
Copyright 2005 ACM 1-59593-226-7/05/0011 ...\$5.00.

but can be applied to *mixnet* networks, following [17], or to any other type of passive storage device. In each case, the application of the herein proposed techniques has the effect of reducing the potential for abuse of universal re-encryption schemes for the creation of unauthorized hidden channels.

In what follows, we discuss in more detail the security model behind these untraceable, passive tags, and show how our cryptographic constructions achieves the desired privacy objective. We stress that the herein proposed mechanism is the first to achieve provable untraceability when the identity of the issuers is itself possibly sensitive data.

## 2. MOTIVATION

In [17] the authors explicitly considered privacy issues of storing ciphertexts in RFIDs. They define a technique, called *universal re-encryption*, that can be used to randomize ElGamal ciphertexts. Basically, the idea is to store an encryption of a message  $m$  under a public key  $y = g^x$  along with an encryption of “1,” that is:

$$[E_y(m), E_y(1)] = [(my^k, g^k), (y^\ell, g^\ell)].$$

The technique consists of randomizing  $E_y(1)$  and use it to conceal  $E_y(m)$ , exploiting the homomorphic property of ElGamal, i.e.,  $E(m) \times E(m') = E(m \times m')$ . More specifically, one computes  $E'_y(1) = (E_y(1))^r = (y^{\ell r}, g^{\ell r})$ , where  $\ell' = \ell r$ , and  $E'_y(m) = E''_y(1)E_y(m) = (my^{k''}, g^{k''})$ , where  $k'' = k + \ell' r'$ .

A known limitation of universal re-encryption is that it does not prevent an adversary from storing into the RFID tag an encryption of a certain message under his own public key. Even if the ciphertext is randomized several times by other trusted devices or readers, the adversary will always be able to decrypt and recover that message and effectively trace the tag.

## 3. RELATED WORK ON RFID PRIVACY

The relevance of privacy protection for RFID tags has been highlighted in many recent works, for instance in [21, 35, 20, 32, 24, 28, 1, 18, 3]. (See [2] for a complete list of references.)

Several strategies and cryptographic schemes have been proposed to address specifically the traceability problem. These solutions, however, make certain assumptions that may be unrealistic for low-cost RFID tags. For instance, certain techniques require RFID tags capable of performing cryptographic operations (from computing hash functions to encrypting or signing with public keys) or tags equipped with special physical properties (such as with protected memory cells or Faraday cages). Other schemes assume that RFID tags implement access control mechanisms or that they can compare values, such as passwords. Clearly solutions based on *killing* the tag after a certain condition is verified are not suitable for our applications where the information stored in the tag should be preserved.

## 4. OUR CONTRIBUTIONS

We introduce a new cryptographic primitive called *insubvertible encryption* that allows authorized users to store an encryption into a storage device that can be randomized by anyone. Insubvertible encryption provides properties such as semantic security and *key privacy* [7]. Moreover, unlike plain *universal re-encryption* [17], our scheme produces

ciphertexts that contain implicit proofs of being “safe” to randomize. In other words, any randomizer will be able to confirm that the ciphertext is intended for legitimate users; if that check fails the randomizer has the option to obliterate the contents with “safe” but meaningless ciphertexts, destroying the adversarial hidden channel and preventing tracing. This mechanism effectively prevents an adversary from storing data encrypted under his own key in order to trace tags, as long as the adversary is not an authorized user in the system.

To the best of our knowledge, no other work has provided solutions with the same security guarantees and in the case where RFID tags are completely passive and unable to perform any type of computation. Indeed, in our model, RFID tags can be seen as passive storage devices, much like floppy disks or memory cards.

We first provide a formal definition of insubvertible encryption. Next, we provide a rigorous specification of the ideal functionality of a system with untraceable tags within the UC/reactive frameworks [14, 13, 27], and we formally prove that such a functionality is implemented by our construction. In short, we prove that our insubvertible encryption provides several desirable properties, including semantic security and *key privacy*, and then we show that there cannot exist a (real world) adversary who can trace tags unless he can “break” one of the properties of insubvertible encryption.

Finally, we introduce an efficient invertible encoding algorithm that encodes bitstrings into elements of an elliptic curve group.

## 5. DEFINITION AND SCOPE OF THE SOLUTION

We now describe more precisely the security properties that our solution provides. In our scheme, we distinguish between legitimate issuers of *marks* in the RFIDs, and illegitimate ones. The legitimate issuers can initiate and re-set the contents of RFIDs, enabling them to use it for recognizing the tag later. Illegitimate issuers can also re-set the value of tags—these are passive entities—but any contents they write to them will be destroyed by honest readers that participate in the scheme.

We note that legitimate issuers could “re-assign” to them a tag that was initially issued by a different (legitimate) issuer. We provide no protection against this, but such a threat is mitigated in practice by the need of legitimate issuers to maintain their reputation within the system. Legitimate issuers receive a certificate from the system authority that they include as part of their *marks*, after proper randomization for unlinkability. While the certificate is anonymous even to other issuers, its validity can be readily ascertained by anyone.

Attached to the randomized certificate, there is an encrypted message, namely the proper tag identifier, and pertinent related information. The randomized certificate is useful as a public key for re-encryption of the message, in a manner similar to the universal re-encryption scheme.

We first give an informal description of the functionality of our scheme. We then formalize that as an ideal functionality in the spirit of the UC/reactive frameworks [14, 13, 27] and then in Section 7.1 prove that our scheme securely implements this functionality.

## 5.1 Definition of Insubvertible Encryption

Each legitimate issuer  $\mathcal{P}_i$  of RFID tags should generate a public key and have it certified by the authority, which has public key  $\mathcal{C}$ . Each issuer is allowed to track tags initiated (marked) by itself. On the other hand, no other party should be able to *mark* RFID tags. By that we mean that only certified public keys may store a *mark* in the tag that is traceable beyond the next interaction with an honest reader. To this end, we require each honest reader to re-randomize the mark.

Note that it is impossible to require a stronger property than “traceable until next interaction with honest reader,” because we assume that the RFID tags are passive. Therefore, a dishonest reader may always overwrite the contents of the tag with arbitrary data; as long as the tag is not presented to a legitimate reader, other colluding readers will be able to recognize the same data and identify the device.

It is also not possible to provide *mark integrity protection* as we assume that the tags are passive, and therefore cannot verify whether the contents to be written to them are legitimate. On the other hand, we require that, if a malicious reader adulterates the contents of a tag destined to a legitimate issuer, that this fact be detectable. By adulteration, we mean changing the stored ciphertext in any way that does not follow the specified protocol. Because the encrypted marks can be destroyed or overwritten by malicious parties, and randomized by honest parties, but not used by an illegitimate party for unintended purposes, we say that they are *insubvertible*, and the corresponding cryptographic construct is called *insubvertible encryption*.

*Insubvertible encryption* consists of a set of efficient algorithms:

**GenerateCAKey:** The certification authority  $\mathcal{C}$  generates a private-public key pair  $(\mathcal{C}_{SK}, \mathcal{C}_{PK})$  for use in the protocol via a randomized algorithm:

$$(\mathcal{C}_{SK}, \mathcal{C}_{PK}) \xleftarrow{\$} \{0, 1\}^k.$$

**GenerateKey:** An issuer  $\mathcal{P}_i$  generates a private-public key pair  $(SK_i, PK_i)$  for use in the protocol via a randomized algorithm:

$$(SK_i, PK_i) \xleftarrow{\$} \{0, 1\}^{\bar{k}}.$$

**RegisterPublicKey:** An issuer  $\mathcal{P}_i$  obtains certificate  $u$  on public key  $PK_i$  by interacting with the authority  $\mathcal{C}$ . The inputs to the algorithm are  $PK_i$  and the secret key of the authority,  $\mathcal{C}_{SK}$ .

$$u \leftarrow PK_i, \mathcal{C}_{SK}$$

**InitiateTag (Encrypt):** An empty tag  $D_j$  is made to store a ciphertext (mark) produced by an authorized issuer  $\mathcal{P}_i$ . The mark  $d$  encrypts a certificate  $u$ , and message  $m$ :

$$d \leftarrow u, m, PK_i$$

**ReadAndDecrypt:** A previously generated ciphertext (mark) is read and, if valid, decrypted.

$$\{m\} \cup \perp \leftarrow d, SK_i,$$

where  $m$  is returned if  $d$  was previously computed as a function of  $u$ , and  $m$ , where the certificate  $u$  corresponds to the provided decryption key  $SK_i$ . If  $d$  is invalid, or if the key  $SK_i$  in decryption differs from the  $SK_j$  involved in encryption,  $\perp$  is returned.

**ReadAndRandomize:** Randomizes ciphertexts (marks) that have been previously generated. The inputs to the algorithm are the existing mark,  $d$ , and the public key of the authority,  $\mathcal{C}_{PK}$ .

$$d' \xleftarrow{\$} d, \mathcal{C}_{PK}$$

Note that we refer to encryption as the *tag initialization*, since that is the effect it has on tags—resetting and discarding previous contents it may have carried up to that point.

## 5.2 Security Formalization: Specification of Ideal Functionality and Adversaries

Apart from the the certification authority  $\mathcal{C}$ , and the set of issuers  $\mathcal{P}_i$ , we denote the reader/randomizers by  $\mathcal{R}_i$ , and the tags by  $\mathcal{D}_i$ .

The adversary is denoted by  $\mathcal{A}$ , and may corrupt any of the randomizers, but neither the issuers nor the certification authority. The adversary subsumes all dishonest parties.

The environment  $\mathcal{E}$  determines all the inputs of the parties and obtains all their outputs. The environment also instructs each of the honest parties  $I_i$ ,  $R_i$ , and  $\mathcal{D}_i$  what action they need to perform. The environment similarly instructs the adversary  $\mathcal{A}$ —which may, however, deviate from the protocol in an arbitrary fashion when computing its outputs.

In the ideal world, there is a trusted party  $\mathcal{T}$  through whom all parties interact. In particular, in the ideal world, values stored in tags are not visible to the adversary directly, and instead it must interact with  $\mathcal{T}$  to obtain readings.

$\mathcal{T}$  keeps a list of what tag  $\mathcal{D}_i$  stores. That is with each tag  $\mathcal{T}$  stores either a triple  $(d, u, m)$  (where  $d \in \{0, 1\}^{\ell_i}$  is a temporary identifier and  $u$  is an index corresponding to a registered public key and  $m$  a message), a triple  $(d, \perp, \perp)$  (indicating that the tag was reset or is empty and where again  $d \in \{0, 1\}^{\ell_i}$  is a temporary identifier), or an arbitrary string  $\rho$  (that the adversary stored on the tag).

Upon initialization,  $\mathcal{T}$  choses a random identifier  $d$  and stores  $(d, \perp, \perp)$  with each  $\mathcal{D}_i$ . Afterwards, the parties  $\mathcal{P}_i$ ,  $\mathcal{R}_i$ , and  $\mathcal{D}_i$  can request one the following actions from  $\mathcal{T}$ .

**RegisterPublicKey( $\mathcal{P}_j$ ):** An issuer  $\mathcal{P}_j$  wants to register a new public key.  $\mathcal{T}$  assigns the next free index  $u$  to  $\mathcal{P}_j$  and sends  $u$  to  $\mathcal{P}_j$ .

**InitiateTag( $\mathcal{P}_j, u, m$ ):** Tag  $\mathcal{D}_i$  wants to be initiated by  $\mathcal{P}_j$  with message  $m$  and public key index  $u$ .  $\mathcal{T}$  checks whether  $u$  is assigned to  $\mathcal{P}_j$  and, if so, chooses a fresh identifier  $d$  and stores the tuple  $(d, u, m)$  with  $\mathcal{D}_i$ .

**ReadAndDecrypt( $\mathcal{P}_j$ ):** Tag  $\mathcal{D}_i$  asks to be read and decrypted by party  $\mathcal{P}_j$ . (Note that  $\mathcal{P}_j$  is honest by assumption.) If  $\mathcal{T}$  has stored a tuple  $(d, u, m)$  with  $\mathcal{D}_i$  and  $u$  is a index to a key assigned to  $\mathcal{P}_j$ , then  $\mathcal{T}$  sends the message  $m$  to  $\mathcal{P}_j$ . Otherwise,  $\mathcal{T}$  sends  $\perp$  to  $\mathcal{P}_j$ .

**ReadAndRandomize( $\mathcal{R}_j$ ):** A tag  $\mathcal{D}_i$  asks to be read and randomized by party  $\mathcal{R}_j$ .

First, consider the case where  $\mathcal{R}_j$  is honest. If  $\mathcal{T}$  has stored a  $(d, u, m)$  with  $\mathcal{D}_i$  for some  $u$  and  $m$ , it chooses a fresh identifier  $d'$  and stores  $(d', u, m)$  with  $\mathcal{D}_i$  (however,  $\mathcal{T}$  will keep the triple  $(d, u, m)$  in its database). Otherwise,  $\mathcal{T}$  chooses a fresh identifier  $d'$  and stores  $(d', \perp, \perp)$  with  $\mathcal{D}_i$ . (Note that if  $\mathcal{T}$  had stored a string  $\gamma$  or  $(d, \perp, \perp)$  with  $\mathcal{D}_i$ , it now stores  $(d', \perp, \perp)$ .)

Assume now that  $\mathcal{R}_j$  is dishonest. If  $\mathcal{T}$  has stored a tuple  $(d, u, m)$  with  $\mathcal{D}_i$ , then  $\mathcal{T}$  sends the current index  $d$  to  $\mathcal{R}_j$ . If  $\mathcal{T}$  has stored a string  $\gamma$  with  $\mathcal{D}_i$ , then  $\mathcal{T}$  sends  $\gamma$  to  $\mathcal{R}_j$ .

Furthermore, it asks  $\mathcal{R}_j$  for a tuple  $(a, b)$ . The value  $a$  indicates which action the reader desires to perform. We consider each case separately:

- $a = 1$ : If  $\mathcal{T}$  has stored a triple  $(d', u, m)$  with  $d' = b$  for some  $u$  and  $m$  in its database, then  $\mathcal{T}$  stores the tuple  $(d', u, m)$  with  $\mathcal{D}_i$ . (Note that  $\mathcal{T}$  could have stored  $(d', u, m)$  with another tag that the adversary had read and now wants to copy in this tag.)
- $a = 2$ : If  $\mathcal{T}$  has stored a triple  $(d', u, m)$  with  $d' = b$  for some  $u$  and  $m$  in its database,  $\mathcal{T}$  chooses a fresh identifier  $\tilde{d}$  and sends  $\tilde{d}$  to  $\mathcal{R}_j$ . Also,  $\mathcal{T}$  stores the tuple  $(\tilde{d}, u, m)$  with  $\mathcal{D}_i$ .
- $a = 3$ :  $\mathcal{T}$  stores the tuple  $(d = b, \perp, \perp)$  with  $\mathcal{D}_i$ .
- $a = 4$ :  $\mathcal{T}$  chooses a fresh identifier  $\tilde{d} \in \{0, 1\}^{\ell_i}$  and sends  $\tilde{d}$  to  $\mathcal{R}_j$ .  $\mathcal{T}$  stores the tuple  $(\tilde{d}, \perp, \perp)$  with  $\mathcal{D}_i$ .
- $a = 5$ :  $\mathcal{T}$  stores the string  $\gamma = b$  with  $\mathcal{D}_i$  (actually,  $\gamma$  could be any string or  $\perp$ ).
- $a \notin \{1, 2, 3, 4, 5\}$ : Otherwise,  $\mathcal{T}$  rejects.

Notice that when  $\mathcal{T}$  chooses a fresh identifier  $d$ , it does so randomly from the set  $\{0, 1\}^{\ell_i}/D$ , where  $D$  is the set of already assigned identifiers.

We assume that the adversary cannot eavesdrop on the conversation between the honest randomizers and a tag. Indeed, if the adversary can eavesdrop on all conversations, all hope for privacy is lost. Nevertheless, an adversary eavesdropping on some randomizer is accounted for by considering that randomizer corrupted and  $(a = 2, b = d)$  is sent to  $\mathcal{T}$  in the `ReadAndRandomize`( $\mathcal{R}_j$ ) operation.

Let us now argue that the ideal model we just described achieves the properties we claimed.

**Privacy.** We require that once a tag is read and randomized by an honest reader, it cannot be traced by the adversary. That is, if between two reads of the tag by an adversary, the tag is read and randomized by an honest reader/randomizer, then the adversary is not able to distinguish whether or not it reads the same tag. It is not hard to see that this is achieved by our ideal model: The only information the adversary ever gets about a tag is either a random temporary identifier or a string that it had written onto the tag. However, once the tag was read or randomized by an honest reader,  $\mathcal{T}$  will have assigned a new temporary identifier to it and thus, if the adversary reads it again, it will see a new temporary identifier which cannot be linked to previous ones.

**Security.** We require that the adversary cannot put an (encrypted) message onto the tag other than copying the

message from another tags. It is straightforward to see that this holds for our ideal model.

## 6. CONSTRUCTION

### 6.1 Cryptographic assumptions

The construction of insubvertible encryption takes place in DDH-hard subgroups of elliptic curves with pairings, a cryptographic setting used in [30, 4], as well as in [8] (only in the full version). We first review some general concepts of pairing groups.

Bilinear maps [9, 11, 19, 15] are defined as follows:

**DEFINITION 6.1 (BILINEAR MAP).** We say a map  $e : G_1 \times G_2 \rightarrow G_T$  is a *bilinear map* if:

1.  $G_1$  and  $G_T$  are groups of the same prime order  $p$  and  $e(\cdot, \cdot)$  is efficiently computable;
2. for all  $a, b \in \mathbb{Z}_p$ ,  $g \in G_1$ , and  $h \in G_2$ , then  $e(g^a, h^b) = e(g, h)^{ab}$  (bilinearity);
3.  $e(\cdot, \cdot)$  is non-degenerate, i.e., if  $g$  generates  $G_1$  and  $h$  generates  $G_2$ , then  $e(g, h)$  generates  $G_T$ .

A pairing allows for efficient solution of the following decisional Diffie-Hellman type problem, called the co-DDH assumption: Given  $g, \tilde{g}$  in  $G_1$ , and  $h, \tilde{h}, h'$  in  $G_2$ , decide if there are exponents  $a$  and  $b$  in  $\mathbb{Z}_p$  such that  $\tilde{g} = g^a$ ,  $\tilde{h} = h^b$ , and  $h' = h^{ab}$ . In many settings, the bilinear maps also enable the solution of the (standard) DDH problem within either group. This is because in such settings a homomorphism  $\sigma : G_1 \rightarrow G_2$  (called a distortion map) allows the pairing to be computed within  $G_1$ . These distortion maps do not exist, however, in all pairing groups, as shown in [33, 34, 16].

To the extent of our knowledge, the first use of DDH-hard pairing groups in cryptography is in an id-based key exchange protocol by M. Scott [30]. That paper cites E. R. Verheul's result [33] as evidence of the existence of such groups, and points to several algorithms proposed in the literature for generating appropriate curves and subgroups [5, 23]. Boneh et al [8] remark that the use of DDH-hard groups could be used to simplify their short group signature constructions, providing higher efficiency.

In [4], Ballard et al. propose a system for correlation-resistant storage based on DDH-hard pairing groups, and provide a detailed formulation of the assumption that DDH-hard pairing groups exist, naming it the “external Diffie-Hellman” assumption, or XDH for short. We follow their terminology:

**DEFINITION 6.2 (XDH assumption [4]).** Let  $G_1$  and  $G_2$  admit a bilinear map, as above. One says that the XDH assumption holds if  $G_1$  is a DDH-hard group, i.e., it is not computationally feasible, given a set of values  $y, y_1, y_2, y_3$  in  $G_1$ , to decide if there are integers  $a, b$  such that  $y_1 = y^a$ ,  $y_2 = y^b$ , and  $y_3 = y^{ab}$ .

XDH groups can be chosen as specific subgroups of MNT curves [23], which have optimized implementations in standing cryptographic libraries [26, 29]. Recently, Barreto et al. have discovered more efficient constructions [6]. These new curves may indeed constitute the most efficient setting to

date in which to instantiate any pairing-based cryptographic protocol.

The unforgeability of certificates released by the certification authority requires the following assumption:

**DEFINITION 6.3.** (*Strong LRSW Assumption*): Let  $G_1$  and  $G_2$  be groups that admit a bilinear map and with no morphisms between them. Let  $S, T \in G_2$ ,  $S = \tilde{g}^s$ ,  $T = \tilde{g}^t$ . Let  $O_{S,T}(\cdot)$  be an oracle that, on input a value  $x \in \mathbb{F}_p$ , outputs a quintuple  $A = (a, a^t, a^{s+xt}, a^x, a^{xt})$  for a randomly chosen  $a \in G_1$ . Then for all probabilistic polynomial time adversaries  $\mathcal{A}$ ,  $\nu(k)$  defined as follows is a negligible function:

$$\begin{aligned} & \Pr[(p, G_1, G_2, g, \tilde{g}) \leftarrow \text{GenerateKey}(1^k); \\ & \quad s \leftarrow \mathbb{F}_p; t \leftarrow \mathbb{F}_p; S = \tilde{g}^s; T = \tilde{g}^t; \\ & (a_1, a_2, a_3, a_4, a_5) \leftarrow \mathcal{A}^{O_{S,T}}(p, G_1, G_2, g, \tilde{g}, S, T) : \\ & \quad x \notin Q \wedge x \in \mathbb{F}_p \wedge x \neq 0 \wedge a_1 \in G_1 \wedge a_2 = a_1^t \\ & \quad \wedge a_3 = a_1^{s+xt} \wedge a_4 = a_1^x \wedge a_5 = a_1^{xt}] = \nu(k), \end{aligned}$$

where  $Q$  is the set of queries that  $\mathcal{A}$  made to  $O_{S,T}(\cdot)$ .

This assumption is a stronger version of the LRSW assumption that was introduced by Lysyanskaya et al. [22] and subsequently used in [12] to build a group signature based on bilinear maps. This assumption is stronger than the original one because here the adversary is not required to output  $\log_{a_1} a_4$  as in the LRSW assumption. We stress that the Strong LRSW assumption is defined over two groups,  $G_1$  and  $G_2$ , that admit a bilinear map and in the case where there are no morphisms between them (in both directions). It not hard to see that the Strong LRSW assumption can be proven to hold in the generic group model [25, 31]. Nevertheless, in the full version of this paper, we will show how to allow morphisms from  $G_2$  to  $G_1$  to exist (in which case, the Strong LRSW assumption does not hold) and rely on a weaker assumption.

## 6.2 Authentication and Encoding of Messages

Our scheme also specifies an authentication algorithm. In the description of the scheme, we assume this is a message authentication code, but we point out that the same results would follow if a signature scheme is used instead. We denote the process of authenticating a message using key  $K_i$  belonging to issuer  $\mathcal{P}_i$  as  $\text{MAC}(K_i, m)$ . The MAC length will be denoted by  $t$ , and reasonable choices for  $t$  would be in the range of 64 – 128 bits depending on the security requirements.

Part of the specification is also an invertible encoding algorithm that we describe in a later section §6.5. For simplicity of notation, we will write  $\text{Enc}(K, M)$  to denote the combined encoding of the message and its authentication tag under the MAC.

## 6.3 Elliptic curve parameters

Following [4], we choose an ordinary elliptic curve  $E$  of low embedding degree (such that the pairings are efficiently computable). Such curves, called MNT curves after [23], can be generated using the complex multiplication (CM) method. This elliptic curve  $E$ , defined over the field  $\mathbb{F}_q$ , has the following characteristics: There is a “large” subgroup  $G_1$

of prime order  $p$  in  $E(\mathbb{F}_q)$ .<sup>1</sup> Its embedding degree  $\ell$  is the smallest integer such that a pairing  $e : G_1 \times G_2 \rightarrow G_T = \mathbb{F}_q^\ell$  exists. (For good security/efficiency trade-offs the current recommendation is  $\ell = 6$  or slightly larger.) We note that  $G_2$  is a subgroup of  $E(\mathbb{F}_q)$ .

## 6.4 Description of the Scheme

We use traditional multiplicative group notation, instead of the additive notation often used in elliptic curve settings. That is, if  $g$  is an element of  $G_1$ , we write  $g^x$  to denote exponentiation by  $x \in \mathbb{F}_p$ .

The following public parameters are shared by all: The prime numbers  $p$  and  $q$ , the elliptic (MNT) curve  $E$ , the groups  $G_1$ ,  $G_2$ , and  $G_T$ . We also assume that generator  $g$  has been chosen in  $G_1$  and generator  $\tilde{g}$  in  $G_2$ , both independently. The *group size* is the value  $k = \lfloor \log_2 p \rfloor$ , and if  $G_1$  coincides with  $E(\mathbb{F}_q)$ , then  $k$  also equals  $\lfloor \log_2 q \rfloor$ ,<sup>2</sup> the elliptic curve *key length*  $k'$ . In any case, since the index of  $G_1$  in  $E(\mathbb{F}_q)$  is small,  $k$  and  $k'$  are close values.

The first two steps consist of generating the necessary public-private key pairs for use in the protocol.

**CA Key Generation (GenerateCAKey):** The private-public key pairs of the authority  $\mathcal{C}$  are as follows:

$$\mathcal{C}_{SK} = (s, t), \mathcal{C}_{PK} = (S = \tilde{g}^s, T = \tilde{g}^t, C),$$

where  $s$  and  $t$  are randomly chosen from  $\mathbb{F}_p$ ;  $S, T$  are in  $G_2$ ; and  $C = (a_1, a_2, a_3, a_4, a_5, c_1, c_2)$  is a *dummy* insubvertible encryption and consists of a certificate  $(a_1, a_2, a_3, a_4, a_5)$  (defined below in RegisterPublicKey), for a randomly chosen key  $y \in G_1$ , and random values  $c_1$  and  $c_2$  chosen in  $G_1$ .

**Issuer Key Generation (GenerateKey):** Issuers of tags generate public keys in  $G_1$ . Each private-public key pair is of ElGamal type; the public key of issuer  $\mathcal{P}_i$  is therefore:

$$(SK_i, PK_i) = (x_i, y_i),$$

where  $x_i$  is randomly chosen from  $\mathbb{F}_p$ , and  $y_i = g^{x_i}$  in  $G_1$ . The key registration algorithm (next) requires the XDH assumption for security, and involves a new construction.

**Key Registration (RegisterPublicKey):** The authority  $\mathcal{C}$ , after verifying the identity of  $\mathcal{P}_i$  and its claim to public key  $y_i$  (where  $y_i = g^{x_i}$ ), computes a random  $w \in \mathbb{F}_p$  and set  $a = g^w$  (in  $G_1$ ). The certificate on the public key  $y_i$  is therefore the quintuple:

$$(a_1, a_2, a_3, a_4, a_5) \leftarrow (a, a^t, a^{s+x_i st}, a^{x_i}, a^{x_i t}) \in G_1^5.$$

In order to verify that the certificate has indeed been generated by the authority, any party can check that:

$$(1) e(a_1, T) = e(a_2, \tilde{g}), \quad (2) e(a_4, T) = e(a_5, \tilde{g}),$$

$$\text{and } (3) e(a_3, \tilde{g}) = e(a_1 a_5, S).$$

**Encryption (InitiateTag):** Issuer  $\mathcal{P}_i$ , with public key  $PK_i = y_i$  and certificate  $(a_1, a_2, a_3, a_4, a_5) = (a, a^t, a^{s+x_i st}, a^{x_i}, a^{x_i t})$

<sup>1</sup>In the optimal case  $G_1 = E(\mathbb{F}_q)$ , but small index values such as 2 or 4 are also allowed, and if index values different from 1 are used, the encoding algorithm must accordingly accommodate this—see 6.5 for details.

<sup>2</sup>This follows from the Hasse-Weil inequality:  $|p - q - 1| \leq 2\sqrt{q}$ .

with  $a \in G_1$ , encrypts a message  $m \in \{0, 1\}^b$  as follows: First, it generates a random value  $r \in \mathbb{F}_p$ , and uses it to randomize the certificate:

$$(a_1, a_2, a_3, a_4, a_5) \leftarrow (a_1^r, a_2^r, a_3^r, a_4^r, a_5^r)$$

Next, the issuer computes the authentication code for  $m$  under its master authentication key  $K$ , i.e.,  $T = \text{MAC}(K, m)$ . The issuer encodes both  $m$  and  $T$  as a single point of  $G_1$ , using the encoding scheme described in section §6.5. Let  $M$  be the corresponding point in  $G_1$  (encoded-and-authenticated version of message). The issuer encrypts  $M$  using ElGamal, with public key  $y_i$  and random value  $k \in \mathbb{F}_p$ :

$$(c_1, c_2) \leftarrow (g^k, My_i^k).$$

The insubvertible encryption of  $m$  is then:

$$(a_1, a_2, a_3, a_4, a_5, c_1, c_2) \leftarrow (a^r, a^{rt}, a^{r(s+x_i st)}, a^{rx_i}, a^{rx_i t}, g^k, My_i^k) \in G_1^7.$$

**Randomization (ReadAndRandomize):** Obtain  $(a_1, a_2, a_3, a_4, a_5, c_1, c_2)$  from the tag. First, verify that  $(a_1, a_2, a_3, a_4, a_5)$  is a valid certificate via the verification procedure given in the **Key Registration** algorithm above. If the certificate is invalid, retrieve the dummy insubvertible encryption  $C$  from the public key of the certification authority and set  $(a_1, a_2, a_3, a_4, a_5, c_1, c_2) \leftarrow C$ . Then, generate random values  $v$  and  $z$  in  $\mathbb{F}_p$  and compute the new encryption:

$$(a_1, a_2, a_3, a_4, a_5, c_1, c_2) \leftarrow (a_1^v, a_2^v, a_3^v, a_4^v, a_5^v, a_1^z c_1, a_2^z c_2).$$

It should be clear from the discussion that the randomization does not change the encrypted value.

**Decryption (ReadAndDecrypt):** To decrypt  $(a_1, a_2, a_3, a_4, a_5, c_1, c_2)$ , the issuer first verifies the randomized certificate (same test as in the **Key Registration** algorithm above), then checks that the certificate is for its own public key, verifying that  $a_1^{x_i} = a_4$ .

If so, it proceeds to decrypt  $M = c_2 / (c_1)^{x_i}$ , and then decode the point  $M$  into the original message  $m$ , while checking the MAC in the process. (See full details in the encoding section.) If all tests succeed, the message  $m$  is returned. Otherwise, failure is reported.

*Discussion of the Scheme.* Insubvertible encryption consists of a certificate,  $(a_1, a_2, a_3, a_4, a_5)$ , attached to an ElGamal encryption,  $(c_1, c_2)$ . Only authorized users possess valid certificates and have the ability to create valid insubvertible encryptions. Certificates have the property of being randomizable, i.e., anyone can generate a new certificate on the same public key starting from a valid one.

If the certificate is invalid, the randomizer will substitute the content of the tag with a randomized version of a dummy (in the sense that it is meaningless but valid) insubvertible encryption from the public key of the authority  $C$ .

An adversary may try to reuse an existing and valid certificate in order to avoid detection and hide some tracing value in the encryption part of the insubvertible encryption. However, the randomization procedure is designed to use elements of the certificate itself to randomize the encryption to ensure that any hidden tracing value is corrupted during randomization.

Finally, notice that the key registration phase is very simple since it is run between mutually trusted parties, i.e., in our model, we do not consider attacks from insiders.

## 6.5 Encoding

We now introduce a new encoding of bitstrings into elements of an elliptic curve group. This encoding is inspired by the hashing scheme introduced by Boneh et al. in [11, 10], but unlike the latter it is invertible (i.e., admits an efficient decoding). We believe this construction is of independent interest.

**Encode-to-Group**( $\cdot$ ) to embed binary strings of fixed length  $b$ , where  $b < k - t - 1$ , into the group  $G_1$ . For a fixed *encoding bandwidth*  $b$ , define  $c = k - b - t - 1$ , and let  $C$  be a  $c$ -bit counter. The encoding algorithm works as follows:

Given a  $b$ -bit message  $M$ , set  $C$  to zero, i.e., to the  $c$ -bit string consisting of only zeros. (If  $M$  is not  $b$ -bits long,  $M$  should be padded via a reversible padding scheme.) Let  $M' = M || C$ , and  $T = \text{MAC}(K, M')$ , where  $K$  is the authentication key of the encoder. The value  $M'' = M' || T = M || C || \text{MAC}(K, M || C)$  will be at most  $k - 1 \leq k' - 1$  bits long, and may be interpreted as the binary expansion of an integer  $X$  smaller than  $q$ , hence a unique value in  $\mathbb{F}_q$ . One then tries to set  $X$  as the  $x$ -coordinate of a point in  $E(\mathbb{F}_q)$ , i.e., checks if the equation  $E(x, y) = 0$  defining the elliptic curve  $E$  has a  $\mathbb{F}_q$ -rational solution for  $y$  when  $x = X$ .<sup>3</sup> If so, then let  $P = (X, Y)$  be the elliptic point. Let  $s$  be the index of  $G_1$  in  $E(\mathbb{F}_q)$  (often  $s = 1$ ). The value  $Q = P^s$  will be a point in  $G_1$ , and by definition, the value of **Encode-to-group**( $M$ ) :=  $Q$ .

If the above check fails, one increments the counter  $C$  and repeats. Since 50% of the values in a finite field admit square roots, the probability that the encoding operation altogether fails is  $2^{-2^c}$ ; this parameter can be tuned with the other performance parameters for satisfactory practical operation.

To invert the encoding operation, we can use the **Decode-from-Group** algorithm. Given a point  $Q$ , we first note that, since the order of  $G_1$  is a large prime  $p$  and the index  $s$  is a small integer, it follows that  $\gcd(s, p) = 1$ . Let  $s^{-1}$  be the inverse of  $s \bmod p$ . Then, if  $P$  is any point of  $E(\mathbb{F}_q)$  such that  $P^s = Q$ , it must be that  $P = Q^{s^{-1}} \cdot S$ , where  $S$  is an  $s$ -torsion point of  $E(\mathbb{F}_q)$ , i.e.,  $S^s = O$ , the identity. Since all  $s$ -torsion points can be efficiently computed<sup>4</sup> (and in fact, should be pre-computed once), it is possible to find all points  $P$  with  $P^s = Q$  efficiently. Let  $P$  be one such point, and read its  $x$ -coordinate  $X = X(P)$ . Reading the string which is the binary representation of  $X$ , parse it as  $M || C || T$ , and verify that  $T = \text{MAC}(K, M || C)$ . If the verification succeeds, return  $M$ . If after parsing all  $P$ 's such that  $P^s = Q$  nothing verifies against the MAC, report failure.

Note that, since decoding (potentially, if  $s > 1$ ) tries several matches against the message authentication code value, the possibility of forging a valid authentication tag is increased. However, since  $s$  is fairly small (in most cases,  $s = 1, 2$ , or  $4$ ), this increase is negligible.

## 7. ANALYSIS

In this section we prove that it is possible to implement a system of untraceable tags with our insubvertible encryption scheme. The idea is to show that if an adversary is in

<sup>3</sup>There will typically be two solutions or none. When two solutions are available, take one of them in a systematic manner—for instance, the one whose binary expression encodes the smallest integer.

<sup>4</sup>Note that  $s$  is a very small value.

```

Encode-to-Group( $K, M$ ):
   $C \leftarrow \underbrace{0 \dots 0}_{c \text{ bits}} ; \quad done \leftarrow \text{False};$ 
  While Not  $done$  And  $C < 2^c$  Do
     $T = \text{MAC}(K, M || C)$ 
     $X = \text{O2I}(M || C || T)$ 
    If  $Y = \text{Solve}[E(X; y) = 0]$  Then
       $P \leftarrow (X, Y)$ 
       $Q \leftarrow P^s$ 
      Return  $Q$ 
    Else
      Increment  $C$ 
  Return  $\perp$ 

```

```

Decode-from-Group( $K, Q$ ):
  For  $P$  such that  $P^s = Q$  Do
     $X = X(P)$ 
     $M' = \text{I2O}(X)$ 
     $M || C || T \leftarrow M'$ 
    If  $T = \text{MAC}(K, M || C)$ 
      Return  $M$ 
  Return  $\perp$ 

```

**Table 1: The encode and decode algorithms. The function O2I maps octet-strings (byte strings) to numbers in  $\mathbb{F}_q$  and I2O is its reverse.**

any way successful to trace a tag, then we can “break” at least one of the properties of insubvertible encryption. These properties are “key privacy” [7], semantic security, and unforgeability of key registration, i.e., that certificates issued on public keys by the certification authority are unforgeable. We first show that our insubvertible encryption enjoys them and then prove the overall security of our untraceable tags scheme.

**Semantic Security.** It is easy to see that our insubvertible encryption scheme inherits this property from the ElGamal encryption scheme. The pair  $(c_1, c_2)$  in our scheme is effectively an ElGamal encryption in  $G_1$  in which DDH is hard. It is well-known that ElGamal encryption is semantically secure under the DDH assumption .

**Unforgeability of Certificates.** It should also be clear that the public key registration scheme is resistant to forgery, even when the adversary has the benefit of acquiring certificate examples. In fact, this statement is logically equivalent to the Strong LRSW assumption. (Notice that the certificate in our scheme is a natural extension of the signature proposed by Camenisch and Lysyanskaya in [12] which relies on the LRSW assumption.)

**Key Privacy.** Next, we show that the scheme provides key-privacy [7], i.e., that it is computationally difficult to distinguish between two samples of marks, when the two samples are created using different issuer public keys. We show that this property follows from the assumption that  $G_1$  is DDH-hard.

LEMMA 7.1. *Under the DDH assumption in  $G_1$ , our in-*

*subvertible encryption scheme provides key privacy.*

PROOF. In order to prove that encryptions under distinct keys are indistinguishable (i.e, that the scheme is key-private), we need to show that given two public keys  $PK_1$  and  $PK_2$  and an insubvertible encryption  $\mu$  of  $m$ , one cannot tell whether or not  $\mu$  is an encryption under  $PK_1$  or  $PK_2$ .

More precisely, we need to show that for sufficiently large  $k$ , any message  $m$ , and any  $C_{SK}$ , the two distributions  $D_1$  and  $D_2$  of the tuples  $(PK_1, PK_2, \mu_1)$  and  $(PK_1, PK_2, \mu_2)$ , are indistinguishable under the DDH assumption, where these distributions are induced, respectively, by

$$(SK_i, PK_i) = \text{GenerateKey}(k),$$

$$d_i = \text{RegisterPublicKey}(PK_i, C_{SK}),$$

$$\text{and } \mu_i = \text{InitiateTag}(PK_i, d_i, m).$$

We prove the claim above via the following intermediate step, in which we show that two distributions  $D_1$  and  $D_3$  of tuples  $(PK_1, PK_2, \mu_1)$  and  $(PK_1, PK_2, \mu_3)$ , with

$$\mu_3 = \text{InitiateTag}(PK_3, d_3, m),$$

are similarly indistinguishable under the DDH assumption.

Let  $D'_1$  be a distribution of DDH tuples  $(g, g^x, g^y, g^{xy})$ , obtained from  $g \in_R G_1$ , and  $x, y \in_R \mathbf{F}_p$  and  $D'_2$  be the distribution of random tuples from  $G_1^4$ . Now, let  $(u, v, w, z)$  be chosen from either  $D'_1$  or  $D'_2$ , and let

$$\begin{aligned} \mu &= (a_1, a_2, a_3, a_4, a_5, c_1, c_2) \leftarrow \\ &\leftarrow (w^r, w^{rt}, w^{rs}z^{rst}, z^r, z^{rt}, w^{\hat{r}}, z^{\hat{r}}m), \end{aligned}$$

and

$$PK_1 = (u, v), PK_2 = (u^{r'}, u^{r''}),$$

with  $r, r', r'', \hat{r}$  randomly chosen from  $\mathbf{F}_p$  and for some  $s, t \in \mathbf{F}_p$ . Obviously, if  $(u, v, w, z)$  is chosen according to  $D'_1$  or  $D'_2$ , then  $(PK_1, PK_2, \mu)$  is distributed according to  $D_1$  or  $D_3$ , respectively. Since  $D'_1$  and  $D'_2$  are indistinguishable under the DDH assumption, it follows that so are  $D_1$  and  $D_3$ . With a similar argument, one can show that  $D_2$  and  $D_3$  are indistinguishable under the DDH assumption. Consequently,  $D_1$  and  $D_2$  are indistinguishable and hence our scheme provides key privacy.  $\square$

## 7.1 Security Proof

We are going to show that there exist no (real world) adversary  $\mathcal{A}$  who can trace tags in a system of issuers and randomizers that apply our insubvertible encryption scheme. That is, we are going to show that no matter how the adversary  $\mathcal{A}$  behaves, he cannot trace a tag unless he brakes one of the properties of our insubvertible encryption scheme listed above.

To this end, we construct a ideal adversary (simulator)  $\mathcal{S}$  that is given black box access to  $\mathcal{A}$  and interacts with  $\mathcal{T}$ . We will then show that the inputs/outputs of the parties in the real and the ideal world are identically distributed. Thus,  $\mathcal{A}$  cannot trace tags as  $\mathcal{S}$  is not able to do so by definition of our idealized functionality.

THEOREM 7.2. *Under the DDH and the strong LRSW assumptions in  $G_1$  and assuming that  $\text{MAC}(\cdot, \cdot)$  is a secure message authentication scheme, one can securely implement a system of untraceable tags with insubvertible encryption.*

PROOF. We show that for every real world adversary  $\mathcal{A}$ , there exists an ideal adversary  $\mathcal{S}$  such that, assuming key-privacy, unforgeability of key registration, and semantic security of our insubvertible encryption scheme, the environment cannot distinguish whether it interacts with the ideal or the real world parties. In other world we show that our cryptographic implementation of an untraceable tags system behaves exactly as the ideal specification provided in Section 5.2.

**Ideal World Adversary.** The ideal world adversary  $\mathcal{S}$  behaves as follows. First, it runs the protocols `GenerateCAKey`, `GenerateKey` and `RegisterPublicKey` playing the roles of the certification authority and the issuer. It also generates key for the MAC scheme we use.

As we assume that the adversary can only corrupt randomizers,  $\mathcal{S}$  interact with  $\mathcal{T}$  only as (dishonest) randomizer. That is, whenever some dishonest  $R_j$  is to read a device, the ideal world adversary  $\mathcal{S}$  receives from  $\mathcal{T}$  a message containing either an index  $d$  or a string  $\gamma$ .

In the latter case,  $\mathcal{S}$  forwards  $\gamma$  to  $\mathcal{A}$ . In the former case we can have two cases depending on whether  $\mathcal{S}$  has stored  $d$  in its database.

**Case I.**  $\mathcal{S}$  has stored  $d$  in its database. In this case, it looks up the encryption it had stored with  $d$  and forwards it to  $\mathcal{A}$ .

**Case II.**  $\mathcal{S}$  has not stored  $d$  in its database. Thus  $\mathcal{S}$  MAC's  $d$  obtaining  $\sigma$ , encrypts  $(d, \sigma)$  with our scheme obtaining  $e$ , sends the result to the adversary, and stores  $(d, e)$  in its database.

Now  $\mathcal{S}$  waits for the adversary's reply  $\gamma$ . If  $\gamma$  is not a syntactically valid mark,  $\mathcal{S}$  sends  $(5, \gamma)$  to  $\mathcal{T}$ . Assume that  $\gamma$  was valid. So  $\mathcal{S}$  decrypts it.

**Case I.** The decryption results in a MAC'ed message  $d$ .  $\mathcal{S}$  checks whether it has stored  $\gamma$  already. If this is the case  $\mathcal{S}$  sends  $(1, d)$  to  $\mathcal{T}$ . Otherwise, it sends  $(2, d)$  to  $\mathcal{T}$ , receives  $\tilde{d}$  from  $\mathcal{T}$  and stores  $\gamma$  along with  $\tilde{d}$ .

**Case II.** Decryption is garbage.  $\mathcal{S}$  checks whether it has stored  $\gamma$  already. If this is the case,  $\mathcal{S}$  looks up  $d$  stored with  $\gamma$  and sends  $(3, d)$  to  $\mathcal{T}$ . Otherwise, it sends  $(4, d)$  to  $\mathcal{T}$ , receives  $\tilde{d}$  from  $\mathcal{T}$  and stores  $\gamma$  with  $\tilde{d}$ .

Notice that that  $\mathcal{T}$  may reject  $\mathcal{S}$ 's answers in the cases above. However, this will happen only when  $\mathcal{S}$  sends to  $\mathcal{T}$  an index  $d$  that  $\mathcal{T}$  has not generated. In this case,  $\mathcal{S}$  halts.

**Indistinguishability of Real and Ideal World.** Next, it remains to argue that the simulator *works*, i.e., that the environment cannot distinguish whether it interacts with the ideal or the real world parties. It is enough to show that the interaction of the ideal world adversary (simulator)  $\mathcal{S}$  with the real world adversary is indistinguishable from the one of the honest parties in the real world. However, having not the same input information as the honest parties, the ideal world adversary has to behave differently in the following points:

- $\mathcal{S}$  uses only one certified public key whereas, in the real world, each issuer  $\mathcal{D}_i$  uses his own certified public key. However, if the real world adversary could distinguish

between  $\mathcal{S}$  and  $\mathcal{D}_i$ , then we could use the adversary to break the *key privacy* of our insubvertible encryption construction.

- $\mathcal{S}$  encrypts the random identifier  $d$  whereas, in the real world, only the message specified by the environment gets encrypted. Because of the semantic security of our insubvertible encryption scheme, the real world adversary  $\mathcal{A}$  cannot distinguish between these two cases. If he does, then we can use the adversary to break the semantic security of our scheme.
- After a tag has been randomized by an honest randomizer, the encryption that  $\mathcal{S}$  provides to  $\mathcal{A}$  contains a (MAC'ed) identifier  $d$ . In the real world, however, the message encrypted and stored into the tag will not change after a randomization performed by a honest randomizer. In addition, notice that the plaintext is either a valid and authenticated message (specified by the environment), some random garbage created by the adversary, or some string encrypted under some key chosen by the adversary. In the simulation, however, it will always be a random identifier  $d$  encrypted under a single public key chosen by  $\mathcal{S}$ . Since  $\mathcal{A}$  cannot register his own public key, we claim that  $\mathcal{A}$  cannot distinguish between the two worlds (ideal or real) thanks to the fact that our insubvertible encryption provides both key privacy and semantic security.
- Finally, we need to consider the case when  $\mathcal{S}$  fails because it provided  $\mathcal{T}$  with a  $d$  that  $\mathcal{T}$  did not generate. This can only happen if  $\mathcal{A}$  sends  $\mathcal{S}$  a valid mark that contains a MAC'ed  $d$  but that  $\mathcal{S}$  has not MAC'ed itself. However, assuming the security of the MAC-scheme we employ, this cannot happen.

To formally prove that the above statements are true, one would have to transform  $\mathcal{A}$  into an adversary that breaks the security properties of our insubvertible encryption scheme. However, such transformations are standard and therefore omitted here for space limitations.

□

## 8. APPLICATIONS

In this section, we discuss the potential applicability of insubvertible encryption. While most suggested uses are RFID related, we also discuss its applications to mixnets. Indeed, insubvertible encryption is a versatile construction that can be used in other contexts apart from RFID privacy—namely, whenever there is distributed exchange of some form of encrypted tokens one may legitimately pose the question of whether user's privacy may be violated by unnoticed side-channels being exploited by malicious parties.

### 8.1 RFID privacy issues

Several retail store chains, most notably Wal-mart, are in the planning or implementation stages of using RFID as enhanced barcodes in their supply chain. Mostly, the motivation for the technological change has been improvements in warehousing and inventory management. However, specially as the RFIDs presence increases to the granularity of

individual consumer items,<sup>5</sup> concerns of privacy are raised. One proposed solution is to terminate tags on purchased items before they leave the store. However, that limits the ability to provide further customized services to the consumer.

For instance, it has been proposed that, if refrigerators were equipped with (relatively low-cost) RFID-reading technology, they could notify a user when the milk is about to expire, or even keep a running list of items stored—and prepare grocer's lists automatically.

More broadly, RFIDs are an integral, low-cost component of the convergence of technologies which is often called ubiquitous computing, and which has the potential to substantially change the way our environment—at home and at work—interacts with us. Business will seek to tailor their services to individual consumer needs, automatically, which requires identification technologies. These trends conflict with current suggestions to disable RFIDs at the point-of-sale as a measure to protect privacy, and therefore techniques such as insubvertible encryption may prove valuable.

We now discuss an obvious difficulty with the use of RFIDs that do not implement access control features, and how to overcome them in the context of insubvertible encryption.

## 8.2 Tolerating Cloning

The RFID scheme we propose has strong privacy properties. However, as it requires that the tag be many-writable, it is vulnerable to cloning. In some applications, this cloning is mostly a problem before the product has left the manufacturer's chain and entered the consumer realm (when then privacy issue becomes more relevant).

One could deploy two RFIDs on each item—or a dual-core single tag. The first tag would store an immutable ID, and it may only be de-activated. The second tag would store encrypted data via insubvertible encryption. As the item leaves the store, the first tag is de-activated and the other is left untouched, providing full consumer privacy post-sales while supporting a level of pre-sale cloning-resistance. Clearly, a user can suppress the signal of the immutable tag using a sticker of foil, and attach a different immutable RFID tag, at the cost of creating physical evidence of tampering which is visible upon inspection. In any case, to fully prevent cloning much more complex technologies are required, for instance the use of tamper-proof tags that engage in interactive protocols for proof-of-knowledge of secrets. Such technologies are probably too expensive for lower security contexts, such as the envisaged use by mass retailers in the near future. In comparison, dual core tags (one read-only and killable core, the other many-writable) in combination with insubvertible encryption schemes would achieve many of these desirable features while requiring minimum capabilities of the tags. In particular, they would enable post-sale applications of RFIDs with consumer privacy protection.

## 8.3 Other Applications

As we have pointed out, insubvertible encryption is a technology with broader applicability than the RFID application realm. For instance, the universal re-encryption approach to mixnets suffers from the same vulnerability to tracing that we have discussed in the context of RFIDs. In this case, it would permit traffic analysis by one of the mixnet nodes (or

<sup>5</sup>Gillette is going to add RFIDs to individually packed items, for instance.

any router in between two mixnet nodes). If the encryption is modified to hide an encryption under the adversary's public key, then traffic analysis becomes feasible.

## 9. CONCLUSIONS

The development of universally re-encryptable schemes has opened the door to several developments in the privacy area, including applications to mixnets, RFIDs, and other contexts where keyless nodes perform anonymizing functions on behalf of other parties. However, without attention to possible introduction of hidden communication channels, the promise of universal re-encryption is not fully realized.

In this paper we introduce the concept of insubvertible encryption, that provides protection against such attacks, by preventing unauthorized parties from introducing spurious tracing elements into legitimate data. Insubvertible encryption, specially in its application to RFIDs, introduces minimal requirements for implementation, supporting even fully passive tags, and provides strong protection against tracing attacks.

*Acknowledgments.* We are grateful to Ari Juels for enlightening discussions on universal re-encryption and RFID privacy. We thank Matt Green and Alex Maestretti for their insightful remarks on the applicability of insubvertible encryption.

Part of J. Camenisch's work reported in this paper is supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT and by the IST Project PRIME. The PRIME project receives research funding from the European Community's Sixth Framework Programme and the Swiss Federal Office for Education and Science. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

## 10. REFERENCES

- [1] G. Avoine. Privacy issues in RFID banknote protection schemes. In *The Sixth International Conference on Smart Card Research and Advanced Applications – CARDIS*, pp. 33–48, 2004. IFIP, Kluwer Academic Publishers.
- [2] G. Avoine. Security and privacy in rfid systems. <http://lasecwww.epfl.ch/~gavoine/rfid/>, Last Access: May 2005.
- [3] G. Avoine and P. Oechslin. RFID traceability: A multilayer problem. In *Financial Cryptography – FC'05*, LNCS of Springer-Verlag. 2005.
- [4] L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage. Johns Hopkins University, Computer Science Department Technical Report # TR-SP-BGMM-050705, <http://spar.isi.jhu.edu/~mgreen/correlation.pdf>, 2005.
- [5] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Proc. of Security in Communication Networks (SCN'02)*, number 2576 in LNCS, pp. 263–273. Springer-Verlag, 2002.
- [6] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Technical Report

- 2005/133, International Association for Cryptologic Research, 2005.
- [7] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology — ASIACRYPT 2001*, vol. 2248 of *LNCS*, pp. 566–582. Springer Verlag, 2001.
- [8] D. Boneh, X. Boyean, and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO '04*, vol. 3152 of *LNCS*, pp. 41–55, 2004. Full version available at <http://crypto.stanford.edu/~dabo/papers/groupsigs.pdf>.
- [9] D. Boneh and M. Franklin. Identity-based encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
- [10] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil Pairing. *Journal of Cryptology*, 17(4):297–319, 2004.
- [11] D. Boneh, H. Shacham, and B. Lynn. Short signatures from the Weil pairing. In *Advances in Cryptology – ASIACRYPT '01*, vol. 2248 of *LNCS*, pp. 514–532, 2001.
- [12] J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Advances in Cryptology — CRYPTO 2004*. Springer Verlag, 2004.
- [13] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Technical Report 2000/067, Cryptology ePrint Archive, International Association for Cryptology, <http://eprint.iacr.org/2000/067>, 2000.
- [14] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of Foundations of Computer Science (FOCS '01)*, pp. 136–145, 2001.
- [15] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the tate pairing. In *Algorithmic Number Theory Symposium*, vol. 2369 of *LNCS*, pp. 324–337, 2002.
- [16] S. D. Galbraith and V. Rotger. Easy decision Diffie-Hellman groups. *Journal of Computation and Mathematics*, 7:201–218, 2004.
- [17] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In *Proc. of the 2004 RSA Conference*, 2004.
- [18] D. Henrici and P. Müller. Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In *Workshop on Pervasive Computing and Communications Security – PerSec 2004*, pp. 149–153, IEEE Computer Society.
- [19] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In *ANTS-IV conference*, vol. 1838 of *LNCS*, pp. 385–394, 2000.
- [20] A. Juels and R. Pappu. Squealing euros: Privacy protection in RFID-enabled banknotes. In *Financial Cryptography – FC'03*, vol. 2742 of *LNCS*, pp. 103–121, Springer-Verlag, 2003.
- [21] A. Juels, R. Rivest, and M. Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy. In *Conference on Computer and Communications Security – ACM CCS*, pp. 103–111, Washington, DC, USA, October 2003. ACM, ACM Press.
- [22] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Proc. of the 6th Annual International Workshop on Selected Areas in Cryptography (SAC '99)*, vol. 1758 of *LNCS*, pp. 184–199, 1999.
- [23] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curves for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
- [24] D. Molnar and D. Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In *Conference on Computer and Communications Security – ACM CCS*, pp. 210–219, Washington, DC, USA, October 2004. ACM, ACM Press.
- [25] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55:165–172, 1994.
- [26] D. Page, N. Smart, and F. Vercauteren. A comparison of MNT curves and supersingular curves. Cryptology ePrint Archive: Report 2004/165, 2004. <http://eprint.iacr.org/2004/165/>.
- [27] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. of Computer and Communications Security (ACM CCS 2000)*, pp. 245–254. ACM Press, 2000.
- [28] J. Saito, J.-C. Ryou, and K. Sakurai. Enhancing privacy of universal re-encryption scheme for RFID tags. In *Embedded and Ubiquitous Computing – EUC 2004*, vol. 3207 of *LNCS*, pp. 879–890, Springer-Verlag, 2004.
- [29] M. Scott. MIRACL library. Indigo Software. <http://indigo.ie/~mscott/#download>.
- [30] M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Technical Report 2002/164, International Association for Cryptologic Research, 2002.
- [31] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology: Proceedings of Eurocrypt'97*, *LNCS*, pp. 256–266. Springer-Verlag, 1997. Revised version: <http://www.shoup.net/papers/>.
- [32] S. Spiekermann and O. Berthold. Maintaining privacy in RFID enabled environments – proposal for a disable-model. In *The First Workshop on Security and Privacy, Conference on Pervasive Computing*, Vienna, Austria, April 2004.
- [33] E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *Advances in Cryptology — EUROCRYPT '01*, vol. 2045 of *LNCS*, pp. 195–210. Springer-Verlag, 2001.
- [34] E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology*, 17:277–296, 2004.
- [35] 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*, vol. 2802 of *LNCS*, pp. 454–469, Springer-Verlag, 2003.