Securely Disseminating RFID Events

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

ABSTRACT

More and more companies are collecting data about each item of their supply chain using RFID tags in order to increase visibility across the supply chain and improve its performance. The data generated by reading an RFID tag is (also) called an event. These events are frequently distributed using event-based networks following the publish-subscribe pattern. This method of dissemination immediately raises security concerns, since supply chain operations’ data is considered sensitive by companies.

Attribute-based encryption has proven successful in enforcing access control in publish-subscribe networks. Nevertheless, existing schemes do not support access control for RFID events. In this paper we present an encryption scheme that enables disseminating events that can only be decrypted by a selected set of parties that have been in possession of the RFID tag. Our scheme enables broadcasting event messages of constant ciphertext size to an entire network while enforcing access control policies via encryption. We prove our scheme secure under the Modified Bilinear Decisional Diffie-Hellman Assumption.

Categories and Subject Descriptors
D.4.6 [Operating Systems]: Security and Protection—Cryptographic controls; C.2.4 [Computer-Communication Networks]: Distributed Systems—Distributed databases

General Terms
Algorithms, Design, Security

Keywords
RFID, Supply Chain Management, Item Tracking, Attribute-Based Encryption, Publish-Subscribe Networks, Visibility Policies

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.

DEBS’11, July 11–15, 2011, New York, New York, USA.
Copyright 2011 ACM 978-1-4503-0423-8/11/07 ...$10.00.

1. INTRODUCTION

More and more companies are equipping their products with RFID tags [7]. Each RFID tag carries a unique identifier and allows tracking the item throughout the supply chain [21]. An RFID tag when read creates an event consisting of identifier, location and time.

When these events are shared among the companies of a supply chain they enable a whole new set of beneficial applications, such as anti-counterfeiting [13], targeted recalls [25] or supply chain benchmarking [14]. Nevertheless companies are very reluctant to do so. These events do not only reveal the information necessary to enable the applications, but also reveal additional information about the company’s operation [16, 20]. They may, for example, reveal strategic supplier relationships, planned promotions or best practices.

In order to decrease these concerns, separate entangled supply chains and increase data sharing in visibility policies have been proposed. These reciprocal policies are based on a simple and intuitive concept: Alice reveals to Bob all events about items that Bob also possessed. If Bob does the same the data sharing is fair in an information-theoretic sense, since both parties reveal information about the same set of items. This fairness should enable them to more easily enter into data sharing agreements.

RFID events are frequently distributed in event-based networks following the publish-subscribe pattern. Publish-subscribe networks [2, 17] are a novel data distribution mechanism. Publishers multicast events which have been enlisted by subscribers. These events – the same term has been coined by the RFID community – are a superset of the RFID events. Usually a publisher does not know all subscribers, since the routing of events is performed by the network [6].

Now, enforcing access control policies has become notoriously difficult, since there is no more policy enforcement point. Fortunately encryption may help. Particularly, attribute-based encryption (ABE) [1, 8, 10, 18] enables encrypting events, such that they can only be decrypted by parties that possess the corresponding attributes. This nicely corresponds to attribute-based access control (ABAC) [27] which allows access based on attributes of subject, object or environment. A clever adaptation to publish-subscribe networks has already been made in [24].

This would be excellent news, but ABE can only enforce subject attributes, i.e. attributes of the party accessing the data. In [11] it has already been noted that visibility policies rely on a combined subject-object attribute. The object is the data or item identifier on the RFID tag and a party may only access the event, if this party (subject) has been
in possession of this item (object). Therefore existing ABE cannot be straightforwardly applied to this problem.

In this paper we present an encryption scheme that enables a party to decrypt only if the party has received a (forwarded) message before and an authorization message. The forwarded message can, for example, be stored on the RFID tag and ensures that the party has been in possession of the item. The authorization message ensures, that the private key holder has enabled this party to decrypt any message it has received a forwarded message for.

Given this encryption scheme a sender in the publish-subscribe network can encrypt his events and be ensured that

a) they can be only read by parties that also had the item
b) they can be only read by parties that he explicitly authorized

These two conditions necessitate two proofs of security: one for the case a party has not received the item and one for the case a party has not been authorized. For both cases we prove ciphertext indistinguishability against a chosen plaintext attack under the (Modified) Decisional Diffie-Hellman Assumption.

In summary this paper contributes

- an attribute-based encryption scheme for enforcing visibility policies in publish-subscribe networks. To the best of our knowledge, this is the first ABE scheme that also involves object-related attributes.
- an IND-CPA style proof in case a party has not been authorized
- an IND-CPA style proof in case a party has not been forwarded the corresponding message

The remainder of the paper is structured as follows. In the next section, we review the visibility policy model. In Section 3 we describe the trivial construction and oppose the advantageous of our scheme. We also give the intuition behind our construction. In Section 4 we describe our encryption scheme in detail and in Section 5 we give its two security proofs. We review related work in Section 6, before we conclude the paper in Section 7.

2. BACKGROUND: VISIBILITY POLICIES

Visibility policies are an extension of attribute-based access control (ABAC). In ABAC the decision whether an access is granted is determined based on the attributes of subject, object and environment [27].

There are sets of attributes for subjects, objects and environment. For each subject, object or environment there is an assignment $\text{ATTR}(s, o, e)$ of a subset of these attributes. A policy rule is a Boolean function of the attributes of subject, object and environment of the request. If the function evaluates to true given the assignment of attributes, access is granted; otherwise it is denied.

Visibility policies do not fit the ABAC model of [27]. In RFID event distribution, subjects are companies and objects are items. A company (subject) is requesting access to data of a specific item (object). This access should be granted if, the item (object) has been in possession of the company (subject). Therefore visibility policies implement a combined subject, object attribute [11].

Let $\text{ATTR}(s, o)$ be the assignment function of combined subject, object attributes for subject $s$ and object $o$. If $o$ has been in possession of $s$, then “$\text{vis}'' \in \text{ATTR}(s, o)$”. One can now write policy rules implementing visibility policies, e.g., granting access to all supply chain partners for their items, but excluding a competitor “Charlie”

$$\text{access}(s, o) = \text{“vis}'' \in \text{ATTR}(s, o) \land \text{“Charlie}'' \notin \text{ATTR}(s)$$

3. TRIVIAL SOLUTION

There is a trivial solution to our problem described above. Simply store on each RFID tag the attribute token in the ABE scheme (or even more simple a password or symmetric key) that enables accessing events for this item. Every party that receives the RFID tag will store the token and can then access events.

Nevertheless this is a bad idea, because the token needs to be harshly safeguarded. It needs to be encrypted on the RFID tag to evade rogue readers and it needs to be stored securely in order to prevent theft. Even worse, the token is not traceable, i.e. if it is leaked, it cannot be determined by whom. As a consequence most parties may not be inclined to safeguard the password or even deliberately reveal it to outsiders.

Our construction avoids this problem by tying the information on the tag to its recipient. The stored information is always unique and can only be used in combination with private information by the recipient. We even include a special operation to authorize forwarding. Without this authorization a party must reveal its private information in order to enable decryption. This implements a strong form of traceability.

The intuition of our scheme is simple. We store a symmetric key on the RFID tag, but this key is not stored in plain. Instead it is encrypted by a key of the recipient. Nevertheless the recipient does not hold this key; it is held by a trusted third party. This trusted third party – we call her Trent – holds a key for each party and in order for Alice to forward Bob Trent has to issue Alice a proxy re-encryption key to Bob. Proxy re-encryption has been introduced in [4]. The basic idea is that a proxy can translate ciphertexts under Alice’s key into ciphertexts under Bob’s key without learning the plaintext, i.e. without decrypting. We now what reverse the scheme and make Alice the proxy while the trusted third party emulates all parties.

While this particular trusted third party is obviously not yet available in supply chains, there are a number of suitable candidates: the already successful PKI certificate authorities, the publish-subscribe network software vendor, the RFID standardization organization or even governmental bodies interested in securing IT infrastructure and supply chains.

We entangle the private keys held by Trent with the secret keys held by the parties and the password stored on the RFID tag, such that only all three pieces together are sufficient to decrypt a ciphertext. This entanglement clearly separates our work from previous ABE schemes. Loosely speaking, we present an ABE scheme where one attribute is encrypted and re-encryptable. The combination of both attributes enables enforcing visibility policies with an extension to the most commonly used subset of ABAC as proposed in [11].
4. ENCRYPTION SCHEME

Our encryption scheme consists of the following algorithms and protocols:

- **Setup**($k$) $\rightarrow$ (prv, pbk) is an algorithm that on input of the security parameter $k$ outputs a private key prv and a corresponding public key pbk.

- **Register**($()$, $()$) $\rightarrow$ ((sk$_A$), (ttk$_A$)) is a protocol between Alice and Trent. The protocol has no input and Alice outputs a secret key sk$_A$ and Trent outputs the corresponding trusted key ttk$_A$. We emphasize that in our use of proxy re-encryption the trusted key is not entirely public and contains secret information as well.

- **Trace**($()$, (ttk$_A$, ttk$_B$)) $\rightarrow$ ((fwd$_{AB}$), ())) is a protocol between Alice and Trent. Alice has no input and Trent inputs two trusted keys ttk$_A$ and ttk$_B$. Alice outputs a forwarding key fwd$_{AB}$ and Trent has no output. We emphasize that Trent is now aware that Alice can forward security tokens to Bob.

- **Forward**($k_A$, fwd$_{AB}$) $\rightarrow$ $k_B$ is an algorithm that on input of a security token $k_A$ and a forwarding key fwd$_{AB}$ outputs a security token $k_B$. An initial security token can be created by random choice.

- **Authorize**((sk$_A$), $()$, (prv, ttk$_B$)) $\rightarrow$ (((), (auth$_{AB}$), ())) is a protocol between Alice, Bob and Trent. Alice inputs her secret key sk$_A$, Bob has no input and Trent inputs her private key prv and a trusted key ttk$_B$. Bob outputs an authorization key auth$_{AB}$ and the other parties have no output.

- **Encrypt**($m$, pbk, $k_A$, sk$_A$) $\rightarrow$ $c$ is an algorithm that on input of a message $m$, a public key pbk, a security token $k_A$ and secret key sk$_A$ outputs a ciphertext $c$.

- **Decrypt**($c$, $k_B$, auth$_{AB}$) $\rightarrow$ $m$ is an algorithm that on input of a ciphertext $c$, a security token $k_B$ and an authorization key auth$_{AB}$ outputs a message plaintext $m$.

Let **Authorize**$_B$((sk$_A$), $()$, (prv, ttk$_B$)) denote Bob’s output auth$_{AB}$ of the **Authorize** protocol run on inputs (sk$_A$, $()$, (prv, ttk$_B$)). Let **Trace**$_A$(((), (ttk$_A$, ttk$_B$)) denote Alice’s output fwd$_{AB}$ of the **Trace** protocol run on inputs ((), (ttk$_A$, ttk$_B$)).

Our encryption is consistent if

\[
\begin{align*}
\forall (prv, pbk) & \leftarrow \text{Setup}(k) \\
\forall ((sk_A), (ttk_A)) & \leftarrow \text{Register}((), ()) \\
\forall ((sk_A), (ttk_B)) & \leftarrow \text{Register}((), ()) \\
\forall k_A & \leftarrow \mathcal{G} \\
\forall m & \leftarrow \mathcal{G} \\
\text{Decrypt} & \left( \text{Encrypt}(m, pbk, k_A, sk_A), \\
& \text{Forward}(k_A, \text{Trace}_A(((), (ttk_A, ttk_B))), \\
& \text{Authorize}_B((sk_A), ((), (prv, ttk_B))) \\
& = m \right) 
\end{align*}
\]

In an example scenario we envision the following use of the algorithms and protocols:

1. Trent executes **Setup** and publishes the public key pbk.

2. The parties Alice and Bob, each execute the protocol **Register** with Trent generating their secret keys sk$_A$ and sk$_B$, respectively, and their trusted keys ttk$_A$ and ttk$_B$, respectively, at Trent.

3. Alice executes the protocol **Trace** with Trent outputting forwarding key fwd$_{AB}$.

4. Alice may authorize Bob to receive messages by executing the protocol **Authorize** with him and Trent. Bob will then obtain authorization key auth$_{AB}$.

Note that this step only needs to be performed once at setup time. In corporate publish-subscribe networks it is key to authorize your recipients, since the network may forward to site you do not intend to disclose your data.

5. Alice when preparing for a shipment of items to Bob, executes **Forward** generating security token $k_B$. The input security token $k_A$ may either have been received by Alice with the items or randomly generated if Alice manufactured the items.

6. Alice sends the security $k_B$ to Bob. Steps five and six may be repeated multiple times, once for each item. Items may also be forwarded along the entire supply chain by other parties and in this case the security token is passed through analogous **Forward** algorithms and send operations. The most elegant way to send the security tokens is to store them on the RFID tags, but our scheme is not tied to this way of transport. The security token could also be sent in an accompanying electronic message.

Note that this step is transitive and only needs to be performed once for each item. The recipient may then forward the item to the next person in the supply chain. The ideal moment for this operation is when anyway shipping the item. There is no need to perform this operation per event sent in the publish-subscribe network.

7. Alice when preparing to send an event encrypts the message using the **Encrypt** algorithm.

8. Alice broadcasts the message in the publish-subscribe network. Steps seven and eight may also be repeated multiple times, once for each event.

We emphasize that there is no need for authorization or otherwise communication besides sending the encrypted event. We maintain the communication pattern for publish-subscribe networks and create no additional overhead besides adding a (necessary) one-time authorization layer.

9. Bob – as an eligible receiver – decrypts the ciphertext and obtains the event message.

Parties receiving the item at a later stage can store the events and decrypt them when receiving the item.

Our example scenario is somewhat restricted. It can be easily extended to more than two parties and multi-directional RFID forwarding and event publishing. In fact, in this scenario the use of our encryption scheme may seem strange, since Alice could simply decide to send or not send, but, of
course, our scheme is still able to enforce visibility policies, if Alice and Bob are several hops apart. However, for clarity and brevity we prefer this example scenario. Also, our security games and proofs follow this scenario, although they could be extended in the same way. Figure 1 depicts the interactions. We emphasize that all communication except, of course, the sending of the security token via RFID (step 6) and the publishing of events (step 8) should be done over secure and authenticated channels.

4.1 Security Definition

In order to assess the security of our encryption scheme, we define two games of ciphertext indistinguishability under chosen plaintext attack. Loosely speaking, in the first game WOAUTH-IND-CPA the adversary is excluded from executing the necessary Authorize protocol and in the second game WOFWD-IND-CPA the adversary is excluded from obtaining the necessary, forwarded security token. In each game the adversary operates as a party of the supply chain in a simulated environment. This environment consists of Trent, one more party Alice in the supply chain and infinitely many items.

4.1.1 Game WOAUTH-IND-CPA

Setup: The simulator hands a public key to the adversary. The adversary may register with Trent via Register. The adversary may request a forwarding key to Alice via Trace. And different to the previous game, the adversary may request to be authorized by Alice via Authorize.

Phase I: The adversary may request Alice to generate (and send) security tokens \( k_A \). The adversary may send security tokens \( k_A \) to Alice. The adversary may request Alice to encrypt (and publish) any plaintext \( m \), even corresponding to a security token (item) \( k_A \) or \( k_B \) of his choice. This phase ends at the discretion of the adversary.

Challenge: The simulator chooses a security token \( k_A^* \). The adversary wins the game, if he correctly guesses \( f^* = f \).

Definition 1. The advantage of the adversary in winning game WOAUTH-IND-CPA is \( \text{Adv}^{\text{WOAUTH-IND-CPA}}(1^n) = |Pr[f^* = f] − \frac{1}{2}|. \)

4.1.2 Game WOFWD-IND-CPA

Setup: The simulator hands a public key to the adversary. The adversary may register with Trent via Register. The adversary may request a forwarding key to Alice via Trace. And different to the previous game, the adversary may request to be authorized by Alice via Authorize.

Phase I: The adversary may request Alice to generate (and send) security tokens \( k_B \). The adversary may send security tokens \( k_B \) to Alice. The adversary may request Alice to encrypt (and publish) any plaintext \( m \), even corresponding to a security token (item) \( k_A \) or \( k_B \) of his choice. This phase ends at the discretion of the adversary.

Challenge: The simulator chooses a security token \( k_B^* \). The adversary wins the game, if he correctly guesses \( f^* = f \).

Definition 2. We say that the Bilinear Decisional Diffie Hellman (BDDH) assumption holds, if given values \( g, g^a, g^b, g^c \in \mathbb{G} \) and \( \hat{e}(g, g)^d \in \mathbb{G}_T \) it is not computationally feasible to decide if \( d = abc \).

Furthermore, we also use the Modified Bilinear Decisional Diffie Hellman assumption which includes the additional element \( g^{-1} \). It has been introduced in [5] and clearly implies the BDDH assumption.
Theorem 1. If the BDDH assumption holds, the adversary’s advantage
\[
Adv^{WORTH}(1^k) < \frac{1}{\text{poly}(k)}
\]
is a negligible function of the security parameter k.

Theorem 2. If the MBDDH assumption holds, the adversary’s advantage
\[
Adv^{WOFWD}(1^k) < \frac{1}{\text{poly}(k)}
\]
is a negligible function of the security parameter k.

4.2 Algorithms and Protocols

We now describe how we implement the algorithms and protocols defined above.

Setup: Given the security parameter k, Trent chooses groups \(G\) and \(G_T\) with bilinear map \(e\). Let \(g\) be a generator of \(G\). Trent also uniformly chooses a random number \(\alpha \in \mathbb{Z}_p^*\). The public key is \(g, g^\alpha\). The private key is \(\alpha\).

Register: Alice contacts Trent with her desire to register – as in all protocols of our encryption scheme via a secure and authenticated channel. Trent uniformly chooses the trusted key \(y_A \in \mathbb{Z}_p^*\). Trent sends \(g^{\alpha y_A}\) to Alice. Alice uniformly chooses a random number \(z_A \in \mathbb{Z}_p^*\). Her secret key is \(z_A, g^{\alpha y_A}\).

\[
A \rightarrow T \quad A
\]
\[
T \quad y_A \leftarrow \mathbb{Z}_p^*
\]
\[
T \rightarrow A \quad g^{\alpha y_A}
\]
\[
A \quad z_A \leftarrow \mathbb{Z}_p^*
\]

Trace: Alice contacts Trent with her desire to forward to Bob. Trent looks up the trusted keys \(y_A\) and \(y_B\) for Alice and Bob, respectively. Trent sends the forwarding key \(y_A^{-1} y_B\) to Alice.

\[
A \rightarrow T \quad B
\]
\[
T \rightarrow A \quad y_A^{-1} y_B
\]

In case Alice requests forwarding keys for more than one party, these keys remain uniformly distributed. All keys are perfect secret shares of the same secret \(y_A^{-1}\).

Forward: Given security token \(k^{y_A}\) and forwarding key \(y_A^{-1} y_B\) Alice computes the security token \(k^{y_B}\).

\[
k^{y_B} = (k^{y_A}) y_A^{-1} y_B
\]

She may forward this token to Bob, e.g. by storing it on the RFID tag for the associated item.

Authorize: Alice uniformly chooses a random number \(r \in \mathbb{Z}_p^*\). She contacts Trent with her desire to authorize Bob and sends along with it \(r^{-1}\). She then looks up her secret key \(z_A\) and sends \(s = rz_A\) to Bob. Trent looks up his private key \(\alpha\), Bob’s trusted key \(y_B\) and sends \(t = \alpha^{-1} y_B^{-1} r^{-1}\) to Bob. Bob computes the authorization key as \(\alpha^{-1} y_B^{-1} z_A = st\).

\[
T \quad r \rightarrow \mathbb{Z}_p^*
\]
\[
A \rightarrow T \quad B, r^{-1}
\]
\[
T \rightarrow A \quad t = \alpha^{-1} y_B^{-1} r^{-1}
\]
\[
B \quad \alpha^{-1} y_B^{-1} z_A = st
\]

Encrypt: Alice (or Bob) may be ready to encrypt. She uniformly chooses a random number \(r \in \mathbb{Z}_p^*\). She looks up the public key \(g^\alpha\), the security token \(k^{y_A}\) and her secret key \(z_A, g^{\alpha y_A}\). Given message \(m\) she computes the ciphertext \(c = (C, D)\)

\[
C = g^{ar}, D = m e(k^{y_A}, g^{\alpha y_A^{-1} z_A})
\]

Decrypt: Bob (or Alice) may be ready to decrypt. He looks up the security token \(k^{y_B}\) and the authorization key \(\alpha^{-1} y_B^{-1} z_A\). Given the ciphertext \(c = (C, D)\) he computes

\[
m = D / e(k^{y_A}, C) \alpha^{-1} y_B^{-1} z_A
\]

For consistency observe that

\[
e(k^{y_A}, g^{\alpha y_A^{-1} z_A}) = e(k, g)^{z_A} = e(k, g^\alpha) \alpha^{-1} y_B^{-1} z_A
\]

We will exploit this relation in our security proofs and simulate one side of the equation per proof.

5. SECURITY PROOFS

We prove Theorems 1 and 2 by reduction to the security assumptions. We give a simulator that given an instance of the assumed hard problem simulates the game WOAUTH-IND-CPA or WOFWD-IND-CPA, respectively. The simulation is indistinguishable from the view in a real attack against our encryption scheme, if the given problem instance is a BDDH quadruple or MBDDH quintuple, respectively. If the problem instance is random, our simulation contains no information about the plaintexts. We can then exploit the advantage of an attacker in the simulated game to break the assumed hard problem.

Proof. We proof Theorem 1 by constructing a simulator that given a BDDH problem instance, simulates the game WOAUTH-IND-CPA.

5.1 Game WOAUTH-IND-CPA

Outline: The simulator is given an instance \(g, g^\alpha, g^b, g^c, \hat{e}(g, g)^d\) of the BDDH problem. It will make its random choices, such that \(\log_2 k = a, r = b\) and \(z_A = c\) in the challenge ciphertext. Note that \(k\) and \(r\) are unique to the challenge ciphertext and \(z_A\) is only used in its given form \(g^{\alpha y_A}\).

Setup: The simulator is given an instance \(g, g^\alpha, g^b, g^c, \hat{e}(g, g)^d\) of the BDDH problem. It uniformly chooses a random number \(\alpha \in \mathbb{Z}_p^*\). The public \(g, g^\alpha\) is given to the adversary.

If the adversary queries to be registered, i.e. it invokes the Register protocol with the simulated Trent, the simulator uniformly chooses \(y_B \in \mathbb{Z}_p^*\). It sends \(g^{y_B}\) to the adversary.

If the adversary queries to request a forwarding key to Alice, i.e. it invokes the Trace protocol with the simulated
Trent, the simulator uniformly chooses \( y_A \in \mathbb{Z}_p^* \). It sends \( y_B^{-1} y_A \) to the adversary.

The simulator uniformly chooses \( y_A \) and \( y_B \), respectively, if the adversary does not query. The adversary may not query to be authorized in this game. A query to authorize Alice to receive messages from the adversary produces no output at the adversary.

**Phase I:** If the adversary queries to be sent a security token, the simulator uniformly chooses \( k \in G \). It sends \( k^y_B \) to the adversary.

The adversary may send security token \( k^y_A \) to the simulated Alice.

The adversary queries to encrypt \( m \). If the adversary specifies security token \( k^y_A \), the simulator may use this value unmodified in the subsequent computation. If the adversary specifies security token \( k^y_B \), the simulator computes \( k^y_A = (k^y_B)^y_B^{-1} y_A \). If the adversary does not specify a security token, the simulator uniformly chooses \( k^y_A \in G \). The simulator now uniformly chooses \( r \in \mathbb{Z}_p^* \). It computes

\[
C = g^{\alpha r}, D = m \cdot \varepsilon(k^y_A, (g^r)^y_A^{-1})^r
\]

If the adversary chooses to end this phase, the simulator proceeds.

**Challenge:** The simulator sends the security token \( k_t^y_B \) to the adversary. The adversary chooses two plaintexts \( m_0 \) and \( m_1 \) and gives them to the simulator. The simulator flips a random coin \( f \in \{0, 1\} \). It computes

\[
C^* = (g^b)^y, D^* = m_f^r \cdot \varepsilon(g, g)^d
\]

It sends the challenge ciphertext \( C^* = (C^*, D^*) \) to the adversary.

**Phase II:** The simulator responds to the queries as in phase I. If the adversary queries for an encryption under \( k^y_B \), the simulator uses \( k = g^a \).

If the adversary chooses to end this phase, the simulator proceeds.

**Guess:** The adversary outputs a guess \( f^* \) of \( f \). If \( f^* = f \), the simulator outputs \( d = ac \) and otherwise \( d = r \).

**Claim:** The view of the adversary is as in a real attack, if the problem instance is a BDH quadruple. The simulated view is perfectly indistinguishable. In case \( d = r \), the challenge ciphertext \( c^* \) contains no information about the plaintext \( m_f^r \). It is uniformly distributed. Therefore, if the adversary has advantage \( \varepsilon \) in breaking the game WOAUTH–IND–CPA, the simulator has advantage \( \frac{1}{2} \) in solving the BDH problem.

\( \square \)

**Proof.** We prove Theorem 2 by constructing a simulator that given a MBDDH problem instance, simulates the game WOFWD-IND-CPA.

### 5.2 Game WOFWD-IND-CPA

Outline: The simulator is given an instance \( g, g^a, g^b, g^c \) of the MBDDH problem. It will make its random choices, such that \( \log_k k = a, r = b \) and \( y_B = c \) in the challenge ciphertext. Note that \( k \) and \( r \) are unique to the challenge ciphertext. Nevertheless the simulator for this game is more sophisticated, since \( y_B \) is also used in the Register, Trace and Authorize protocols. We therefore need the additional element \( g^{c-1} \) for the Register protocol.

Due to the additional random elements the output of the other two protocols can be simulated independently. We give the details for the individual phases.

**Setup:** The simulator is given an instance \( g, g^a, g^b, g^c, g^{c-1}, \varepsilon(g, g)^d \) of the MBDDH problem. It uniformly chooses a random number \( \alpha \in \mathbb{Z}_p^* \). The public \( g, g^a \) is given to the adversary.

If the adversary queries to be registered, i.e. it invokes the Register protocol with the simulated Trent, the simulator sends \( g^{c-1} \) to the adversary.

If the adversary queries to request a forwarding key to Alice, i.e. it invokes the Trace protocol with the simulated Trent, the simulator uniformly chooses \( \beta \in \mathbb{Z}_p^* \) and sends it to the adversary. Note that the real element \( y_B^{-1} y_A \) is uniformly distributed in \( \mathbb{Z}_p^* \) as well, since \( y_A \) is uniformly distributed in \( \mathbb{Z}_p^* \). Furthermore, this is the first query answer involving the random element \( y_A \) and therefore it can be chosen independently.

If the adversary queries to be authorized, i.e. it invokes the Authorize protocol with the simulated Alice and Trent, the simulator uniformly chooses two random numbers \( \gamma, \delta \in \mathbb{Z}_p^* \). It sends \( \gamma \) as Alice to the adversary and \( \delta \) as Trent. Note that the real element \( s = rz_A \) is uniformly distributed in \( \mathbb{Z}_p^* \), because \( z_A \) is uniformly distributed in \( \mathbb{Z}_p^* \) and that the same holds for \( t = \alpha^{-1} y_B r^{-1} \) and \( r^{-1} \). This is the first query answer involving the random choices \( r^{-1} \) and \( z_A \). Therefore the random choices are independent. Any subsequent query answer in phases I and II will be consistent with the random choices.

As in game WOAUTH–IND–CPA a query to authorize Alice to receive messages from the adversary produces no output at the adversary.

**Phase I:** If the adversary queries to be sent a security token, the simulator uniformly chooses \( \kappa \in \mathbb{Z}_p^* \). It sends \( k^y_B = (g^\kappa)^y \) to the adversary.

The adversary may send security token \( k^y_A \) to the simulated Alice.

The adversary queries to encrypt \( m \). If the adversary specifies security token \( k^y_B \), the simulator sets \( k' = k^y_B \).

If the adversary specifies security token \( k^y_A \), the simulator computes

\[
k' = (k^y_A)^{y^{-1}}
\]

If the adversary does not specify a security token, the simulator uniformly chooses

\[
k \overset{R}{\leftarrow} \mathbb{Z}_p^*, k' = (g^\kappa)^y
\]

The simulator uniformly chooses \( r \in \mathbb{Z}_p^* \) and sends to the adversary

\[
C = g^{\alpha r}, D = m \cdot \varepsilon(k', g^{\alpha r} y^\delta)
\]

Note that the use of \( \gamma \) and \( \delta \) does not contradict the independency claim for \( \gamma \) and \( \delta \), since we perfectly maintain the distributions of the encryption scheme. We simulate the reverse operation to game WOAUTH–IND–CPA and use the bilinear map of the decryption function instead.

If the adversary chooses to end this phase, the simulator proceeds.

**Challenge:** There is no need to choose the security token \( k_t^y_B \) (or \( k_t^y_A \)) yet, since no output is produced at the adversary,
but assume $\kappa^* = a$. The adversary chooses two plaintexts $m^n_0$ and $m^n_1$ and gives them to the simulator. The simulator flips a random coin $f \in \{0, 1\}$. It computes

$$C^* = (g^b)^{\alpha r}, D^* = m^n_f \cdot \langle \theta, g \rangle^d$$

It sends the challenge ciphertext $c^* = \langle C^*, D^* \rangle$ to the adversary.

**Phase II:** The simulator responds to the queries as in phase I. If the adversary queries for an encryption under $k^n_D$, the simulator uniformly chooses $r \in \mathbb{Z}_p^*$ and computes

$$C = g^r, D = m \cdot \langle \theta, g \rangle^\alpha g^r$$

Note that the adversary does not know $k^n_D$, but he might know the associated unique identifier of the RFID item.

If the adversary chooses to end this phase, the simulator proceeds.

**Guess:** The adversary outputs a guess $f^*$ of $f$. If $f^* = f$, the simulator outputs $d = abc$ and otherwise $d = r$.

**Claim:** The view of the adversary is as in a real attack, if the problem instance is an MBDDH quintuple. The simulated view is perfectly indistinguishable, even for the protocols involving the replaced random choice $y$. In case $d = r$, the challenge ciphertext $c^*$ contains no information about the plaintext $m^n_f$. It is uniformly distributed. Therefore, if the adversary has advantage $\epsilon$ in breaking the game $WOFWD - IND - CPA$, the simulator has advantage $\frac{\epsilon}{2}$ in solving the MBDDH problem.

\[\square\]

6. RELATED WORK

Our encryption scheme is an instance of attribute-based encryption (ABE). ABE is a special form of identity-based encryption (IBE). The concept of IBE has been proposed in [22] and the first practical implementation has been presented in [3].

IBE is an alternative to public-key cryptography. Instead of creating private/public-key pairs, one can use any string, i.e. the identity, to encrypt. Then there is an authoritative private key generator that can create the private key for an identity. As a result there is no need to distribute public keys before communication any longer.

The basic idea of ABE is now simple. Let the identity be an attribute, similar to an attribute in attribute-based access control. For example, let there be an attribute for being older than 18 years and one for possessing a driver’s license. The challenge is to combine attributes, i.e. decryption should only be possible, if a set of attributes are available. This enables to specify complex policies.

This idea has been realized early and the first proposal has been made in [23]. It already allowed combining attributes by logical AND and OR combinations. It also noted the close relation to access control and the ability to enforce access control on published data. The term attribute-based encryption has been first used in [10]. An implementation based on threshold combination of attributes has been introduced in [19]. Note that threshold allows to express logical AND combinations as $n - out - of - n$ threshold and logical OR combinations as $1 - out - of - n$ threshold. A generalization to AND and OR combinations has then been made again in [8]. This has been extended to also include negations, i.e. logical NOT, in [18].

A challenge of attribute-based encryption is that, if parties collude, they are able to decrypt the ciphertext of the union of their attributes. Collusion resistance has been built into ABE in [1].

ABE has also been combined with proxy re-encryption in [26]. They use it for revocation, but do not extend the policy model. ABE has been adapted to publish-subscribe networks in [24]. As we already mentioned, we furthermore extend the notion of ABE to also include object-related attributes, in particular our visibility policies. We are not aware of any ABE scheme that addresses this challenge.

IBE (not even ABE) has been used to protect item-level data in a shared setting before. In [9] a specialized access control enforcement for data from ubiquitous devices – like RFID – is proposed. It also does not yet include object-related policies or the more powerful visibility policies.

Enforcement of visibility policies in distributed databases can be achieved using the authentication protocol of [15]. Of course, one cannot authenticate each event in a publish-subscribe network.

7. CONCLUSION

In this paper we have studied an attribute-based encryption scheme tailored for the use in RFID-enabled supply chains. In such supply chains data exchange is sensitive and restrictive access-control policies are necessary. Nevertheless information is often distributed in publish-subscribe networks where there is no access control policy enforcement point and recipients might even be unknown. Our encryption scheme enables the publisher to selectively encrypt its messages, such that only recipients that have had possession of the same item are able to decrypt. We also enable a trusted third party to trace the forwarding of items, such that there is an incentive to keep security-relevant information secret. We prove our scheme secure in the standard model using common security assumptions.

Future work includes to extend the scheme to a cloud-based setting with a central, but encrypted database. In this case, there is a need to query the database about an item without compromising the confidentiality of events.

8. REFERENCES


