RFID-Based Supply Chain Partner Authentication and Key Agreement

Florian Kerschbaum
SAP Research
Karlsruhe, Germany
florian.kerschbaum@sap.com

Alessandro Sorniotti
SAP Research and Institut Eurécom
Sophia Antipolis, France
alessandro.sorniotti@sap.com

ABSTRACT

The growing use of RFID in supply chains brings along an indisputable added value from the business perspective, but raises a number of new interesting security challenges. One of them is the authentication of two participants of the supply chain that have possessed the same tagged item, but that have otherwise never communicated before. The situation is even more complex if we imagine that participants to the supply chain may be business competitors. We present a novel cryptographic scheme that solves this problem. In our solution, users exchange tags over the cycle of a supply chain and, if two entities have possessed the same tag, they agree on a secret common key they can use to protect their exchange of business sensitive information. No rogue user can be successful in a malicious authentication, because it would either be traceable or it would imply the loss of a secret key, which provides a strong incentive to keep the tag authentication information secret and protects the integrity of the supply chain. We provide game-based security proofs of our claims, without relying on the random oracle model.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Cryptographic controls*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*

General Terms

Algorithms, Security

Keywords

RFID, Supply Chain, Proof of Possession, Authentication, Key Agreement

1. INTRODUCTION

Radio Frequency Identification (RFID) is a modern technology that supports tracking and tracing of tagged items in supply chains. Each item is equipped with an RFID tag that carries a unique identifier. This tag can be read via radio frequency communication and multiple tags can be read at once.

There are active and passive RFID tags. Active RFID tags have their own power supply while passive tags solely operate on the power of the signal emitted by the reader. The reader is a special device that can interoperate with the tags and read the identifiers stored in their memory. More complex and powerful tags can store information in memory and even perform simple cryptographic operations such as hashing.

A major application of RFID is supply chain management [26, 7, 1]. In the supply chain each item can now be tracked using its unique identifier. The benefit of this tracking and tracing technology unleashes its true potential, when the supply chain partner share their event data. An event happens when a tag is read. At its most basic level this generates a tuple

$$\langle organization, identifier, timestamp \rangle$$

This tuple is usually augmented with additional information, such as reader identifier, type of event (e.g. receiving, shipping, unpacking, etc.), and additional fields depending on the type of event.

Companies are interested in sharing information linked to events for many reasons: firstly, a consumer may be interested in knowing the steps that the product she purchased has gone through. A company may need to recall particular flawed products and is interested in knowing the list of retailers that have actually sold them.

In order to share the data linked with events, companies connect to a global network, currently being standardized by the EPCglobal consortium. This network contains a discovery service, which stores all companies that have event data for a specific tag. In order to retrieve all information about a tag, one contacts the discovery service with its request which then returns the list of all companies to contact. Then one can contact each company individually and ask to retrieve its event data.

The main challenge with this system, is that on the one hand companies have an incentive to share this information so as to facilitate their business, on the other, this information is highly confidential and (possibly competing) companies are reluctant to trust one another. Therefore, a big concern is the possibility of espionage of a competitor's supply chain [24], carried out for instance by retrieving the event data about items in a competitor's supply chain.

Imagine two companies – which might have never communicated before – that contact each other with the help of the discovery service and need to mutually authenticate: the only thing they have ever had in common is that they have both been in possession of the same tag at some point. These companies need to prove to each other that they have possessed the same tag.

There are a number of attacks that might happen in this scenario:

1. An impostor might request information about tags he has never possessed, for example in order to track the supply chain of his competitor.

2. A malicious company might supply rogue information about tags he had never possessed, for instance so as to hide the origin of counterfeited products.

1.1 A Simple Solution

A simple solution to this problem is to store a shared secret on the tag, so that everyone who possessed that tag knows it and can use it to secure subsequent communications.

Unfortunately, this solution has many disadvantages. First, there is no incentive for someone who possessed the item not to divulge the shared secret, since this action cannot be traced back to him. Second, the tag could be maliciously read by an outsider who does not legitimately belong to the supply chain.

Therefore, in order to have a secure solution to the problem at hand, we need to develop a more complex scheme than a simple shared secret.

1.2 Our contribution

In this paper we present a novel scheme implemented in two protocols that solve the aforementioned problem. The intuition is that the information stored on the tag is tied to a secret key or identity (since one of our two solutions is ID-based). Only the holder of the trapdoor information (that identity's private key or a secret value) can actually prove possession of the tag. The information stored on the tag is updated as the tag changes possession; the update is performed with a special re-encryption key or through the help of a trusted third party (TTP). The involvement of the TTP makes it possible to trace the item throughout the supply chain.

Our solution overcomes the disadvantages of the simple solution. If someone illegitimately requests information for a tag, he can be exposed by the TTP. On the other hand, impersonation is possible only if a party's private trapdoor is exposed. A supply chain partner that wants to let another party authenticate, must then decide to either be traceable or to reveal his secret key and relinquish all the business sensitive information to him.

A good property of our solution, is that RFID can be effectively used together with complex cryptographic primitives: indeed, tags just act as carrier of cryptographic envelopes, that are then used off-the-tag to perform complex security protocols. Our technology already works with the simplest tags specified by the EPCglobal standard (class 1 tags). The sole requirement is that tags must be able to store the minimum amount of information required to perform the cryptographic operations. Tags do not necessarily need to be rewritable: one could simply throw away the old tag and put a new one: the ever decreasing price of hardware can totally support this approach. Obviously rewriting the information on the tag is the more elegant solution, and we will use it throughout the remaining description in the paper.

The remainder of the paper is structured as follows. The next section describes the related work. Section 3 introduces some cryptographic background; in Section 4 we give a rapid overview of the different algorithms that compose our scheme, which we thoroughly present in Section 5. Security proofs of both schemes are given in Section 6, before we compare our two schemes in Section 7. The last section presents the conclusions.

2. RELATED WORK

The security of RFID-based systems has been object of intense research over the past few years, given the many threats related to the adoption of this technology [18]. Many papers have focused on privacy-related issues [15, 22, 20]. RFID authentication protocols have received a lot of attention as well [28]. However, to the best of our knowledge, ours is among the first works that address secure interaction among participants of an RFID-enhanced supply chain. An interesting key distribution application for RFID using advanced cryptography has been presented in [19]. Its main advantage is the use of aggregate packaging along the supply chain while maintaining user's privacy.

In the first scheme that we propose, the re-encryption part was inspired by proxy re-encryption and proxy re-signatures. They were introduced in [8] and later improved in many papers, e.g. by [3, 4, 12, 16]. Note that we are using a different setup than proxy re-encryption. In our case the server (TTP) holds all (private) keys, but the clients (users) perform the re-encryption. We require to be able to apply re-signatures multiple times and not just once (as in e.g. in [3]). We inherit some disadvantages, such as bidirectionality.

The signature or authentication part of our first scheme borrows from [11]. In that work, short signatures are formed using bilinear maps and one secret key. Our first scheme is very similar in that it ties the public key to the secret key chosen for re-encryption. Therefore the authenticator has to present both a signature of the challenge and the tag authentication information.

The key agreement part of our first scheme is inspired by traditional secured Diffie-Hellman key exchange [14]. It is not to be mistaken with the tripartite version of [17]. It enhances two-party Diffie-Hellman to work in groups with bilinear maps.

The second scheme that we propose borrows interesting ideas from Secret Handshakes. Secret Handshakes are first introduced in 2003 by Balfanz et al. [5] as mechanisms designed to prove group membership, and share a secret key, between two fellow group members. Additional properties are that non-members must not be able to either impersonate group members or to recognize legitimate group members. In [2], Ateniese et al. present the first Secret Handshake protocol that allows for matching of properties different from the user's own. Our construction of tag information is similar to Ateniese et al.'s credentials. However, in secret handshakes, the loss of credentials has tragic consequences on the security of the scheme, whereas in our dynamic scenario, we have to account for the possibility that tag information is read by somebody different from the intended recipient. Despite this, the scheme must still be secure: this is probably the most remarkable distinction of our scheme from secret handshakes.

3. PRELIMINARIES

3.1 Cryptographic Pairings

Given a security parameter k, let $(\mathbb{G}_1, *)$ and $(\mathbb{G}_2, *)$ be two groups of order p for some large prime p, where the bit-size of p is determined by the security parameter k. Our scheme uses a computable, non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ for which the *Computational Diffie-Hellman Problem (CDH)* problem is assumed to be hard. In what follows, we denote $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$.

Modified Weil or Tate pairings on supersingular elliptic curves are examples of such maps. We recall that a bilinear pairing satisfies the following three properties:

- Bilinear: for $g, h \in \mathbb{G}_1$ and for $a, b \in \mathbb{Z}_p^*$, $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$
- Non-degenerate: $\hat{e}(g, g) \neq 1$ is a generator of $\mathbb{G}_2$
- Computable: there exists an efficient algorithm to compute $\hat{e}(g, h)$ for all $g, h \in \mathbb{G}_1$

3.2 Identity-Based Encryption

Identity-based encryption (IBE) was introduced in [25] as an alternative to public-key encryption. In IBE any string can be used as an encryption key, e.g. one can encrypt an e-mail using the recipient's e-mail address. The recipient then obtains the decryption key (for his identity) from a trusted third party after successful authentication.

The procedures of an IBE scheme are

- **Setup**: The TTP publishes public parameters.
- **Encrypt**: One transforms a plaintext using an arbitrary string as key into a ciphertext.
- **Get Decryption Key**: One obtains a decryption key for an identity from the TTP.
- **Decrypt**: One transforms the ciphertext using the decryption key into its plaintext again.

The first practical IBE scheme was presented in [10]. It is based on cryptographic pairings described in Section 3.1 and is proved IND-ID-CCA secure under the bilinear decisional Diffie-Hellman assumption in the random oracle model.

In our scheme every party has an identity; following the Alice and Bob convention, we refer to a message encrypted for Alice as $IBE_A(m)$. We will treat identity based encryption as a building block and will build our scheme independently on top of it.

3.3 Hash functions

In this section we describe the hash functions that are used in one of the two presented protocols. Our construction leverages on the work proposed by Boneh and Boyen in [9] and later improved by Waters in [27].

Let $g \xleftarrow{R} \mathbb{G}_1$; let us also choose $n+1$ random values $u_0, u_1, \ldots, u_n \xleftarrow{R} \mathbb{Z}_p^*$; we assign $U_0 = g^{u_0}, U_1 = g^{u_1}, \ldots, U_n = g^{u_n}$. If $v \in \{0,1\}^n$ is an n-bit string, let us define $h(v) = u_0 + \sum_{i \in V} u_i \in \mathbb{Z}_p^*$, where $V \subseteq \{1, \ldots, n\}$ is the set of indexes i for which the ith bit of v is equal to 1. We also define $H(v) = U_0 \prod_{i \in V} U_i = g^{h(v)} \in \mathbb{G}_1$.

With such an approach, we can represent in $\mathbb{G}_1$ strings of size n, or alternatively, strings of arbitrary length, preprocessed with a hash function whose block size is n.

We will use this approach in the second protocol that we present, whereas in the first, we will use standard hash functions.

4. SUPPLY CHAIN PARTNER AUTHENTICATION

Assume Trent is a trusted third party that supports users in updating the information stored on the tag as the tag changes of possession. Then our supply chain partner authentication consists of the following algorithms or protocols.

Setup: Trent publishes some public parameters about the system known to every participant.

Register: A new company Alice wants to join the supply chain and is contacting Trent to register. They setup public/private/secret information tied to Alice's identity. Trent returns to Alice her public and private information, keeping the secret one for himself.

Initialize: Alice just created a new item and attached a tag to it. She creates the secret information on the tag. She does so without Trent's intervention.

Ship: Alice intends to ship the item to Bob and contacts Trent seeking for his support. Trent either delivers the re-encryption key that can be used to update the tag information for its next owner, or updates the information himself.

Receive: Bob receives the item and stores its information, so that he can later authenticate request for event data.

Authenticate: Alice and Bob want to mutually authenticate as having owned the same tag. They exchange random challenges to salt the instance of the protocol, and mutually verify whether the other has owned the same tag. Afterward, they share a key which they use to protect subsequent communications.

Using the proposed approach, we could envisage the following scenario. The production of a complex good needs the cooperation of different agents. This process often involves different companies that take part to the supply chain. For instance three different companies A, B and C may cooperate as follows: company A has an item and - according to its usual business - needs to ship it along to another company for further processing. The "next" company is not known in advance and company A chooses company B (but could easily have chosen company B'). A then performs the shipping operation invoking the ship algorithm. Similarly, B ships the item down to company C. Eventually the chain stops.

At a later point in time, company A and company C may need to interact on the basis of the ownership of the tagged item described above. Notice that A and C have never interacted before, and may not have any pre-established business relationship whatsoever. Company A and company C have kept in a database the association of the tag ID with the cryptographic information stored within the tag at the moment of its receipt. They use this information to perform a handshake that, if successful, allows them to safely rely on one another as business partners with respect to that tag, and to share a key used to secure further communication.

5. SOLUTIONS

In this section we present two solutions to the aforementioned problem. We describe the implementation of the algorithms and protocols that we introduced in the last section.

5.1 RFIDAuth1

In this section we introduce RFIDAuth1, a first protocol based on bilinear pairings. RFIDAuth1 consists of the following algorithms:

- Setup

  According to the security parameter k, Trent chooses $(p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, \hat{e})$, where g and $\tilde{g}$ are random generators of $\mathbb{G}_1$. He picks $\alpha \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and sets $S = g^\alpha$. The system's public parameters are $\mathsf{params} = \{p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, S, \hat{e}\}$. The value α is instead kept in Trent's secret storage.

- Register

  Alice wants to register with Trent to enter the supply chain partner network. She chooses two random elements $y_A, z_A \overset{R}{\leftarrow} \mathbb{Z}_p$. She sends $\tilde{g}^{y_A}$ and g^{z_A} to Trent. Trent chooses a random element x_A from $\mathbb{Z}_p^*$ and returns to Alice $(g^{x_A}\tilde{g}^{y_A})^{\alpha^{-1}}$ and g^{x_A}. These and all subsequent message exchanges with Trent need to be conducted over secure and authenticated channels.

  Notice that there are three different types of information related to Alice: Alice's public information, represented by the pair $(g^{z_A}, (g^{x_A}\tilde{g}^{y_A})^{\alpha^{-1}})$; Trent gives this pair to anybody interested in dealing with Alice (similarly to public key certificates). Alice's private information is represented by the tuple g^{x_A}, y_A, z_A. Alice's secret information, known only to Trent is the value x_A.

A $\longrightarrow$ T	$\tilde{g}^{y_A}$, g^{z_A}
T $\longrightarrow$ A	g^{x_A}, $(g^{x_A}\tilde{g}^{y_A})^{\alpha^{-1}}$

Figure 1: Registration Protocol

- Initialize

  Alice wants to initialize a new tag. She chooses a random element $t_{tag} \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes $X_1 = g^{t_{tag}}$ and $X_2 = (g^{x_A})^{t_{tag}^{-1}}$. Recall that Trent sent g^{x_A} to Alice as her hidden part of the re-encryption key during the register protocol. Alice stores the pair (X_1, X_2) on the tag and should destroy t_{tag} immediately.

- Ship

  Alice intends to ship the item with the tag to Bob. Alice contacts Trent and indicates her intention by sending A and B. Trent retrieves x_A for Alice and x_B for Bob and returns $k_{A,B} = x_A^{-1}x_B \ mod\ p-1$ to Alice. Alice computes

$$X_2' = X_2^{k_{A,B}} = ((g^{x_A})^{t_{tag}^{-1}})^{x_A^{-1}x_B} = (g^{x_B})^{t_{tag}^{-1}}$$

  She stores the pair (X_1, X_2') on the tag (and erases the old one).

A $\longrightarrow$ T	A, B
T $\longrightarrow$ A	$x_A^{-1}x_B \ mod\ p-1$

Figure 2: Ship Protocol

- Receive

  Bob receives a tag and stores the pair (X_1, X_2) in his database. If he ships the item further, he applies the ship protocol.

- Authenticate

  Let us assume that Alice and Charlie want to authenticate as having possessed a common tag, and in case of mutual successful check, want to share a key. One of the two indicates so by sending the tag identifier to the other. Let us assume that Charlie starts. Mind that Alice has $(X_{1A} = g^{t_{tag}}, X_{2A} = g^{x_A t_{tag}^{-1}})$ and Charlie has $(X_{1C} = g^{t_{tag}}, X_{2C} = g^{x_C t_{tag}^{-1}})$. Alice contacts Trent and retrieves Charlie's public information g^{z_C}, and $(g^{x_C}\tilde{g}^{y_C})^{\alpha^{-1}}$. She then chooses a random element $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and sends g^r as a random challenge to Charlie. Charlie uses his secret key y_C to compute $(g^r)^{y_C}$ and retrieves X_{2C} from his database. He sends both values to Alice.

  Alice retrieves X_{1A} from her database and checks whether

$$\frac{\hat{e}(X_{1A}, X_{2C})^r \hat{e}(g^{y_C r}, \tilde{g})}{\hat{e}(S, (g^{x_C}\tilde{g}^{y_C})^{\alpha^{-1}})^r} = \frac{\hat{e}(g^{t_{tag}}, g^{t_{tag}^{-1}x_C})^r \hat{e}(g^{y_C r}, \tilde{g})}{\hat{e}(g^\alpha, (g^{x_C}\tilde{g}^{y_C})^{\alpha^{-1}})^r} = 1$$

  holds. The same check is performed by Charlie, who queries Trent for Alice's public information, sends a random challenge g^s to Alice and receives in response $g^{y_A s}$, X_{2A}. Then he checks if

$$\frac{\hat{e}(X_{1C}, X_{2A})^s \hat{e}(g^{y_A s}, \tilde{g})}{\hat{e}(S, (g^{x_A}\tilde{g}^{y_A})^{\alpha^{-1}})^s} = \frac{\hat{e}(g^{t_{tag}}, g^{t_{tag}^{-1}x_A})^s \hat{e}(g^{y_A s}, \tilde{g})}{\hat{e}(g^\alpha, (g^{x_A}\tilde{g}^{y_A})^{\alpha^{-1}})^s} = 1$$

  holds. If the check is successful at both sides, both users are certain that they have possessed the tag and continue with the key agreement.

  In order to establish a shared secret, upon a successful mutual proof of possession, they set the secret key K to

$$\begin{aligned} K = &\quad \hat{e}(g^{z_C}, g^s)^{z_A r} \\ = &\quad \hat{e}(g, g)^{r z_A s z_C} \\ = &\quad \hat{e}(g^{z_A}, g^r)^{z_C s} \end{aligned}$$

  Subsequent communications between A and C are securely protected through the use of the secretly shared key K[1].

[1]Notice that no eavesdropper can reconstruct the key from

C $\longrightarrow$ A	*id*
A $\longleftrightarrow$ T	public information protocol
C $\longleftrightarrow$ T	public information protocol
A $\longrightarrow$ C	g^r
C $\longrightarrow$ A	$g^s, g^{y_C r}$, X_{2C}
A $\longrightarrow$ C	$g^{y_A s}$, X_{2A}
A $\longleftrightarrow$ C	data exchange protected by K

Figure 3: Authentication Protocol

A $\longrightarrow$ T	C
T $\longrightarrow$ A	g^{z_C}, $(g^{x_C}\tilde{g}^{y_C})^{\alpha^{-1}}$

Figure 4: Public Information Protocol

Note that the public information protocol can be replaced by a certificate signed by Trent. Then there is no need for any communication with Trent during the authentication protocol.

5.1.1 *Stronger Tracing*

There are some limitations on tracing when Trent hands out re-encryption keys. Alice does not have to obtain a re-encryption key for every tag, but only once per partner she is sending items to. This reduces the burden on Trent, but also reduces traceability.

In this situation Trent can build a graph of who can send to whom, but not a graph per item of who has sent to whom. The resulting overlay graph of all items might be to close to complete in order to reveal any useful tracing information.

Furthermore, there are some cryptographic limits on the re-encryption key. Given the re-encryption key $k_{A,B} = x_A^{-1}$ $x_B \ mod \ p-1$ one can compute the re-encryption key $k_{B,A} = k_{A,B}^{-1} = x_B^{-1} x_A \ mod \ p-1$. Given the re-encryption keys $k_{A,B}$ and $k_{B,C}$ one can compute the re-encryption key $k_{A,C} = k_{A,B} k_{B,C}$. To counter these attacks, Trent would have to include the reverse of each edge and compute the transitive closure of the graph.

An alternative is to involve Trent in every Ship operation. The *Get Re-encryption Key* operation therefore ceases to exist and the *Ship* operation is modified as shown in Figure 5. Assume Alice wants to send a tagged item to Bob. Alice has read $X_2 = (g^{x_A})^{t_{tag}^{-1}}$ from the item in a previous receive operation or produced it in an initialize operation. Alice sends T_A and Bob's identity to Trent which responds with $X_2' = ((g^{x_A})^{t_{tag}^{-1}})^{x_A^{-1} x_B} = (g^{x_B})^{t_{tag}^{-1}}$.

A $\longrightarrow$ T	B, $X_2 = (g^{x_A})^{t_{tag}^{-1}}$
T $\longrightarrow$ A	$X_2' = (g^{x_B})^{t_{tag}^{-1}}$

Figure 5: Strong Tracing Ship Protocol

Trent can compute $g^{t_{tag}^{-1}} = X_2^{x_A^{-1}}$ and store the triple $\langle g^{t_{tag}^{-1}}, A, B\rangle$ in his database. The identifier $g^{t_{tag}^{-1}}$ is unique per tag and is never changed, such that Trent can build a complete forwarding pedigree of the tag. Trent can then clearly identify any party divulging authentication information if an impostor for a tag is identified.

the information on the wire, because no probabilistic polynomial time algorithm can reconstruct $\hat{e}(g,g)^{r z_A s z_C}$ from g^r, g^s, g^{z_A} and g^{z_C}.

5.2 RFIDAuth2

In this section, we enrich the previous scheme, enabling the use of identities. A user's public information is simply the hash of his identity. This way we effectively remove the need for certificates issued by Trent. The *Public Information Protocol* is therefore no longer needed. We put ourselves in the stronger tracing scenario, introduced in Section 5.1.1, and we thus require Trent's support upon each execution of the Ship algorithm.

Another advancement with respect to the previous scheme is that we develop a neater authentication and key derivation scheme, borrowing from the field of Secret Handshakes. In this scheme we adopt the hashing approach outlined in Section 3.3.

RFIDAuth2 consists of the following algorithms:

- Setup

  According to the security parameter k, Trent chooses $(p, \mathbb{G}_1, \mathbb{G}_2, g, \hat{e})$, where g is a random generator of $\mathbb{G}_1$. He also picks $u_0, u_1, \ldots, u_n \xleftarrow{R} \mathbb{Z}_p^*$ and assigns $U_0 = g^{u_0}, U_1 = g^{u_1}, \ldots, U_n = g^{u_n}$. Finally, he picks $\alpha \xleftarrow{R} \mathbb{Z}_p^*$ and sets $S = g^{\alpha}$ and $S' = g^{\alpha^{-1}}$. The system's public parameters are $\mathsf{params} = \{p, \mathbb{G}_1, \mathbb{G}_2, g, S, S', \hat{e}, U_0, \ldots, U_n\}$. The values $u_0, u_1, \ldots, u_n$ and α are instead kept in Trent's secret storage.

  Trent finally initializes an IBE scheme (see Section 3.2 for further details), and distributes its public parameters to every user in the system.

- Register

  Alice wants to register with Trent to enter the supply chain partner network. She just needs to prove her identity to Trent, and then she receives from Trent the secret key associated to her identity. In addition, she receives the value $I_A = H(A)^{\alpha}$ which she stores secretly.

- Initialize

  Alice wants to initialize a new tag. She chooses a random value $t_{tag} \xleftarrow{R} \mathbb{Z}_p^*$ and she computes $X_1 = S^{t_{tag}} \, {I_A}^r$, $X_2 = g^r$ and $X_3 = H(A)^r$. Alice stores the tuple $(X1, X2, X3)$ on the tag and should destroy t_{tag} immediately. Notice that the Initialize phase does not require the intervention of Trent.

- Ship

  Alice intends to ship the item with the tag to Bob. Alice contacts Trent and indicates her intention by sending the tag's ID, A, B and the pair $(X1, X2, X3)$, read from the tag upon reception. We remind the reader that $X_1 = S^{t_{tag}} \, {I_A}^r$, $X_2 = g^r$ and $X_3 = H(A)^r$. Trent checks whether $\hat{e}(X_3, g) = \hat{e}(H(A), X_2)$, thus checking if the tuple really corresponds to the identity of Alice. Trent in turn picks $s \xleftarrow{R} \mathbb{Z}_p^*$ and computes

$$\begin{cases} X_1' = \dfrac{X_1}{X_3^{\alpha}} \, {I_B}^s = S^{t_{tag}} \, {I_B}^s \\ X_2' = g^s \\ X_3' = H(B)^s \end{cases}$$

Then, Trent sends X'_1, X'_2, X'_3 to Alice. Upon receipt, Alice stores the tuple on the tag (and erases the old one).

For tracking purposes, Trent can compute $S^{t_{tag}} = \dfrac{X_1}{X_3^{\alpha}}$ and store the triple $\langle S^{t_{tag}}, A, B\rangle$ in his database. The identifier $S^{t_{tag}}$ is unique per tag and is never changed, such that the TTP can build a complete forwarding pedigree of the tag. Trent can then clearly identify any party divulging authentication information if an impostor for a tag is identified.

A $\longrightarrow$ T	tag ID, A, B, $S^{t_{tag}}$ $I_A{}^r$ and $H(A)^r$
T $\longrightarrow$ A	$S^{t_{tag}}$ $I_B{}^s$, g^s and $H(B)^s$

Figure 6: Ship Protocol

- Receive

  Bob receives a tag and stores the tuple (X_1, X_2, X_3) in his database, associating it to the ID of the tag. Bob checks whether $\hat{e}(X_3, g) = \hat{e}(H(B), X_2)$, to verify if the received tuple was indeed destined to his identity. If he ships the item further, he applies the ship protocol.

- Authenticate

  Let us assume Bob and Alice want to authenticate, proving to one another that they have possessed a given tag. Let us assume that Bob initiates the handshake by sending the tag identifier to Alice. Notice that both Alice and Bob possess the values (X_1, X_2, X_3) read from the tag upon Receive; let us add the subscript A (resp B) to identify Alice's (resp. Bob's) tuple.

  Alice picks a random nonce $n_A \in \mathbb{Z}_p^*$, and then computes $IBE_B(H(B)^{n_A}, (S')^{n_A})$ and sends it to Bob. Similarly Bob picks a random $n_B \in \mathbb{Z}_p^*$, computes $IBE_A(H(A)^{n_B}, (S')^{n_B})$ and sends it back to Alice.

  If both Alice and Bob have taken part to the supply chain for the product identified by the tag, they can derive a common shared key,

$$\begin{aligned} K = & \left(\frac{\hat{e}(X_{1A}, (S')^{n_B})}{\hat{e}(H(A)^{n_B}, X_{2A})}\right)^{n_A} \\ = & \left(\frac{\hat{e}(S^{t_{tag}}\ I_A{}^r, (S')^{n_B})}{\hat{e}(H(A)^{n_B}, g^r)}\right)^{n_A} \\ = & \ \hat{e}(g, \tilde{g})^{t_{tag} n_A n_B} \\ = & \left(\frac{\hat{e}(S^{t_{tag}}\ I_B{}^s, (S')^{n_A})}{\hat{e}(H(B)^{n_A}, g^s)}\right)^{n_B} \\ = & \left(\frac{\hat{e}(X_{1B}, (S')^{n_A})}{\hat{e}(H(B)^{n_A}, X_{2B})}\right)^{n_B} \end{aligned}$$

  thus proving to each other that they have legitimately handled the tag. In order to seal the handshake, they can use any challenge-response protocol known in the literature in order to prove to one another knowledge of the shared key without leaking it.

  Subsequent communications are protected using K to setup a secure channel.

B $\longrightarrow$ A	ID
A $\longrightarrow$ B	$IBE_B(H(B)^{n_A}, (S')^{n_A})$
B $\longrightarrow$ A	$IBE_A(H(A)^{n_B}, (S')^{n_B})$
A $\longleftrightarrow$ B	challenge-response based on K
A $\longleftrightarrow$ B	data exchange protected with K

Figure 7: Handshake

6. SECURITY ANALYSIS

In this section we analyze the security of both presented schemes. Being an authentication and key agreement scheme, we intuitively need to prove that no attacker can fool a legitimate user into thinking that *he is* somebody and that *he has owned* a given tag. We approach our investigation about the security of the scheme using game-based proofs, which allow us to study in a single proof many different attacks. In the security proofs we do *not* rely on the random oracle model [6].

We consider active attacks, but do not consider framing attacks where two parties collude to provide false information to an intermediate party, such that the intermediary cannot authenticate. First we believe that this collusion is difficult, since the intermediary is free to choose the next party and second the tracing information would reveal a backward flow even in the weak tracing scheme.

We therefore present two games for the first scheme, proving that an attacker cannot ship items at his will and that an attacker cannot authenticate as another user. For the second scheme we design a single, slightly broader game where we prove that an attacker is not able to impersonate any other user: this includes protection from forgery, illegitimate shipping and attacks to the authentication and key agreement scheme.

In our scheme every ship event needs the involvement of the TTP and the security proofs confirm that this is unavoidable. Our scheme allows therefore complete tracing of the shipping events in the strong tracing version of the first protocol and in the second protocol. Trent can thus built an entire shipping graph for each item. No participant outside of that graph can perform a successful handshake.

This fact has an interesting consequence: if an unauthorized party is successful in an illegitimate handshake, he could be blamed, and the TTP could also trace which participant leaked the information. Therefore there is a strong incentive not to disclose the information on the tag on purpose and the entire supply chain is tightly controlled.

6.1 Assumptions

In this Section we state well-known hard problems upon which the security of our scheme is based.

Definition 1. *The Computational Diffie-Hellman Problem (CDH) is hard if, for all probabilistic, polynomial-time algorithms B,*

$$\mathsf{AdvCDH}_B := Pr[B(g, g^a, g^b) = g^{ab}]$$

is negligible in the security parameter.

This probability is taken over random choice of $g \in \mathbb{G}_1$, $a, b \in \mathbb{Z}_q^*$. This assumption is one of the most renowned in the cryptographic community, and was introduced in [14].

Definition 2. *The modified Computational Diffie-Hellman Problem (mCDH) is hard if, for all probabilistic, polynomial-time algorithms B,*

$$\mathsf{AdvmCDH}_B := Pr[B(g, g^a, g^b, g^{b^{-1}}) = g^{ab}]$$

is negligible in the security parameter.

This probability is taken over random choice of $g \in \mathbb{G}_1$, $a, b \in \mathbb{Z}_q^*$. This assumption, although not standard, has been used in a number of publications [23, 21].

Definition 3. *The Bilinear Decisional Diffie-Hellman Problem (BDDH) is hard if, for all probabilistic, polynomial-time algorithms B,*

$$\mathsf{AdvBDDH}_B := Pr[B(g, g^a, g^b, g^c, g^x) = \top \text{ if } x = abc] - \tfrac{1}{2}$$

is negligible in the security parameter.

This probability is taken over random choice of $g \in \mathbb{G}_1$, $a, b, c, x \in \mathbb{Z}_q^*$. This complexity assumptions is well known in the cryptographic community, and has been used in the proofs of many cryptographic schemes, for instance [13].

6.2 Security of RFIDAuth1

Let us consider the security of the RFIDAuth1 protocol. There are two types of attacks: First, an attacker could try to create a tuple (X_1, X_2) for another user without ever having obtained a re-encryption key for that user, or without Trent's support in the case of strong traceability. This corresponds to actively leaking the secret information on the tag eluding TTP's traceability. We introduce the game Reencrypt to capture this attack. We show that is hard to win this game without knowledge of Trent's secret information x.

Second, an attacker could steal or otherwise obtain a tuple (X_1, X_2) for another user and then try to authenticate as that user. This corresponds to getting hold of a tag and then trying to authenticate as its legitimate owner. We introduce the game Authenticate to capture this attack. We show that is hard to win this game without knowledge of that user's secret information y.

The combination of these two games shows that the attacker has no option to successfully authenticate without the sender being traceable or someone revealing its secret keys. On the one hand the sender must involve Trent in shipping the item and is therefore traceable. On the other hand if he reveals his secret keys, he gives away completely his authentication capabilities to an attacker.

There is a third game for an observer of the key agreement to infer the secret key. It can be trivially shown that this hard without knowledge of any user's secret information z. Since the challenge of the key agreement is tied to the proof of possession, an active party cannot intercept.

6.2.1 Re-Encryption Resistance

Consider an adversary A that has as its goal to perform the ship protocol without the support of Trent. A is allowed to freely perform all the algorithms of the protocol. Then A picks two users $I_\circ$ and I_* of his choice; the simulator B Initializes a challenge tag as $I_\circ$ and supplies all the relevant information about $I_\circ$ and I_* to A, except the values $K_{I_*,\cdot}$ and $K_{\cdot,I_*}$ and the secret information related to I_*. Eventually, B submits to the attacker the pair $(X_{1I_\circ}, X_{2I_\circ})$ and A outputs his guess for the information X_{2I_*}. We call this game Reencrypt.

Theorem 1. *If an adversary A has a non-null advantage*

$$\mathsf{Reencrypt}_A := Pr[A \text{ wins the game } \mathsf{Reencrypt}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve a given instance of the modified Computational Diffie-Hellman Problem (mCDH).

Proof. We define B as follows. B is given a random instance $(g, g^a, g^b, g^{b^{-1}})$ of the mCDH problem and wishes to use A to compute g^{ab}. The algorithm B simulates an environment in which A operates.

Setup The simulator B picks and publishes the public parameters according to the rules of the protocol.

Queries The attacker can Register at his will as any identity I he chooses. A can Initialize any tag as any user of his choice. We remind the reader that he can perform this operation autonomously without the involvement of the simulator. The Ship protocol is executed as mandated in the protocol specification, therefore A is free to ask B to perform the ship protocol on any tag he has received or on any tag he has initialized. Then, the attacker can engage in authentication protocols with every user of his choice: in this case, B creates all the simulated parties I (except I_*) by selecting x_I, y_I, and z_I thus knowing all the secret information. Finally, A can perform the receive protocol, declaring a target user I and thus receiving $(X_1 = g^{t_{tag}}, X_2 = g^{x_I t_{tag}^{-1}})$ from B, where $t_{tag} \xleftarrow{R} \mathbb{Z}_p^*$.

Challenge The attacker A then chooses an identity $I_\circ$, for which B has already answered all his queries in the previous phase, and I_* such that he does not know $K_{I_*,\cdot}$ and $K_{\cdot,I_*}$ and the secret information y_{I_*} and z_{I_*}. A asks for the public information about I_*; B answers with $g^{z_{I_*}}$, $(g^a \tilde{g}^{y_{I_*}})^{\alpha^{-1}}$. Finally, a can receive tags destined to I_*; to do so, B picks $t_{tag} \xleftarrow{R} \mathbb{Z}_p^*$ and sends to A the pair $X_1 = g^{t_{tag}}$ and $X_2 = g^{t_{tag}^{-1} a}$. Eventually B sends A the information linked to the tag object of the challenge, crafted as follows: $X_{1I_\circ} = g^{b^{-1}}$ and $X_{2I_\circ} = (g^b)^{x_{I_\circ}}$ and A outputs its guess for X_{2I_*}.

Analysis of A's response If A has won the game, $X_{2I_*} = g^{ab}$ and B can give the same answer to the received instance of mCDH. □

6.2.2 Authentication Resistance

Consider an adversary A that has as its goal to perform the authentication protocol as a user without owning the secret material for the latter, in particular the secret values y and $z \in \mathbb{Z}_p^*$, only known by the user. This game shows that a user is protected in case of theft of on-tag credentials (the pair (X_1, X_2)) which is always possible using rogue readers. A is allowed to freely perform all the algorithms of the protocol (as user A, of course). Then A picks a user I_* of his choice; A receives as well all tags destined for I_*. Eventually, A engages in the authentication protocol, producing the values that should convince the simulator that he is I_* and has possessed the item. We call this game Authenticate. Note that this game also rules out a user intentionally leaking the on-tag credentials to a third party.

THEOREM 2. *If an adversary A has a non-null advantage*

$$\mathsf{Authenticate}_A := Pr[A \textit{ wins the game } \mathsf{Authenticate}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve a given instance of the Computational Diffie-Hellman Problem (CDH).

PROOF. We define B as follows. B is given a random instance (g, g^a, g^b) of the CDH problem and wishes to use A to compute g^{ab}. The algorithm B simulates an environment in which A operates.

Setup The simulator B picks $g \xleftarrow{R} \mathbb{G}_1$, $\beta \xleftarrow{R} \mathbb{Z}_p^*$ and sets $\tilde{g} \leftarrow g^\beta$ and publishes the public parameters according to the rules of the protocol.

Queries The attacker can Register at his will as any identity I he chooses. A can Initialize any tag as any user of his choice. We remind the reader that he can perform this operation autonomously without the involvement of the simulator. The Ship protocol is executed as mandated in the protocol specification, therefore A is free to ask B to perform the ship protocol on any tag he has received or on any tag he has initialized. Then, the attacker can engage in authentication protocols with every user of his choice: in this case, B creates all the simulated parties I (except I_*) by selecting x_I, y_I, and z_I thus knowing all the secret information. Finally, A can perform the receive protocol, declaring a target user I and thus receiving $(X_1 = g^{t_{tag}}, X_2 = g^{x_i t_{tag}^{-1}})$ from B, where $t_{tag} \xleftarrow{R} \mathbb{Z}_p^*$.

Challenge The attacker A then chooses the identity I_* he wishes to authenticate as, amongst the ones not queried before. A receives I_*'s public information $g^{z_{I_*}}$ and $(g^{x_{I_*}} g^{a\beta})^{\alpha^{-1}}$. A can receive tag information destined to I_*: B picks $t^{tag} \xleftarrow{R} \mathbb{Z}_p^*$ and sends A $(X_1 = g^{t_{tag}}, X_2 = g^{x_{I_*} t_{tag}^{-1}})$. To trigger the challenge, A sends B the tag ID of one of the tags received as I_*. Now, B answers with a random challenge g^b. A must then answer – according to the protocol – with $(g^b)^{y_{I_*}}$ and X_{2I_*}.

Analysis of A's response If A has won the game, $(g^b)^{y_{I_*}}$ $= g^{ab}$ and B can give the same answer to the received instance of CDH. $\square$

6.3 Security of RFIDAuth2

In this section we investigate the security of the RFIDAuth2 protocol. Let us first of all analyze a simple attack and show how the scheme prevents it. A user could eavesdrop the communications that occur upon the Ship protocol, or just simply get hold of a tag and extract the tuple (X_1, X_2, X_3), and try to use that tuple to engage in a successful handshake with a legitimate participant. This is in short what is described in Section 6.2.2 for the protocol RFIDAuth1.

The latter – as we have seen – has a built-in protection against this type of attack, namely the secrecy of values x and $y \in \mathbb{Z}_p^*$. RFIDAuth2 instead leverages on Identity Based Encryption to prevent this type of attack: it is clear that – since handshake challenges are destined to a user and encrypted under the public key of his identity – mere eavesdropping of tag information will not help to break the secrecy of the challenge sent, and therefore every attack of this kind is vain. We do not need to prove this, as we assume the existence of a perfect IBE scheme that does not leak any information.

Nonetheless, in order to show that the scheme is sound, we present in the next Section the security proof of resistance to impersonation, wherein we "switch off" IBE (or similarly, we give all the private keys to the attacker). What we prove, in short, is that with all the information in the hands of the adversary *but* the one associated to a challenge tag and a challenge user, the adversary is not able to impersonate the latter. This game is broad enough as to include privacy of the key exchange from an eavesdropper, collusion of several participants, forgery of rogue tag information and so forth.

6.3.1 *Impersonation Resistance*

Consider an adversary A that has as its goal to run a successful handshake – thus convincing another user that he has actually owned a tag – without disposing of the legitimate information. In particular, A does not have the tuple $(X_{1v_*}, X_{2v_*}, X_{3v_*})$ for a given user v_* and a given tag, both object of the challenge.

A is allowed to freely perform all the algorithms of the protocol. Then the simulator B Initializes a challenge tag, and yet the adversary is able to get the information to perform a successful handshake for that tag as any user of his choice (except the one object of the challenge).

Finally, the attacker picks a challenge user v_* and is required to run a successful handshake, convincing the simulator that he is user v_* having owned the challenge tag. In particular, at the end of the game, the attacker is required to output the key K. We call this game Impersonate.

THEOREM 3. *If an adversary A has a non-null advantage*

$$\mathsf{Impersonate}_A := Pr[A \textit{ wins the game } \mathsf{Impersonate}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve a given instance of the Bilinear Decisional Diffie-Hellman Problem (BDDH).

PROOF. We define B as follows. B is given a random instance (g, g^a, g^b, g^c, g^x) of the BDDH problem and wishes to use A to check whether $x = abc$. The algorithm B simulates an environment in which A operates.

Setup The simulator B sets an integer $m = 4q$ where q is an upper bound on the number of identities that the adversary will consider throughout his queries to the various protocols. B then chooses $k \xleftarrow{R} \{0, n\}$ and chooses two random vectors $X = \{x_i\}_{i=1}^n \xleftarrow{R} \{0, m-1\}^n$ and $Y = \{y_i\}_{i=1}^n \xleftarrow{R} \mathbb{Z}_p^{*n}$. Following Boneh and Boyen [9] and Waters [27], we define the functions $F(v) = (p - mk) + x_0 + \sum_{i \in V} x_i$, $J(v) = y_0 + \sum_{i \in V} y_i$ and $K(v)$ as

$$K(v) = \begin{cases} 0, & \text{if } x_0 + \sum_{i \in V} x_i = 0 \; mod \; m; \\ 1, & \text{otherwise}; \end{cases}$$

The simulator sets g as the one received from the BDH challenge, $U_0 = (g^b)^{p-km+x_0} g^{y_0}$ and $U_i = (g^b)^{x_i} g^{y_i}$;

he then picks $\alpha \overset{R}{\leftarrow} \mathbb{Z}_p^*$, sets $S = g^\alpha$ and $S' = g^{\alpha^{-1}}$ and publishes the public parameters according to the rules of the protocol. Notice that now, $H(v) = U_0 \prod_{i \in V} U_i = g^{bF(v)+J(v)}$, where V is the set of indexes i for which the ith bit of the string at hand equals to 1.

Queries First of all, the attacker receives *all* IBE private keys: this way, the protection of IBE is disabled. In the rest of this proof therefore, we omit the notation $IBE.(\cdot)$.

The attacker can Register at his will as any identity v_i he chooses, different from v_*, receiving from Trent the value I_{v_i}.

A can Initialize any tag as any user of his choice. We remind the reader that he can perform this operation autonomously without the involvement of the simulator.

Upon execution of the Ship protocol, the attacker A sends to B the ID of a tag, two identities v_i and v_j and the tuple $(X_1 = S^{t_{tag}} \, {I_{v_i}}^r,\ X_2 = g^r, X_3 = H(v_i)^r)$. B computes

$$\begin{cases} X_1' = \dfrac{X_1}{X_3^\alpha} \, {I_{v_j}}^s = S^{t_{tag}} \, {I_{v_j}}^s \\ X_2' = g^s \\ X_3' = H(v_j)^s \end{cases}$$

as mandated by the Ship protocol, and sends the tuple (X_1', X_2', X_3') back to A.

Finally, A can perform the receive protocol by simply storing the received tuple and associating it to the tag id.

B then Initializes a new tag, which will be the object of the challenge. A is then entitled to receive – for any user v_i of his choice – the information necessary to run a successful handshake as that user. A therefore sends v_i to B. If $K(v_i) = 0$, B aborts and outputs a random guess. If not, B picks a $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes

$$\begin{cases} X_1 & = \left((g^a)^{\frac{-J(v_i)}{F(v_i)}} \left((g^b)^{F(v_i)} g^{J(v)} \right)^r \right)^\alpha \\ & = g^{\alpha ab} \left(g^{bF(v_i)+j(v_i)} \right)^{\alpha \tilde{r}} \\ & = S^{ab} \, {I_{v_i}}^{\tilde{r}} \\ X_2 & = (g^a)^{\frac{-1}{F(v_i)}} g^r = g^{\tilde{r}} \end{cases}$$

where $\tilde{r} = r - a/F(v_i)$. With the pair (X_1, X_2), the attacker can perform any handshake he wants, but cannot perform the ship protocol.

In addition, given the pair (X_1, X_2) for two identities v_i and v_j, the attacker can check – through the execution of a handshake protocol – whether the credentials received where indeed linked to the queried identities. Therefore, the simulation offered by B to A is perfect.

Challenge The attacker A then chooses an identity v_* he has not queried before; if $x_0 + \sum_{i \in V} x_i \neq km$ the simulator aborts and submits a random guess. Otherwise we have $F(v_*) = 0 \; mod \; p$, which means that $H(V_*) = g^{J(v_*)}$. B then sends as challenge the pair $(H(v_*)^c = (g^c)^{J(v_*)}, S'^c)$ according to the description of the handshake protocol. A answers with $(H(v_i)^r, S'^r)$, and then outputs the key K.

Analysis of A's response If A has won the game, $K = \hat{e}(g,g)^{abcr}$. Therefore, B can solve the BDDH problem by checking whether $\hat{e}\left(g^x, (S'^r)^{\alpha^{-1}}\right) = K$ holds. A detailed analysis of the probability that B does not need to abort has been presented in [27] and we therefore omit it here. $\square$

7. COMPARSION

We have given two schemes implementing our RFID-basd supply chain partner authentication. In this section we want to give some guidelines on when to choose which. The first set of algorithms RFIDAuth1 allows for weak tracing of RFID items. In weak tracing no interaction (when using certificates) with the trusted third party is required besides initially obtaining the re-encryption keys. The parties can therefore act autonomously and RFIDAuth1 should be used when weak authentication is sufficient.

The protocols RFIDAuth2 offers a security that relies on weaker assumptions, but only offers strong tracing. In strong tracing the shipper needs to contact the trusted third party for every shipment, but the trusted third can now track every item. Furthermore RFIDAuth2 uses identities rather than certificates, enabling easier key management. Therefore RFIDAuth2 should be used, if strong authentication is required.

8. CONCLUSION

In this paper, we have presented a novel scheme to replace shared secrets when forwarding items tagged with RFID. Our scheme discourages disclosure of authentication information by tying it to a secret key or identity. Either one discloses the secret key or forwards the information according to the protocol specification, but if one does so, he is traceable by a trusted third party.

We have presented two protocols implementing this scheme: a key-based and an identity-based one. The key-based protocol allows the trusted third party to hand out re-encryption keys, while the identity-based protocol reduces the number of interactions.

We proved both protocols secure in a game-based security definition based on common security assumptions. Our scheme can be applied even to the simplest tags if the information is sent along over the network.

Our scheme presents a novel approach to the problem that reaches beyond current security applications in securing the integrity of supply chains. We anticipate that, due to its simplicity in application and strong security guarantees, our scheme has wide applications in securing RFID-supported supply chains. Future work is to incorporate its security into the query answer of the discovery service.

9. ACKNOWLEDGEMENTS

This work has partially been financed by the European Commission through the ICT programme under Framework 7 grant FP7-213531 to the SecureSCM project, under the Seventh Framework Programme IST Project "Trusted Architecture for Securely Shared Services" (TAS^3), grant agreement number 216287, and under the Sixth Framework Programme IST Project "Wirelessly Accessible Sensor Populations" (WASP), contract number IST-034963.

10. REFERENCES

[1] A. Asif and M. Mandviwalla. Integrating the supply chain with rfid: A technical and business analysis. In *Communications of the Association for Information Systems, vol. 15*, pages 393–427, 2005.

[2] G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *Network and Distributed System Security Symposuim*, pages 159–177. The Internet Society, 02 2007. CERIAS TR 2007-24.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1), 2006.

[4] G. Ateniese and S. Hohenberger. Proxy re-signatures: new definitions, algorithms, and applications. In *ACM Conference on Computer and Communications Security*, 2005.

[5] D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H.-C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196, 2003.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[7] Y. Bendavid, S. F. Wamba, and L. A. Lefebvre. Proof of concept of an rfid-enabled supply chain in a b2b e-commerce environment. In *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pages 564–568, New York, NY, USA, 2006. ACM.

[8] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, 1998.

[9] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[10] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[11] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4), 2004.

[12] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *ACM Conference on Computer and Communications Security*, 2007.

[13] H. Chabanne, D. H. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT*, pages 542–558, 2005.

[14] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, Nov 1976.

[15] S. Garfinkel, A. Juels, and R. Pappu. Rfid privacy: an overview of problems and proposed solutions. *Security & Privacy, IEEE*, 3(3):34–43, May-June 2005.

[16] M. Green and G. Ateniese. Identity-based proxy re-encryption. In *Conference on Applied Cryptography and Network Security*, 2007.

[17] A. Joux. A one round protocol for tripartite diffie-hellman. *Journal of Cryptology*, 17(4), 2004.

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

[19] A. Juels, R. Pappu, and B. Parno. Unidirectional key distribution across time and space with applications to rfid security. In *USENIX Security Symposium*, 2008.

[20] A. Juels and S. A. Weis. Defining strong privacy for rfid. *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 342–347, March 2007.

[21] S. Lal and P. Kushwah. Multi-pkg id based signcryption. Cryptology ePrint Archive, Report 2008/050, 2008.

[22] H. Lee and J. Kim. Privacy threats and issues in mobile rfid. *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pages 5 pp.–, April 2006.

[23] B. Libert and D. Vergnaud. Multi-use unidirectional proxy re-signatures. *CoRR*, abs/0802.1113, 2008.

[24] B. D. Santos and L. Smith. Rfid in the supply chain: panacea or pandora's box? *Communications of the ACM*, 51(10), 2008.

[25] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[26] S. F. Wamba and H. Boeck. Enhancing information flow in a retail supply chain using rfid and the epc network. *J. Theor. Appl. Electron. Commer. Res.*, 3(1):92–105, 2008.

[27] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[28] Y. Yousuf and V. Potdar. A survey of rfid authentication protocols. *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 1346–1350, March 2008.