Scalable Data Access Control in RFID-Enabled Supply Chain

Saiyu Qi*†, Yuanqing Zheng†, Mo Li†, Yunhao Liu‡ and Jinli Qiu§
syqi@ust.hk, yuanqing1@e.ntu.edu.sg, limo@ntu.edu.sg, yunhao@greenorbs.com, jinliqiu1984@gmail.com
*Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong
†School of Computer Engineering, Nanyang Technological University, Singapore
‡TNLIST, School of Software, Tsinghua University, China
§Department of Computer Science and Technology, Xi'an Jiaotong University, China

***Abstract*—By attaching RFID tags to products, supply chain participants can identify products and create product data to record the product particulars in transit. Participants along the supply chain share their product data to enable information exchange and support critical decisions in production operations. Such an information sharing essentially requires a data access control mechanism when the product data relates to sensitive business issues. However, existing access control solutions are ill-suited to the RFID-enabled supply chain, as they are not scalable in handling a huge number of tags, introduce vulnerability to the product data, and performs poorly to support privilege revocation of product data. We present a new scalable data access control system that addresses these limitations. Our system provides an item-level data access control mechanism that defines and enforces access policies based on both the participants' role attribute and the products' RFID tag attribute. Our system further provides an item-level privilege revocation mechanism by allowing the participants to delegate encryption updates in revocation operation without disclosing the underlying data contents. We design a new updatable encryption scheme and integrate it with Ciphertext Policy-Attribute Based Encryption (CP-ABE) to implement the key components of our system.**

I. Introduction

In supply chain data management, participants can attach RFID tags to products, automatically identify products and create product data to record the product particulars in transit. The product data can thus be shared among participants, which facilitates information exchange and supports critical decisions in production operations [1], [2].

To facilitate data management, a service provider is usually appointed to coordinate the data sharing between participants. Each participant only needs to communicate directly with the service provider for data submission/retreival. In such a paradigm, each product is associated with an RFID tag with a unique ID that can be used to index its relevant product data managed by the service provider. When a participant receives a tagged product, it can use the tag ID as an index to submit its own product data or retrieve the corresponding product data submitted by other participants. In practice, the service provider could be a commercial supply chain manager such as GT Nexus [3] or a certain participant in the supply chain.

Product data stored in the service provider can be closely related to sensitive business issues and thus the data privacy becomes a natural concern. Strict data access control should be applied which disallows unauthorized data access from other entities or even the service provider itself. For instance, the pharmaceutical supply chain tracks tagged drugs and establishes a pedigree for each drug, e.g., counterfeit certificate, time of delivery, manufacturers, etc [4]. The drug pedigrees should be accessed by retailers and consumers to check whether the drugs are from trustworthy participants [5]. However, drug pedigrees may contain sensitive business matters and suffer various malicious accesses. Drug counterfeiters and competitive manufacturers can exploit software vulnerabilities to gain unauthorized access to the service provider; a curious service provider may actively explore the content of its managed drug pedigrees; and any attackers with physical access to the service provider can access all the drug pedigrees in memory. Therefore, participants highly desire the data confidentiality and access control based on participant defined policies.

A straightforward but inefficient approach is to encrypt product data using cryptographic primitives and distribute decryption keys only to authorized participants. A large scale supply chain, however, needs to handle millions of products going through many participants. The products can spread worldwide and thus the intermediate participants as well as the final consumer of a product are generally unknown in advance. Therefore, it is hard to manage and distribute decryption keys in advance.

In practice, instead of designating a particular partner, a participant may want to provide access to the ones with certain attributes. For instance, a USA drug manufacturer may allow access to the ones that have both 'USA' and 'FDA' attributes, or the ones that are 'Retailer' or 'Consumer'. Recently, a new cryptographic primitive called Ciphertext Policy-Attribute Based Encryption (CP-ABE) [9] is proposed to provide attribute based access control. With CP-ABE, a user can specify access policies based on logical expressions over user attributes for data encryptions before uploading to a third-party database. A key authority assigns each user a credential corresponding to the set of attributes that describe the user's features (e.g., company nationality, business domain, etc). CP-ABE ensures that only users with attributes satisfying the logical expression can decrypt the data. In addition, as the database only stores encrypted data, it cannot learn any nontrivial information about the data.

Despite its benefits, CP-ABE is ill-suited to secure the product data of RFID-enabled supply chains for three main reasons. First, CP-ABE was designed to protect data associated

978-1-4799-6204-4/14 $31.00 © 2014 IEEE
DOI 10.1109/ICNP.2014.28

with users, providing data access control in user-level; it is thus not scalable to protect product data associated with products, which requires item-level data access control. Second, CP-ABE relies on a key authority to manage the credentials of attributes, introducing vulnerability to product data. All the supply chain participants have to trust the key authority to securely manage their credentials; if the authority is compromised or the credentials are occasionally exposed, the product data of the whole supply chain can be decrypted. Third, CP-ABE introduces substantial credential issuing overhead for supply chain participants to revoke access privilege of product data in item-level.

In this paper, we develop a scalable data access control system for RFID-enabled supply chain that (1) provides item-level data access control and (2) enforces item-level privilege revocation.

Our system provides a new item-level data access control mechanism, which defines and enforces role attributes (for participants) and tag attributes (for products) to regulate item-level data access control. A participant may define an access policy to be a logical expression over the role attributes and a tag attribute. An authorized partner must have the credentials of both satisfactory role attributes and the tag attribute so as to access the corresponding data of the particular product, which preclude adversaries outside the supply chain (who lack the credential of tag attributes) or unauthorized participants within the supply chain (who lack the credential of satisfactory role attributes). More importantly, the credentials of role attributes for participants are issued by a key authority while the tag attributes and their credentials can be locally generated by participants and stored in tags for distribution. Our system uses this mechanism to address the first two limitations of CP-ABE. Specifically, the mechanism scales to support millions of products where the generation and maintenance of their tag attributes and credentials do not have to go through the key authority. Moreover, since the credentials of tag attributes are locally managed by participants, the key authority does not have the ability to decrypt the product data of the supply chain.

Our system provides a new item-level privilege revocation mechanism in a way that overcomes the inefficiency hurdles of CP-ABE. Generally, our revocation mechanism consists of two steps. When a participant wants to revoke the data access privilege of some participants for a product, it first updates the tag attribute/credential pair stored in the tag. It then delegates the service provider to update the tag attribute contained in the access policies of the corresponding encrypted product data without revealing any additional information about the data. Our system uses this mechanism to address the third limitation of CP-ABE. The mechanism does not involving the key authority to issue any credentials to participants and only incurs minimal computation and communication overhead.

We devise a new encryption scheme called updatable encryption and integrate it with Ciphertext Policy-Attribute Based Encryption (CP-ABE) [9] to enforce the key components of our system. With complete design, we further evaluate the performance of our system through large-scale experiments to show its efficiency and scalability.

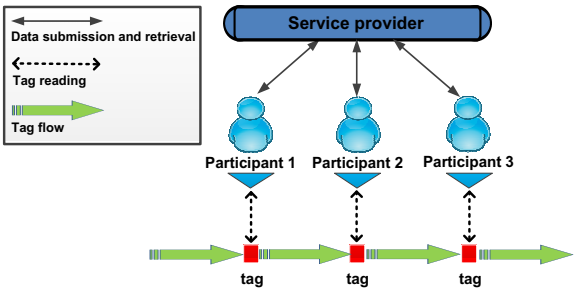


Fig. 1. An example of RFID-enabled supply chain.

The rest of this paper is organized as follows. We describe the system model and problem in Section II. We present the design details of our system in Section III. In Section IV, we conduct theoretical analysis to prove the security properties of our schemes. In Section V, we examine the efficiency of our system. At last, we review the related works in Section VI and conclude this paper in Section VII.

II. System Model and Problem Description

A. RFID-enabled Supply chains

An RFID-enabled supply chain is typically composed of three components: RFID tags (attached to products), readers (owned by participants for tag interrogation) and a database (at the service provider for data management). Supply chain participants (e.g., manufacturers, deliverers, and retailers) create, store, and share product data through the service provider and leverage the RFID tags to track the product records in the database.

Figure 1 depicts an illustrative example, where the supply chain consists of three participants. When a product enters the supply chain, it is labeled with a tag. The tagged product then flows through the supply chain and is sequentially processed by the participants. The participant can create its own product data on the product and set access policy for other participants to access.

As for the supply-chain structure, we assume that participants may only know their direct upstream and downstream partners and may continuously join and leave the chain. For ease of presentation, we assume that one centralized service provider is appointed for data management. In practice, however, the service provider can be deployed in a distributed manner or in a cloud.

B. Security model

In this paper, we consider honest but curious service provider, i.e., the service provider follows our proposed system in general, but may try to explore as much product data as possible based on participant inputs. We also consider that participants may try to access product data outside the scope of their access privileges independently or cooperatively. Communication channel between the participants and the service provider are assumed to be secured using security protocols like SSL.

C. Design goals

In this paper, we mainly focus on the following four design goals:

- **Data privacy:** The basic design goal is to prevent unauthorized entities from learning any nontrivial information about the product data submitted by participants. As the product data may contain highly sensitive information about products (e.g., counterfeit information, business relationship between participants, etc.), we need to strictly ensure data privacy.
- **Item-level access control:** When a tagged product flows through a supply chain, each participant should be able to specify an access policy to its own product data about the product. The policy should be fine-grained so that the participant can precisely define the authorized partners.
- **Item-level privilege revocation:** We consider the dynamic feature of supply chains where upstream participants may leave the supply chain after completing their production tasks. When a participant receives a tagged product, it should be able to: 1) revoke the access privilege of its upstream participants to the product data generated by its downstream participants, and 2) at the same time preserve the access privilege of the downstream participants to the product data generated by the upstream participants.
- **Scalability and efficiency:** We want to achieve item-level access control and privilege revocation with high scalability. Also, We want to incur small memory cost that is affordable for low cost RFID tags and prevent implementing complex cryptographic algorithms on tags.

D. Limitations of CP-ABE

CP-ABE fails to achieve the above design goals with efficiency and security guarantees mainly due to following three reasons.

First, CP-ABE is not scalable to protect product data associated with products, which requires item-level data access control. Suppose a tagged product enters the supply chain and a participant wants to share its product data about the product with its partners. As the data is associated with the product, the participant regulates: (1) who are retailers in USA or France *and* (2) who have the privilege to process the product (within the same supply chain) can access this data. Note that the condition (2) is necessary to preclude a retailer in USA but in another supply chain to access this data.

To enforce such a regulation, a strawman CP-ABE solution is to define a small set of role attributes (e.g. 'retailer', 'USA', 'France') to describe the features of participants and a large set of tag attributes to identify each tagged product, and appoint a key authority to issue credentials of proper attributes to each participant. In this way, the participant can specify an access policy ('retailer' AND ('USA' OR 'France') AND 'TagAtt') for its product data; and a participant with the credential of attributes {'retailer', 'USA', 'TagAtt'} can decrypt the data. However, this means that when a tagged product flows through the supply chain, each participant within the chain has to apply a credential for its role attributes as well as the tag attribute, which poses heavy computation and communication overhead on the key authority when handling millions of tagged products.

As the overhead of a credential is proportional to the number of its attributes, a better way is to manage role attributes and tag attributes separately. A participant applies from the key authority a credential of its role attributes at once and credentials of tag attributes for tagged products. Unfortunately, CP-ABE disallows a participant to jointly use two different credentials to decrypt the product data. This is due to the collusion-resistance property of CP-ABE, which states that multiple credentials can only be able to decrypt an encryption if at least one of the credentials could decrypt it on its own. Originally, this property is used to prevent large-scale data leakage from an attacker that manages to get a hold of a few credentials.

Second, all the participants have to trust the key authority to securely manage their credentials, which introduces vulnerability to their product data. The credentials may be leaked due to ill-managements. An adversary can exploit software vulnerabilities to gain unauthorized access to servers; curious or malicious administrators at a hosting can snoop on the credentials; and attackers with physical access to servers can access all credentials in memory. The leaked credentials could be used to decrypt the product data of the whole supply chain stored in the service provider. Trusting a key authority to securely manage their credentials is often unacceptable to supply chain participants whose product data is highly sensitive.

Third, CP-ABE introduces substantial credential issuing overhead to support item-level revocation. To revoke the access privilege of its upstream participants for the data of a product, a participant has to inform the key authority to issue two credentials with the old tag attribute and a new tag attribute (respectively) for each of its downstream participants. For instance, suppose the revoked participants use the tag attribute 'TagAtt' to regulate the access policy of their product data. A downstream participant with role attributes {'retailer', 'USA'} has to acquire a credential of attributes {'retailer', 'USA', 'TagAtt'} and a credential of attributes {'retailer', 'USA', 'New-TagAtt'} from the key authority. The downstream participant can then: (1) use the new tag attribute 'New-TagAtt' to regulate the access policy of its product data to preclude the access privilege of the revoked participants; (2) still use the former credential to access the product data of the revoked participants; and (3) use the latter credential to access the product data of other downstream participants.

III. System design

A. Design principle

1) Item-level data access control: We use two types of attributes: role attributes and tag attributes to specify data access policies. Role attributes generally refer to different properties of participants. For instance, 'USA' describes the location of participants; 'drug' describes the production type of manufacturers. Role attributes alone however do not provide sufficient granularity to specify item-level data access policies. We thus introduce tag attributes to identify tagged products. A tag attribute is similar in essence to a unique tag ID, but is

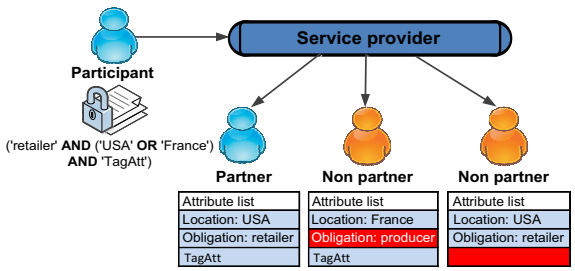


Fig. 2. An example of item-level data access control.

carefully designed to enforce cryptographic functionality. We use 'TagAtt' to denote a tag attribute.

Access policy to the data of a product is defined as a logical expression over role attributes AND the associated tag attribute. As a result, a participant can specify access policy in item-level by incorporating the tag attribute in the policy when encrypting product data. Figure 2 illustrates a usage scenario where a participant wants to share its product data with retailers in USA and France. The participant can specify the access policy as ('retailer' AND ('USA' OR 'France') AND 'TagAtt'). By doing so, the unauthorized participant with role attribute 'producer' and the one without the tag attribute 'TagAtt' are precluded from the data, while an authorized participant with attributes satisfying the access policy can access. We note that the service provider is also excluded from the product data since it does not have satisfiable attributes.

The data access privileges of participants are regulated by attribute credentials, i.e., an authorized participant must have the credentials of satisfiable attributes to decrypt product data. Our system allows the credentials of attributes to be issued in a distributed manner. Specifically, the small set of role attributes and their credentials are managed by a key authority, while the large set of tag attributes and their credentials are managed locally by participants. A participant generates encrypted product data with access policy defined from role attributes and tag attributes and uploads the policy enforced encryptions to the service provider.

When a downstream participant receives a tagged product, it can retrieve the policy enforced encryptions (submitted by upstream participants) about the product from the service provider. Without credential of the tag attribute, however, the participant cannot decrypt the data. As large number of tagged products can flow through many participants within the supply chain, it is hard to distribute the credentials of tag attributes for all the participants in advance. To solve this problem, we leverage tags as a natural medium to distribute the credentials of their attributes to all the participants (within the supply chain). Although commodity passive tags only have very limited memory (e.g., 512 bits in ALN-9640 tags), our compact design of tag attribute credentials can fit in such a small space. We put the management of tag credentials in the hands of the participants, thus do not involving the key authority to issue credentials of tag attributes and disabling its ability to decrypt the product data.

Enforcement of distributed credential issuing: We enforce the above distributed credential issuing mechanism by combing symmetric encryption scheme with CP-ABE [9]. We apply CP-ABE to encode the credentials of role attributes into CP-ABE decryption keys. Initially, the key authority generates a public/master key pair for a list of role attributes and publishes the corresponding public key. Each participant then requests the key authority for a credential of a set of role attributes which the participant possesses.

To incorporate tag attributes, we encode credentials of tag attributes into symmetric keys. When a tagged product enters the supply chain, the first participant of the chain generates a tag attribute and a symmetric key. The tag attribute-symmetric key pair forms a tag token and is stored in the attached tag. To prevent the tag token to be stolen by an outsider of the supply chain, a participant can use lightweight encryption schemes such as AES to encrypt the tag token and share the decryption key with its direct upstream and downstream participants.

We use double encryption paradigm to generate policy enforced encryption. Given a tagged product, a participant first encrypts to a logical expression over role attributes using the CP-ABE scheme and then encrypts the result by using the symmetric key contained in the tag token. Such a paradigm precisely enforces the access policy desired in our system (a logical expression over role attributes AND a tag attribute). Participants can decrypt the policy enforced encryption only if they possess the credentials of both satisfiable role attributes and the tag attribute.

2) Item-level privilege revocation: Item-level privilege revocation means that a participant could revoke the access privileges of its upstream participants for the data of a product. To preserve regular data access control after the revocation, the participant should be able to complete two tasks: 1) revoke the access privileges of its upstream participants to the product data generated by its downstream participants, and 2) meanwhile preserve the access privileges of the downstream participants to the product data generated by the upstream participants.

Our revocation mechanism allows participants to update tag tokens locally (to complete task 1) and delegates the service provider to update the access policies of the policy enforced encryptions submitted by revoked participants (to complete task 2). The update operation does not require decryption, which only involves lightweight overhead while still preserving data privacy. Comparing with CP-ABE, Our mechanism allows the participant to complete revocation operation by itself, thus does not involving the key authority to issue credentials to any participants.

Figure 3 illustrates a usage scenario of our revocation mechanism, where participant 2 wants to revoke the access privilege of the upstream participant 1 for the data of a product. To do so, participant 2 directly updates the tag token stored in the attached tag. Participant 2 then sends a delegation request to the service provider to enable it to update the tag attribute of the policy enforced encryptions submitted by participant 1. After the two steps, participant 1 is precluded from the product data of participant 3 as participant 3 will use the updated tag token to encrypt its data, while participant 3 can still access the product data of participant 1. During the above process, the key authority is not involved for credential issuing.

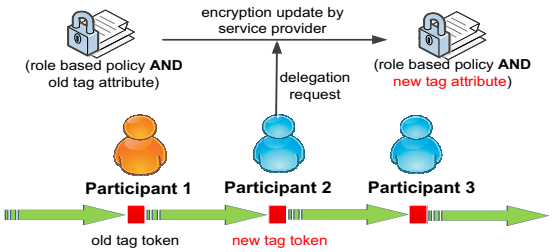


Fig. 3. An example of item-level revocation.

Enforcement of access policy updating: General symmetric encryption scheme does not support access policy updates without decryption. To solve this problem, we design a new encryption scheme called updatable encryption scheme to support this functionality. We use this scheme to replace the general symmetric encryption scheme. As a result, the credentials of tag attributes become the secret keys of the updatable encryption scheme.

Our updatable encryption scheme is inspired by the Proxy Re-Encryption [16], in which a proxy can update an encryption under an old key to another encryption under a new key without decryption. Such an approach, however, requires 1024-bit encryption keys to achieve standard security, while current passive tags only have 512-bit user memory. Furthermore, the security of the scheme is not proved in formal security model. Besides, recent proposals [22]–[24] only allow one-time update operation, while we desire multi-time update operations to support multiple revocation operations by different participants within the same supply chain.

We design a new updatable encryption scheme to overcome the above drawbacks. The tag token consists of a tag attribute and a secret key of the updatable encryption scheme. Our design ensures the length of the tag token to be suitable for current passive tags. In a revocation operation, the participant sends a re-key to the service provider for it to update the access policies of policy enforced encryptions. Our design also ensures that the leakage of re-key does not break the data privacy and the policy enforced encryptions can be updated multiple times. We define two security models in supply chain setting to formalize malicious revoked participants and malicious service provider, respectively. We prove the security of our updatable encryption scheme in the two models.

B. Cryptographic primitives: CP-ABE

The CP-ABE scheme [9] consists of four algorithms: (Setup, Encrypt, KeyGen, Decrypt).

- Setup(λ): On input a security parameter λ, the algorithm outputs a public key *PK* and a master key *MSK*.
- Encrypt(*PK*, *RP*, m): On input the public key *PK*, an access policy *RP* and a message m, the algorithm outputs an encryption *CT*.
- KeyGen(*MSK*, *S*): On input the master key *MSK* and a set S of attributes, the algorithm outputs a private key *SK*. An attribute can be any string.
- Decrypt(*PK*, *CT*, *SK*): On input the public key *PK*, an encryption *CT* and a private key *SK*, if the set S of attributes associated with *SK* satisfies the access policy *RP* associated with *CT*, the algorithm outputs a message m.

C. Updatable encryption

Our updatable encryption scheme builds on bilinear map. We briefly introduce it before describing our scheme.

Bilinear map: Bilinear map [21] is a mathematical tool to construct a rich types of cryptographic primitives. We present a few facts related to efficiently computable bilinear maps. Let G_1, G_2, G_T be groups of prime order p. Let g and h be generators of G_1 and G_2 respectively. Let e be a bilinear map: $G_1 \times G_2 \rightarrow G_T$. The bilinear map e has three properties: (1) for all $u \in G_1$, $v \in G_2$ and x, $y \in Z_p$, $e(u^x, v^y) = e(u, v)^{xy}$; (2) $e(g, h) \neq 1$; (3) $(p, G_1, G_2, G_T, e, g, h)$ can be efficiently generated and the bilinear map e: $G_1 \times G_2 \rightarrow G_T$ is efficiently computable.

Our updatable encryption scheme consists of six algorithms: (USetup, UKeyGen, UEncrypt, UDecrypt, UKeyUpdate, EncUpdate). The first algorithm USetup outputs some parameters for other algorithms to use. The subsequent three algorithms (UKeyGen, UEncrypt, UDecrypt) consist of a general symmetric key encryption scheme. The last two algorithms (UKeyUpdate, EncUpdate) are designed for key updating and encryption updating respectively. The functionalities of the six algorithms are shown as follows:

- USetup(λ): On input a security parameter λ, the algorithm outputs a public parameter *ugp* and a private parameter *usp*.
- UKeyGen(*usp*): On input the private parameter *usp*, the algorithm outputs a secret key *USK*.
- UEncrypt(m, *ugp*, *USK*): On input a message m, the public parameter *ugp* and a secret key *USK*, the algorithm outputs an encryption C^{UP}.
- UDecrypt(C^{UP}, *USK*): On input an encryption C^{UP} and a secret key *USK*, the algorithm outputs a message m.
- UKeyUpdate(*USK*): On input a secret key *USK*, the algorithm outputs a new secret key *USK'* and a re-key *rk* between the old/new key pair.
- EncUpdate(C^{UP}, *rk*): On input an encryption under *USK* and a re-key *rk*, the algorithm outputs a new encryption $C^{UP'}$ under *USK'*. This algorithm updates an encryption under a secret key to an encryption under a new secret key by using the re-key between the two secret keys.

Table I presents the details of our updatable encryption scheme. We explain our design in three steps. We first show how to use a secret key *USK* to encrypt/decrypt an encryption C^{UP}. We then describe how to update *USK* to a new key *USK'* as well as a re-key *rk*. Finally, we describe how to use *rk* to update C^{UP} encrypted by *USK* to a new encryption $C^{UP'}$ encrypted by *USK'*.

USK contains two elements $(g^a, h^{\frac{x}{a}})$. In the execution of UEncrypt(m, *ugp*, *USK*), *ugp*=$X=e(g,h)^x$ is used to generate a secret X^s, which is used to hide m in an encryption element C_1. Also, the element g^a of *USK* is used to generate another encryption element C_2. The final encryption C^{UP} consists of two encryption elements (C_1, C_2).

In the execution of UDecrypt(C^{UP}, *USK*), $h^{\frac{x}{a}}$ is used with C_2 to reconstruct the secret X^s through bilinear map. C_1 is then divided by X^s to recover the message m.

TABLE I
DETAILS OF UPDATABLE ENCRYPTION

- USetup(λ): Chooses a random number $x \in Z_p$ and computes $X=e(g,h)^x$. Outputs *ugp*=X and *usp*=x.
- UKeyGen(*usp*): Given *usp*=x, chooses a random number $a \in Z_p$, computes g^a and $h^{\frac{x}{a}}$. Outputs *USK*=$(g^a, h^{\frac{x}{a}})$.
- UEncrypt(m, *ugp*, *USK*): Given m, *ugp*=X and *USK*=$(g^a, h^{\frac{x}{a}})$, chooses a random number $s \in Z_p$ and computes $C_1=mX^s$ and $C_2=(g^a)^s$. Outputs $C^{UP}=(C_1, C_2)$.
- UDecrypt(C^{UP}, *USK*): Given $C^{UP}=(C_1, C_2)$ and *USK*=$(g^a, h^{\frac{x}{a}})$, computes:

$$e(C_2, h^{\frac{x}{a}}) = e((g^a)^s, h^{\frac{x}{a}}) = e(g,h)^{xs} = X^s$$

Computes $m=\frac{mX^s}{X^s}$ and outputs m.
- UKeyUpdate(*USK*): Given a *USK*, first updates *USK* as:

$$USK' = (g^{aa'}, h^{\frac{x}{aa'}}) \leftarrow USK = (g^a, h^{\frac{x}{a}})$$

where a' is chosen randomly from Z_p. Sets *rk*=a'. Outputs *USK'* and *rk*.
- EncUpdate(C^{UP}, *rk*): Given C^{UP} and *rk*, updates C^{UP} as:

$$C^{UP'} = (C_1, C_2^{rk}) \leftarrow C^{UP} = (C_1, C_2)$$

where $(C_1, C_2^{rk})=(C_1, ((g^a)^s)^{a'})=(C_1, (g^{aa'})^s)$. Outputs $C^{UP'}$.

In the execution of UKeyUpdate(*USK*), both the elements $(g^a, h^{\frac{x}{a}})$ of *USK* are refreshed by a random number $a' \in Z_p$. The distribution of the new secret key *USK'* is identical to a secret key generated by UKeyGen(*usp*). The re-key *rk* between *USK*/*USK'* is a'.

In the execution of EncUpdate(C^{UP}, *rk*), the second element C_2 of C^{UP} is refreshed by *rk*. By doing so, we convert the encryption under *USK* into a new encryption under *USK'*. The update operation does not require any decryption.

D. Item-level access control and revocation

In the following, we describe how to integrate our updatable encryption scheme with CP-ABE to achieve item-level access control and revocation. For clarity, we term "a tagged product" as "a tag" in this subsection.

Initialization: The service provider, key authority and participants need to exchange some cryptographic parameters. The key authority generates (*PK*, *MSK*)$\leftarrow$Setup(λ) for its managed role attributes. It publishes *PK* and the role attributes and keeps *MSK* secret. Also, the first participant in the supply chain generates (*ugp*, *usp*)$\leftarrow$USetup(λ). The participant delegates the service provider to publish *ugp* and keeps *usp* secret. On the other hand, each participant maintains a table to store the published *PK*, role attributes and *ugp*. Each participant then selects a subset of suitable role attributes and acquires the corresponding credential from the key authority.

Tag preparation: When a tag T enters the supply chain, the first participant generates a random number as tag attribute att_T and a secret key $USK_T \leftarrow$UKeyGen(*usp*) as the corresponding credential. The participant sets the tag token as (att_T, USK_T) and saves the tag token into T. When T flows through the supply chain, each participant can perform three operations: data submission, data retrieval and privilege revocation by using the tag token.

Data submission: This operation allows a participant with tag token (att_T, USK_T) to specify an access policy to its product data and submit the policy enforced encryption to the service provider. Our system uses general symmetric encryption scheme to enlarge the plaintext size supported by both updatable encryption scheme and CP-ABE—updatable encryption scheme/CP-ABE only encrypts a symmetric key and use the key to encrypt the plaintext. For simplicity, we will not explicitly indicate the usage of the general symmetric encryption scheme.

The participant first defines a role based policy *RP* and ABE-encrypts the product data bound to *RP*. The resulted ABE encryption *CT* contains a policy part P and an encrypted data part C. The participant then encrypts C by using the secret key USK_T contained in the tag token to get the policy enforced encryption (C^{UP}, P). Algorithm 1 presents the encryption process.

Algorithm 1 (role attributes, *PK*, product data, *ugp*, tag token (att_T, USK_T))

1: Define *RP* according to the role attributes
2: *CT* = *(P, C)* $\leftarrow$ Encypt(*PK*, *RP*, product data)
3: $C^{UP} \leftarrow$ UEncrypt(*C*, *ugp*, USK_T)

Finally, the participant submits a record (att_T, C^{UP}, *P*) to the service provider.

Data retrieval: This operation allows a participant with tag token (att_T, USK_T) to retrieve the product data submitted by other participants. The participant first uses the tag attribute att_T as an index to retrieve all the submitted records about the tag. For each retrieved record (att_T, C^{UP}, *P*), the participant decrypts C^{UP} by using the secret key USK_T. The decrypted C and P forms a valid ABE encryption CT. After that, the participant uses its credential of role attributes to decrypt the ABE encryption. Algorithm 2 presents the key operations in data retrieval.

Algorithm 2 (record (att_T, C^{UP}, *P*), tag token (att_T, USK_T), *PK*, credential of role attributes)

1: $C \leftarrow$ UDecrypt(C^{UP}, USK_T)
2: *CT* $\leftarrow$ *(P, C)*
3: $m \leftarrow$ Decrypt(*PK*, *CT*, credential of role attributes)

Revocation: This operation allows a participant with the tag token (att_T, USK_T) of a tag T to revoke the access privilege of its upstream participants for the data of T. The participant first generates a new secret key USK'_T and a re-key *rk*: (USK'_T, *rk*)$\leftarrow$UKeyUpdate(USK_T).

The participant updates the tag token to (att_T, USK'_T) by overwriting the original tag token in T. The participant then uses USK'_T to generate policy enforced encryption for its product data. By doing so, the participant precludes its upstream participants from the newly uploaded data.

In addition, the participant needs to update the submitted policy enforced encryptions about T so that the downstream participants can still decrypt them with the new tag token. To do so, the participant sends (att_T, *rk*) to the service provider. For each submitted record (att_T, C^{UP}, *P*) about T, the service provider updates C^{UP} as $C^{UP'} \leftarrow$EncUpdate(C^{UP}, *rk*) and updates the record as (att_T, $C^{UP'}$, *P*). After the update

operation, all the downstream participants can still access these records using the new tag token.

IV. SECURITY ANALYSIS

We first analyze the security properties of our updatable encryption scheme as the security of our system builds on these properties.

A. Security analysis of updatable encryption scheme

In the following, we prove that our updatable encryption scheme provides data privacy against revoked participants as well as the service provider. We formally define the desired security properties based on interactive game, which consists of an adversary (a malicious revoked participant or the service provider) and a challenger (our updatable encryption scheme). The adversary interacts with the challenger to attack our scheme. We then show that our updatable encryption scheme satisfies these security properties.

The data privacy of an encryption scheme can be formalized as *IND-CPA* [25]. Comparing with the standard definition of *IND-CPA*, a revoked participant and the service provider also own additional knowledge about revoked secret key and re-keys, respectively. We thus extend *IND-CPA* to define two security properties: *Revoked Secret Key Resistant (RSKR)-IND-CPA* and *Re-Key Resistant (RKR)-IND-CPA*. We prove that our updatable encryption scheme satisfies the two properties.

Definition 1. *RSKR-IND-CPA*: An updatable encryption scheme satisfies *RSKR-IND-CPA* if all polynomial time adversaries have at most a negligible advantage to win the following 3-phase game.

Phase 1: The challenger runs the algorithms of the updatable encryption scheme to generate a public parameter *UGP*, a private parameter *usp*, a secret key *USK*. The challenger returns (*UGP*, *USK*) to the adversary. Upon receiving the pair, the adversary returns a polynomial bounded number q to the challenger.

Phase 2: The challenger runs the algorithms of the updatable encryption scheme to generate a sequence of secret keys USK_1, USK_2,$\cdots$, USK_q from *USK*. The challenger then allows the adversary to send a polynomial bounded number of queries. In each query, the adversary selects a number $i\in[1, q]$, a message m and sends (i, m) to the challenger. The challenger generates an encryption C_i^{UP} of m by USK_i and returns C_i^{UP} to the adversary.

Phase 3: The adversary selects a number $i^*\in[1, q]$, generates two equal-length messages m_0^*, m_1^* and returns a challenge (i^*, m_0^*, m_1^*) to the challenger. The challenger selects a message m_η^* from (m_0^*, m_1^*) by flipping a fair bit η. The challenger then generates an encryption $C_{i^*}^{UP}$ of m_η^* by USK_{i^*} and returns $C_{i^*}^{UP}$ to the adversary.

Finally, the adversary returns a guess η'. If η=η', the adversary wins the game. We define the adversary's advantage in the above game as $|P[\eta=\eta']-\frac{1}{2}|$.

Theorem 1. Our updatable encryption scheme satisfies *RSKR-IND-CPA* based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption.

DBDH Assumption. Let y_1, y_2, y_3, $z\in Z_p$ be chosen at random. Let g and h be the generators of G_1 and G_2 respectively. The DBDH assumption [21] is that no probabilistic polynomial-time adversary $\mathcal{B}$ can distinguish the tuple (g, g^{y_1}, g^{y_3}, h, h^{y_1}, h^{y_2}, $e(g,h)^{y_1y_2y_3}$) from the tuple (g, g^{y_1}, g^{y_3}, h, h^{y_1}, h^{y_2}, $e(g,h)^z$) with more than a negligible advantage. The advantage of $\mathcal{B}$ is:

$$|P[\mathcal{B}(g, g^{y_1}, g^{y_3}, h, h^{y_1}, h^{y_2}, e(g,h)^{y_1y_2y_3})] = 0 - P[\mathcal{B}(g, g^{y_1}, g^{y_3}, h, h^{y_1}, h^{y_2}, e(g,h)^z)] = 0|$$

where the probability is taken over the random choice of the generators g and h, the random choice of y_1, y_2, y_3, z in Z_p, and the random bits used by $\mathcal{B}$.

Proof: Suppose there exists a polynomial time adversary $\mathcal{A}$ that can break *RSKR-IND-CPA* of our updatable encryption scheme with advantage ϵ. We can construct an adversary $\mathcal{B}$ to break the DBDH assumption with advantage $\frac{\epsilon}{2}$ as follows:

The challenger of DBDH game selects three groups: G_1, G_2, G_T with an efficient bilinear map e and generators of G_1 and G_2: g, h. The challenger flips a fair bit μ. If μ=0, the challenger generates (Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Z)=(g, g^{y_1}, g^{y_3}, h, h^{y_1}, h^{y_2}, $e(g,h)^{y_1y_2y_3}$); otherwise it generates (Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Z)=(g, g^{y_1}, g^{y_3}, h, h^{y_1}, h^{y_2}, $e(g,h)^z$). The challenger then gives the generated (Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Z) to $\mathcal{B}$.

Setup. $\mathcal{B}$ computes X=$e(Y_2, Y_6)$=$e(g,h)^{y_1y_2}$. $\mathcal{B}$ chooses a random number $\beta\in Z_p$ and computes $(Y_2)^\beta$=$g^{y_1\beta}$ and $Y_5^{\frac{1}{\beta}}$=$h^{\frac{y_1}{\beta}}$=$h^{\frac{y_1y_2}{y_2\beta}}$. $\mathcal{B}$ sets *UGP*=X and *USK*=($g^{y_1\beta}$, $h^{\frac{y_1y_2}{y_1\beta}}$) and returns (*UGP*, *USK*) to $\mathcal{A}$. The distribution of the constructed (*UGP*, *USK*) is identical with the valid ones generated by the algorithms of updatable encryption scheme. After that, $\mathcal{B}$ receives q from $\mathcal{A}$.

Query. $\mathcal{B}$ maintains a list with size q. When receiving a query (i, m) from $\mathcal{A}$, $\mathcal{B}$ checks if there exists a pair (i, β_i) in the list. If yes, $\mathcal{B}$ chooses a random number $s\in Z_p$ and constructs an encryption C_i^{UP}:

$$C_i^{UP} = (C_{i1}, C_{i2}) = (mX^s, (g^{\beta_i})^s)$$

If no, $\mathcal{B}$ chooses a random number $\beta_i\in Z_p$, adds (i, β_i) into the list and generates an encryption C_i^{UP} as above. $\mathcal{B}$ then returns C_i^{UP} to $\mathcal{A}$. The distribution of the constructed encryption is identical to the valid ones generated by the algorithms of updatable encryption scheme with the secret key USK_i.

Challenge. $\mathcal{A}$ submits a challenge (i^*, $m_0{}^*$, $m_1{}^*$) to $\mathcal{B}$. $\mathcal{B}$ flips a fair bit η, retrieves (i^*, β_{i^*}) from its maintained list and constructs an encryption $C_{i^*}^{UP}$:

$$C_{i^*}^{UP} = (C_{i1^*}, C_{i2^*}) = (m_\eta^* Z, (Y_3)^{\beta_{i^*}})$$

If μ=0, then Z=$e(g,h)^{y_1y_2y_3}$. Let s=y_3, then we have C_{i1^*}=$m_\eta^* e(g,h)^{y_1y_2y_3}$=$m_\eta^*(e(g,h)^{y_1y_2})^{y_3}$=$m_\eta^* X^s$ and C_{i2^*}=$(Y_3)^{\beta_{i^*}}$=$(g^{\beta_{i^*}})^{y_3}$=$(g^{\beta_{i^*}})^s$. The distribution of the constructed encryption is identical with the valid ones generated by the algorithms of updatable encryption scheme with the secret key USK_{i^*}. If μ=1, then Z=$e(g,h)^z$. We then have C_{i1^*}=$m_\eta^* e(g,h)^z$. Since z is random, C_{i1^*} is actually a random element of G_T from the view of $\mathcal{A}$ and the constructed encryption reveals no information about $m_\eta{}^*$.

Guess. $\mathcal{A}$ submits a guess η' of η. If η'=η, $\mathcal{B}$ will output μ'=0 to indicate that it was given a valid DBDH-tuple;

otherwise it will output μ'=1 to indicate that it was given a random 4-tuple.

In the case where μ=1, $\mathcal{A}$ gains no information about η. Therefore, we have $P[\eta' \neq \eta | \mu=1]=\frac{1}{2}$. Since $\mathcal{B}$ guesses μ'=1 when $\eta' \neq \eta$, we have $P[\mu'=\mu|\mu=1]=\frac{1}{2}$.

In the case where μ=0, $\mathcal{A}$ sees an encryption of m^*_η. The advantage of $\mathcal{A}$ is ϵ. Therefore, we have $P[\eta'=\eta|\mu=0]=\frac{1}{2}+\epsilon$. Since $\mathcal{B}$ guesses μ'=0 when $\eta'=\eta$, we have $P[\mu'=\mu|\mu=0]=\frac{1}{2}+\epsilon$.

Combing the above analysis, the overall advantage of $\mathcal{B}$ in the DBDH game is:

$$
\begin{aligned}
&P[\mu=0]P[\mu'=\mu|\mu=0]+P[\mu=1]P[\mu'=\mu|\mu=1]-\frac{1}{2}\\
&=\frac{1}{2}P[\mu'=\mu|\mu=0]+\frac{1}{2}P[\mu'=\mu|\mu=1]-\frac{1}{2}\\
&=\frac{1}{2}P[\eta'=\eta|\mu=0]+\frac{1}{2}P[\eta'\neq\eta|\mu=1]-\frac{1}{2}\\
&=\frac{1}{2}(\frac{1}{2}+\epsilon)+\frac{1}{2}\times\frac{1}{2}-\frac{1}{2}=\frac{\epsilon}{2}.
\end{aligned}
$$

■

Definition 2. *RKR-IND-CPA*: An updatable encryption scheme satisfies *RKR-IND-CPA* if all polynomial time adversaries have at most a negligible advantage to win the following 3-phase game.

Phase 1: The challenger runs the algorithms of the updatable encryption scheme to generate a public parameter *UGP*, a private parameter *usp*, a secret key *USK*. The challenger returns *UGP* to the adversary. Upon receiving it, the adversary returns a polynomial bounded number q to the challenger.

Phase 2: The challenger runs the algorithms of the updatable encryption scheme to generate a sequence of secret keys USK_1, USK_2,$\cdots$, USK_q started from *USK* and record the corresponding re-keys rk_1, rk_2,$\cdots$, rk_q. The challenger then allows the adversary to send a polynomial bounded number of queries. In each query, the adversary selects to send j where $j\in[1, q]$, or (i, m) where $i\in[0, q]$ and m is a message. Upon receiving j, the challenger returns rk_j. Upon receiving (i, m), if $i\in[1, q]$, the challenger generates an encryption C_i^{UP} of m by USK_i and returns C_i^{UP}; if i=0, the challenger generates an encryption C_0^{UP} of m by *USK* and returns C_0^{UP}.

Phase 3: The adversary generates two equal-length messages m_0^*, m_1^* and returns a challenge (m_0^*, m_1^*) to the challenger. The challenger selects one of the two messages by flipping a fair bit η, generates an encryption C^{UP} of m^*_η by *USK* and returns C^{UP} to the adversary.

Finally, the adversary returns a guess η'. If $\eta=\eta'$, the adversary wins the game. We define the adversary's advantage in the above game as $|P[\eta=\eta']-\frac{1}{2}|$.

Theorem 2. Our updatable encryption scheme satisfies *RKR-IND-CPA* based on the DBDH assumption.

Proof: Suppose there exists a polynomial time adversary $\mathcal{A}$ that can break *RKR-IND-CPA* of our updatable encryption scheme with advantage ϵ. We build an adversary $\mathcal{B}$ to break the DBDH assumption with advantage $\frac{\epsilon}{2}$. In the following, we just describe the difference with the proof of Theorem 1:

Similar with the proof of Theorem 1, the challenger of DBDH game gives $\mathcal{B}$ proper $(Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Z)$ based on flipping result of a fair bit μ.

In the setup phase, $\mathcal{B}$ computes $X=e(Y_2, Y_6)=e(g, h)^{y_1 y_2}$. $\mathcal{B}$ chooses a random number $\beta\in Z_p$ and computes g^β. $\mathcal{B}$ sets $UGP=X$ and $USK=(g^\beta, h^{\frac{y_1 y_2}{\beta}})$. Note that $\mathcal{B}$ does not have enough knowledge to compute $h^{\frac{y_1 y_2}{\beta}}$. However, as $\mathcal{B}$ does not need to show *USK* to $\mathcal{A}$, $\mathcal{A}$ will not be aware of this event. $\mathcal{B}$ then returns *UGP* to $\mathcal{A}$. After that, $\mathcal{B}$ receives q from $\mathcal{A}$.

In the query phase, $\mathcal{B}$ chooses a sequence of random numbers α_1, α_2,$\cdots$, $\alpha_q\in Z_p$. The distribution of the constructed sequence is identical with the sequence of re-keys rk_1, rk_2,$\cdots$, rk_q generated by the algorithms of the updatable encryption scheme. $\mathcal{B}$ then informs $\mathcal{A}$ to send queries. When receiving j, $\mathcal{B}$ returns α_j. When receiving (i, m), $\mathcal{B}$ chooses a random number $s\in Z_p$ and constructs an encryption C_i^{UP}:

$$C_i^{UP}=(C_{i1}, C_{i2})=(mX^s, (g^{\beta\alpha_1\alpha_2\cdots\alpha_i})^s) \text{ if } i\in[1, q]$$

$$C_i^{UP}=(C_{i1}, C_{i2})=(mX^s, (g^\beta)^s) \text{ if } i=0$$

$\mathcal{B}$ then returns C_i^{UP} to $\mathcal{A}$.

In the challenge phase, $\mathcal{B}$ constructs an encryption C^{UP}:

$$C^{UP}=(C_1, C_2)=(m^*_\eta Z, (Y_3)^\beta)$$

If μ=0, then $Z=e(g, h)^{y_1 y_2 y_3}$. Let $s=y_3$, then we have $C_1=m^*_\eta e(g, h)^{y_1 y_2 y_3}=m^*_\eta(e(g, h)^{y_1 y_2})^{y_3}=m^*_\eta X^s$ and $C_2=(Y_3)^\beta=(g^\beta)^{y_3}=(g^\beta)^s$. The distribution of the constructed encryption is identical to the valid ones generated by the algorithms of updatable encryption scheme with the secret key *USK*. If η=1, then $Z=e(g, h)^z$. We then have $C_1=m^*_\eta e(g, h)^z$. Since z is random, C_1 is actually a random element of G_T from the view of $\mathcal{A}$ and the constructed encryption reveals no information about m^*_η.

Finally, the guess phase is similar with the proof of Theorem 1. The overall advantage of $\mathcal{B}$ in the DBDH game is thus $\frac{\epsilon}{2}$.

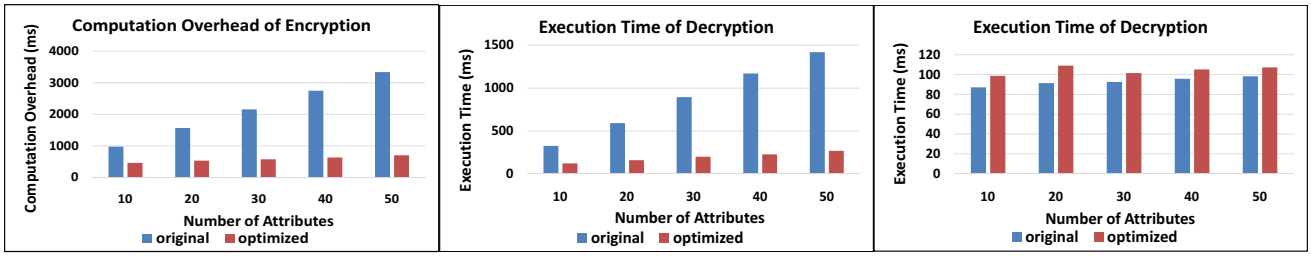
■

B. Security analysis of our system

Data privacy: In our system, participants follow double encryption paradigm to encrypt its product data before submission. As a result, the service provider has to decrypt two layers of encryptions to access the underlying product data. Considering a tagged product flows through the supply chain; the *RKR-IND-CPA* property of updatable encryption scheme prevents the service provider to decrypt the outside layers of submitted policy enforced encryptions, and the security property of CP-ABE scheme prevents the service provider to decrypt the inside layers of submitted policy enforced encryptions. Double encryption paradigm also resists collusion attack between the service provider and the key authority, since the successful decryption requires tag attribute credentials.

Item-level access control: As our access policy is defined over both role attributes `AND` tag attribute, it restricts that the potential authorized participants must possess the tag. The role attributes on the other hand are able to define a desired set of properties to specify authorized participants. The key authority is also excluded from accessing the product data as it does not manage the credentials of tag attributes.

Item-level revocation: We need to prove that a revoked participant with old tag attribute credentials cannot decrypt any encryption associated with a new tag attribute. Considering

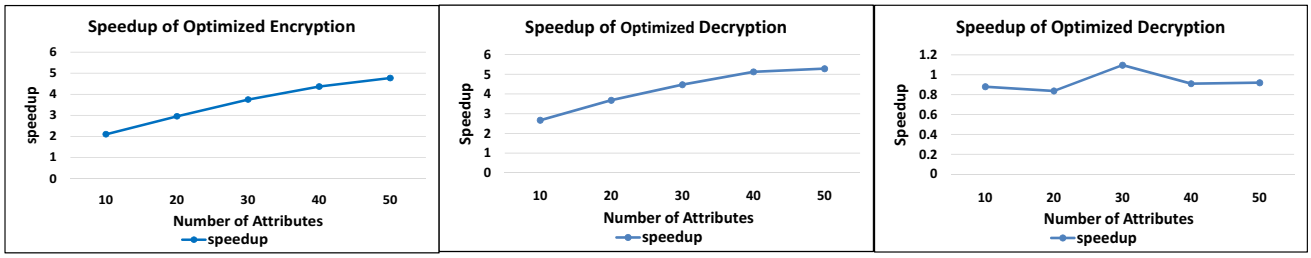


(a) Computation overhead (ms) of encryption operation

(b) Computation overhead (ms) of decryption operation with all AND policy

(c) Computation overhead (ms) of decryption operation with all OR policy

Fig. 4. Performance comparison between optimized encryption/decryption operations and encryption/decryption operations.



(a) Speedup ratio between optimized encryption operation and encryption operation

(b) Speedup ratio between optimized decryption operation and decryption operation with all AND policy

(c) Speedup ratio between optimized decryption operation and decryption operation with all OR policy

Fig. 5. Speedup ratio between optimized encryption/decryption operations and encryption/decryption operations.

a tagged product flows through the supply chain, the *RSKR-IND-CPA* property of updatable encryption scheme prevents each revoked participant with its old tag attribute credential to decrypt any submitted policy enforced encryption associated with a new tag attribute.

V. Performance Evaluation

We evaluate our system in four steps. We first investigate the performance of encrypting/decrypting product data for a tagged product at the participant side. We then compare the performance of our system against CP-ABE in achieving item-level access control and item-level privilege revocation. We next study the overall performance of our system through large-scale experiments. Finally, we evaluate whether our system can meet the resource requirements of commodity C1G2 RFID system. We use a java implementation [20] of CP-ABE and implement our updatable encryption scheme using a java version [17], [18] of the Pairing Based Cryptography (PBC) library [19].

Recall that our system consists of three types of entities: supply chain participant, service provider and key authority. Each entity has its own server to execute cryptographic operations. Besides, each supply chain participant owns readers to collect tag tokens, which are then sent to its server for further processing. The servers of the three types of entities are connected through network for communication. In our experiment, we adopt high-end workstations to conduct the computation tasks. The workstations are equipped with 16-core AMD Opteron Processor 6320 and 16GB RAM running on Ubuntu 13.10 OS.

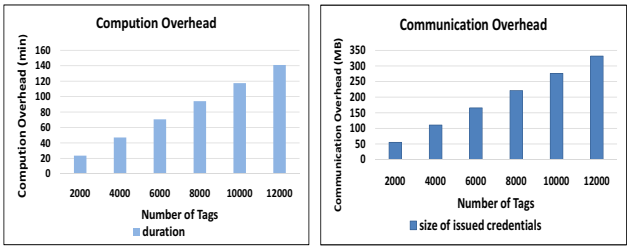
A. Basic evaluation

In our system, a participant needs to frequently encrypt its product data and decrypt other participants' product data. We thus evaluate the basic overhead of encrypting/decrypting a policy enforced encryption. Figure 4 shows the overhead of encryption/decryption with 108 KB data. We compare the optimized performance using multi-cores and the original performance using single-core.

Figure 4. (a) shows the encryption overhead as a function of the number of attributes contained in the associated policy. Our findings show that the overhead grows linearly with the number of attributes and use multi-cores can greatly increase the speed (as shown in figure 5. (a)). The reason is that the encryption operation consists of multiple independent sub tasks, which can be assigned to multi-cores for parallel execution.

On the other hand, the decryption overhead depends on the type of policies associated with the encryptions. Specifically, the policy of a policy enforced encryption regulates a minimum set of role attributes that a credential of role attributes need to be satisfied for successful decryption. Due to the design of CP-ABE, the decryption overhead grows linearly with the size of this set. Figure 4. (b)(c) show two computational extreme cases: all AND policy (all attributes in the policy are connected by AND) and all OR policy (all attributes in the policy are connected by OR) respectively.

In the all AND case, the decryption overhead grows linearly with the size of the policy (as shown in figure 4. (b)). The reason is that the minimum set is all the role attributes in the policy. Besides, using multi-cores can greatly increase the



(a) Computation overhead (min) as a function of tag number

(b) Communication overhead (MB) as a function of tag number

Fig. 6. Overhead of CP-ABE in achieving item-level data access control.

speed (as shown in figure 5. (b)). The reason is that processing each of the in-set role attributes raises several sub tasks and all these tasks are independent, which can be parallel executed by multi-cores. In the all `OR` case, the decryption overhead grows slowly with the size of the policy (as shown in figure 4. (c)). The reason is that the minimum set is any one of the role attributes in the policy. Besides, using multi-cores cannot increase the speed (as shown in figure 5. (c)). The reason is that processing one role attribute raises few sub tasks and the ability of multi-cores is not fully utilized.

B. Comparison of our system with CP-ABE

We now compare the performance of our system against CP-ABE in achieving item-level access control and item-level privilege revocation. All our comparison experiments fully utilize the 16 cores of our PC to improve the running speed.

Access control overhead: CP-ABE requires the key authority to issue credentials for each of the participants in item-level. We assume each participant is described by 20 role attributes. A CP-ABE credential thus corresponds to 21 attributes (20 role attributes and a tag attribute). Figure 6. (a) plots the computation overhead of the key authority to compute credentials of a batch of tagged products for one participant. The computation overhead grows linearly with the number of products. When a batch of 12000 tagged products flow through the supply chain, the key authority has to cost more than 2 hours to compute the credentials for each of the participants. Suppose the supply chain consists of 15 participants, the total computation delay can be as high as 30 hours.

Figure 6. (b) plots the communication overhead of the key authority to distribute the credentials of tagged products for one participant. Similar to the computation overhead, the traffic volume grows linearly with the number of products. The key authority needs to distribute 340 MB credentials for each participant when a batch of 12000 tagged products flow through the supply chain. Moreover, credentials are actually secret keys for decryption purposes. These credentials thus need to be distributed through secure channels, which incurs additional overhead for channel maintenance. As a large scale supply chain needs to handle millions of products, it poses heavy burdens on the key authority and renders such an approach inapplicable.

Instead, our system avoids fussy credential issuing task and leverages tags to locally distribute tag attribute credentials. By doing so, our system avoids the computation overhead and the communication overhead between the key authority and the participants. Instead, participants can immediately read the tag attribute credential of a product from the attached tag, greatly improving product handling efficiency.

TABLE II
REVOCATION COMPARISON IN HANDLING A BATCH OF 6000 TAGGED PRODUCTS

Scheme	Computation overhead	Communication overhead
CP-ABE	140 mins	340 MB
Our system	27 mins	6000×168 bits≈0.96 MB

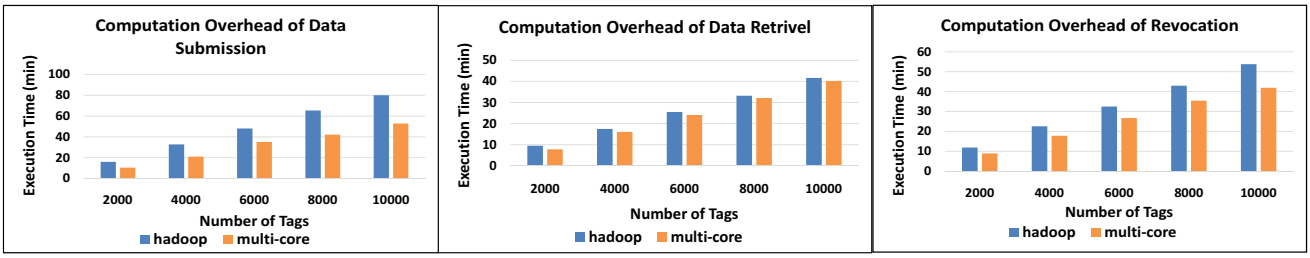
Revocation overhead: In a revocation operation, CP-ABE relies on the key authority to issue two types of credentials to each of the unrevoked participants in item-level. For a batch of tagged products and for a unrevoked participant, the computation overhead of the key authority doubles the computation overhead shown in Figure 6. (a). The unrevoked participant needs to acquire an old credential and an updated credential for each tagged product of the batch to access their product data. For the same reason, the communication overhead for credential distribution of the key authority also doubles the communication overhead shown in Figure 6. (b).

Instead, our system allows the participant who conduct the revocation operation to locally update the tag tokens of a batch of tagged products and delegate the service provider to update policy enforced encryptions for the batch. Both of the update operations are computation efficient. The participant also needs to send a short 168-bit re-key for each product of the batch to the service provider for it to update the encryptions, which is communication efficient. Table II compares the computation overhead and communication overhead of CP-ABE and our system to handle a batch of 6000 tagged products. As we can see, our system achieves obvious efficiency advantage.

C. Large-scale experiments

A real supply chain often processes a large amount of tagged products. To understand the performance of our system in this setting, we evaluate the overhead of our system through large-scale experiments. We test our experiments on two computing platforms: a single PC and a cluster of PCs. The first platform is exactly our 16 cores PC; while the second platform is a cluster of three PCs scheduled by hadoop infrastructure. We guarantee that the two platforms have the same hardware configuration (e.g., same number of CPUs).

We evaluate the three core operations of our system, namely data submission, data retrieval and privilege revocation. We generate random product data following normal distribution. Figure 8 shows the computation overhead incurred by the three operations when processing a large amount of tagged products. We vary the number of products from 2000 to 10000. In the data submission, we measure the time to encrypt policy enforced encryptions for all the products. The time can be as fast as about 50 minutes for a batch of 10000 products. In the data retrieval, we measure the time to decrypt policy enforced encryptions for all the products. The time can be as fast as about 40 minutes for a batch of 10000 products. In the privilege revocation, we measure the time to update the policy enforced encryptions for all the products. Also, The time can

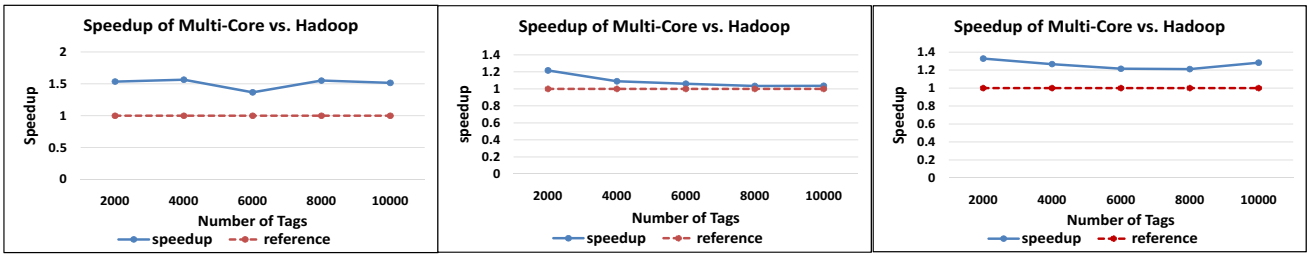


(a) Computation overhead (min) of data submission as a function of tag number

(b) Computation overhead (min) of data retrieval as a function of tag number

(c) Computation overhead (min) of data privilege revocation as a function of tag number

Fig. 7. Performance of our system in handling huge numbers of tagged products.



(a) Speedup of data submission as a function of tag number

(b) Speedup of data retrieval as a function of tag number

(c) Speedup of privilege revocation as a function of tag number

Fig. 8. Evaluation of our system on a multi-cores single PC and a cluster of PCs with hadoop.

be as fast as about 41 minutes for a batch of 10000 products. Overall, we believe the resulting latency is acceptable for real applications.

In the evaluation of the three operations, the first platform always performs better than the second platform. This means that the first platform assigns higher rate of computation resource to complete the three operations. On the other hand, the second platform leverages hadoop to support convenient task submission mode, robust data storage and extendibility, which consumes part of computation resource.

D. Implementation on commodity C1G2 RFID systems

We describe our implementation on commodity C1G2 RFID systems. Our system requires each tag to carry a tag token, which consists of a tag attribute and a secret key of the updatable encryption scheme. We select type f elliptic curve of PBC library to implement the updatable encryption scheme and the secret key is 496 bits long. We further set the tag attribute as a hash of the secret key so that it does not need to be explicitly stored in the tag. As a result, a tag token is only 496 bits long. When the secret key is updated in a revocation operation, the tag attribute is also changed. In this case, the participant can requests the service provider to use the new tag attribute to index all the submitted records.

An RFID reader can write/read tag tokens to/from an RFID tag with the C1G2 communication primitives. In particular, we use the Write command to write tag token into RFID tags, which allows the reader to write a 16-bit data block per operation. To transfer more data, the reader needs to divide the large trunk of data into several 16-bit blocks and write them via multiple Write operations. The Read command on the other hand supports the bulk data collection, which allows the reader to collect up to 512 bits per Read operation.

We use the Alien ALR 9900+ commodity RFID reader with the default settings (e.g., 30dBm transmission power) to interrogate commodity passive RFID tags. The data transfer program is developed based on the Alien RFID reader SDK codes. Our implementation only requires the C1G2 routine operations so we believe our design can also be implemented on other commodity RFID systems.

Current commodity RFID tags typically have different sizes of non-volatile memory which can be used to store the tag tokens. We test with two different types of widely used passive tags – ALN-9640 and AD-224 tags both with 512-bit user memory. As our tag token adopts compact 496-bit tag token, the user memory can accommodate the tag tokens. Our design does not require any modifications to the commodity passive tags or implement additional cryptographic functionality on the tags.

TABLE III
COMMUNICATION TIME OF TAG TOKEN TRANSFER

Types of passive tags	Writing time	Reading time
ALN-9640	509 ms	107 ms
AD-224	501 ms	100 ms

We focus on the communication overhead between the reader and the tags in tag token transfer. As the tag tokens are transferred using the C1G2 Write/Read primitives, the performance is largely dictated by the throughput of the commodity RFID system. Table III shows the communication overhead involved in the 496-bit tag token transfer. According

to the experiment results, it requires more time to write tag tokens into tags, because as mentioned the Write command only allows the reader to write a 16-bit data block per Write operation. To transfer the 496-bit tag token, the reader needs to first divide the tag token into several blocks and transfer them separately which takes longer time. In comparison, the Read operation takes less time since it only requires one Read operation to collect the whole 496-bit tag token.

VI. Related work

Besides the original CP-ABE scheme [9], there are also many other constructions of CP-ABE [26]–[30]. We briefly discuss the suitability of these schemes for our system. Many new CP-ABE schemes [27]–[30] extend the primary CP-ABE [9] from the aspects of security strength, flexibility and efficiency. Our system can immediately inherent these advantages by replacing the primary CP-ABE with these new CP-ABE schemes. Recently, Lewko *et.al* [26] propose a distributed version of CP-ABE (DCP-ABE) called DCP-ABE. In their scheme, multiple authorities could claim their own attributes and manage the corresponding credentials. On the other hand, one could jointly use the credentials across them for decryption purpose. This property enables DCP-ABE a good candidate to support item-level access control. However, DCP-ABE cannot be solely used to achieve item-level privilege revocation as the access policies of its encryptions cannot be updated without decryption.

Various applications in the RFID-enabled supply chain have been proposed [6]–[8], [11], [12], which heavily rely on a central service provider to enforce data privacy. Li *et al*. [6] propose to store tag data in a central database and share the data among different supply chain partners. Zanetti et al. [7] design a clone tag detection system, where a central server collects tracing data for tagged products when they move in the supply chain. Blass et al. [8] and Elkhiyaoui et al. [11] design several tag path authentication techniques, which rely on a central manager to issue path information for participants. Kerschbaum *et al*. [12] propose to establish trust relations leveraging RFID tags which requires a key authority to distribute secret keys. Different with our system, however, all these works do not provide any security mechanism against the central service provider and just assume that it is trustworthy.

VII. conclusion

We present a scalable data access control system for RFID-enabled supply chain. While many access control schemes have been proposed to protect product data, they are not scalable in the supply chain scenario involving a large number of RFID tags. We propose an item-level data access control mechanism that defines access policies based on both participant identity as well as tag credential. We further design a new updatable encryption scheme to achieve efficient privilege revocation. Experiment results based on the large-scale simulation and the real implementation on commodity RFID systems demonstrate the scalability and efficiency of our system.

VIII. ACKNOWLEDGEMENT

We acknowledge the support from Singapore MOE AcRF Tier 1 grant MOE2013-T1-002-005, NTU Nanyang Assistant Professorship (NAP) grant M4080738.020.

References

[1] “Data Sharing in the Pharmaceutical Supply Chain: A Series of Case Studies”, www.hcsupplychainresearch.org/WP/IBM/_whitepaper.pdf.
[2] G.M. Gaukler, R.W. Seifert, “Applications of RFID in Supply Chains”, in Proceedings of Supply Chain Design and Management, 2007.
[3] http://www.gtnexus.com/.
[4] “Epedigree - Wikipedia, the free encyclopedia”, http://en.wikipedia.org/wiki/Epedigree.
[5] “Pharma Logistics: Can RFID Heal Supply Chain Security?”, http://www.inboundlogistics.com/cms/article/pharma-logistics-can-rfid-heal-supply-chain-security/.
[6] Y. Li and X. Ding, “Protecting RFID Communications in Supply Chains”, in Proceedings of ACM ASIACCS, 2007.
[7] D. Zanetti, S. Capkun, and A. Juels, “Tailing RFID Tags for Clone Detection”, in Proceedings of NDSS, 2013.
[8] E. Blass, K. Elkhiyaoui and R. Molva, “Tracker: Security and Privacy for RFID-based Supply Chains”, in Proceedings of NDSS, 2011.
[9] J. Bethencourt, A. Sahai and B. Waters, “Ciphertext-Policy Attribute-Based Encryption”, in Proceedings of IEEE S&P, 2007.
[10] “Securing RFID Data for the Supply Chain”, www.verisign.com/static/028573.pdf.
[11] K. Elkhiyaoui, E. Blass and R. Molva, “CHECKER: on-site checking in RFID-based supply chains”, in Proceedings of ACM WiSec, 2012.
[12] F. Kerschbaum and A. Sorniotti, “RFID Based Supply Chain Partner Authentication and Key Agreement”, in Proceedings of ACM WiSec, 2009.
[13] R. Baden, A. Bender, N. Spring, B. Bhattacharjee and D. Starin, “Persona: An Online Social Network with User-Defined Privacy”, in Proceedings of ACM SIGCOMM, 2009.
[14] J. A. Akinyele, C. Lehmann, M. Green, M. Pagano, Z. Peterson, A. Rubin, “Self-Protecting Electronic Medical Records Using Attribute-Based Encryption”, in Proceedings of ACM CCS SPSM 2011.
[15] N. Santos, R. Rodrigues, K. P. Gummadi, S. Saroiu, “Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services”, in Proceedings of USENIX Security, 2012.
[16] M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography”, in Proceedings of EUROCRYPT, 1998.
[17] jPBC: Java Pairing Based Cryptography, http://gas.dia.unisa.it/projects/jpbc.
[18] A. De Caro, and V. Iovino, “jPBC: Java pairing based cryptography”, in Proceedings of IEEE ISCC, 2011.
[19] B. Lynn. “The pbc library”, http://crypto.stanford.edu/pbc/.
[20] http://junwei-wang.github.io/cpabe/.
[21] D. Boneh and X. Boyen, “Efficient Selective Identity-Based Encryption Without Random Oracles”, Journal of Cryptology, 2011.
[22] G. Ateniese, K. Fu, M. Green and S. Hohenberger, “Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage”, in Proceedings of NDSS, 2005.
[23] M. Green and G. Ateniese, “Identity-Based Proxy Re-encryption”, in Proceedings of ACNS, 2007.
[24] T. Isshiki, M. Nguyen and K. Tanaka, “Proxy Re-Encryption in a Stronger Security Model Extended from CT-RSA2012”, in Proceedings of CT-RSA, 2013.
[25] O.Goldreich, Foundations of Cryptography: Part 2, Cambridge University Press, 2004.
[26] A. Lewko and B. Waters, Decentralizing Attribute-Based Encryption, in Proceedings of EUROCRYPT, 2011.
[27] R. Ostrovsky, A. Sahai and B. Waters, Attribute-Based Encryption with Non-Monotonic Access Structures, in Proceedings of ACM CCS, 2007.
[28] B. Waters, ”Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization”, in Proceedings of PKC, 2011.
[29] T. Okamoto and K. Takashima, ”Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption”, in Proceedings of CRYPTO, 2007.
[30] A. Lewko and B. Waters, ”Unbounded HIBE and Attribute-Based Encryption”, in Proceedings of Eurocrypt, 2011.