Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

SECO: Secure and scalable data collaboration services in cloud computing

Xin Dong [a], Jiadi Yu [a,*], Yanmin Zhu [a], Yingying Chen [b], Yuan Luo [a], Minglu Li [a]

[a] Department of Computer Science and Engineering, Shanghai Jiao Tong University Shanghai, 200240, PR China
[b] Department of Electrical and Computer Engineering, Stevens Institute of Technology Hoboken, 07030, USA

ARTICLE INFO

Article history:
Received 3 April 2014
Received in revised form
21 November 2014
Accepted 12 January 2015
Available online 28 January 2015

Keywords:
Data collaboration
Data security
HIBE
One-to-many encryption
Cloud computing

ABSTRACT

Cloud storage services enable users to remotely store their data and eliminate excessive local installation of software and hardware. There is an increasing trend of outsourcing enterprise data to the cloud for efficient data storage and management. However, this introduces many new challenges toward data security. One critical issue is how to enable a secure data collaboration service including data access and update in cloud computing. A data collaboration service is to support the availability and consistency of the shared data among multi-users. In this paper, we propose a secure, efficient and scalable data collaboration scheme SECO. In SECO, we employ a multi-level hierarchical identity based encryption (HIBE) to guarantee data confidentiality against untrusted cloud. This paper is the first attempt to explore secure cloud data collaboration services that precludes information leakage and enables a one-to-many encryption paradigm, data writing operation and fine-grained access control simultaneously. Security analysis indicates that the SECO is semantically secure against adaptive chosen ciphertext attacks (IND-ID-CCA) in the random oracle model, and enforces fine-grained access control, collusion resistance and backward secrecy. Extensive performance analysis and experimental results show that SECO is highly efficient and has only low overhead on computation, communication and storage.

1. Introduction

Cloud computing (Armbrust et al., 2009), the long-held dream of computing as a utility, is rapidly evolving to revolutionize the way how data is stored/used. Cloud computing benefits data users in that it allows convenient access and use of storage resources offered by a cloud server provider (CSP). Challenges in security, however, posed by outsourcing data to the cloud, come along with benefits. Upon loss of physical possession of the outsourced data, users no longer control their data. But, CSPs may be untrusted and could monitor at will, lawfully or unlawfully, the data stored in the cloud and the communication between users and cloud. As a result, outsourcing users' data to the cloud initiates a series of problems about security and privacy. Examples of security breaches never stop showing up (Arrington, 2006; Wilson, 2008; Ren et al., 2012; Ateniese et al., 2007). Therefore, maintaining data availability and confidentiality becomes critical to

* *Corresponding author*. Tel./fax: +86 21 3420 5856.
E-mail address: jiadiyu@sjtu.edu.cn (J. Yu).
http://dx.doi.org/10.1016/j.cose.2015.01.003

enable wide deployment of CSP-based data service with high quality.

One important security issue is how to ensure secure data storage service when utilizing cloud services (Arora et al., 2013; De Capitani di Vimercati et al., 2010; Samarati and De Capitani di Vimercati, 2010). For instance, enterprises can outsource their data into the cloud and then enable their employees to access these data. However, cloud servers are untrusted and they may disclose the confidential information about an enterprise to their business competitors or even hide data leakage to maintain their reputations. In order to ensure data security, companies and enterprises usually have to encrypt the data before outsourcing it into the cloud. Recently, the notion of secure cloud storage services has been proposed in the content of ensuring remotely stored data under different systems and security models (Ateniese et al., 2007; Yu et al., 2010; Wang et al., 2010a; Dong et al., 2013). These existing works addressed secure cloud storage and data access issue either by introducing attribute-based encryption (ABE) (Goyal et al., 2006) for fine-grained access control (Wang et al., 2010b; Dong et al., 2014), or by utilizing owner-write-user-read mechanism (Wang et al., 2009) to achieve cryptography-based access control and only support coarse grained access control. ABE-based schemes are data-read sharing services, while owner-write-user-read mechanism is a one-to-one encryption paradigm meaning encrypted data can only be decrypted by a particular recipient. Consequently, existing solutions mainly focus on how to afford secure data access control (read) for cloud users. None of these works considers that multiple users operate (read/write) encrypted data collaboratively in cloud computing, i.e., data collaboration services.

A *Data Collaboration* service is to support the availability and consistency of the shared cloud data among multi-users. Let's consider a typical data collaboration service scenario: Alice, who is the boss of a company, pays a CSP for a secure data collaboration service, and assigns her two colleagues, Jack and Bob, to work collaboratively on a project. Alice first encrypts the project data and stores the encrypted data into the cloud. Then, Alice authorizes Jack and Bob to access the encrypted data so that they can modify the data. After modifying the data, Jack or Bob re-encrypts the data and sends it to the cloud. Within these three members, anyone who modifies the data, then determines the access privilege of the data. In total, three members work together and share data in a collaborative way. To avoid information leakage, the data have to be restrained within the reach of these three members. Thus the access policy of the above scenario is: authorized users can access the encrypted data while CSP and other unauthorized users know nothing about the data in data collaboration services.

To realize secure data collaboration services in cloud computing, we face the following challenges. Firstly, since a confidential data involves more than one recipient, the encryption paradigm should be one-to-many that indicates multiple recipients can decrypt the encrypted data. Secondly, authorized users have the privilege to operate the cloud data, so the encryption paradigm should support data writing operation. Thirdly, in order to ensure data security among users, the system should provide fine-grained access control to the users. To the best of our knowledge, there is no existing solution to tackle the problems of secure data collaboration services in cloud computing.

In this paper, we propose a scalable scheme (**SECO**) to enable secure cloud data collaboration with explicit dynamic data/users. For cloud data security, we employ a multi-level hierarchical identity-based encryption (HIBE) scheme, which contains a root private key generator (PKG), a series of lower-level PKGs and independent domains. The root PKG only generates private keys for lower-level PKGs, and lower-level PKGs in turn generate private keys for entities in their next level. A domain consists of a D-PKG and a number of individual users who cooperate to complete a project. During data collaboration, to achieve one-to-many encryption paradigm, a user in a domain encrypts data with the public parameters and multiple recipients' public keys so that only the intended domain recipients are able to decrypt the data. To support writing operation, every authorized user can encrypt the decrypted data after modifying (read/write) it, and then sends it into the cloud to share with other domain users. The data writing operation does not introduce security problems. To realize fine-grained access control, each authorized user which encrypts data can decide on the intended decryption recipients.

Specifically, the main contributions of this paper can be summarized as following three aspects:

- We propose a data collaboration service, SECO, which enables secure, efficient and scalable data collaboration in cloud computing, which realize one-to-many encryption paradigm, writing operation and fine-grained access control simultaneously without any information leakage. Our work is the first attempt to explore secure data collaboration in cloud computing.
- We prove that SECO is semantically secure against adaptive chosen ciphertext attacks (IND-ID-CCA) in the random oracle model under the Bilinear Diffie-Hellman assumption (Boneh and Franklin, 2001), and SECO also enforce collusion resistance and backward secrecy for cloud data collaboration services.
- We have conducted extensive theoretical analysis and real experiments to evaluate the performance of SECO. The result indicates that SECO introduces low overhead on computation, communication and storage while improves the effectiveness and efficiency.

The remainder of this paper is organized as follows: Section related work discusses related works; Section 3 introduces the system model, threat model and our design goals; Section 4 presents the detail design of SECO; Section 5 provides the security definition and security proof of SECO; Sections 6 and 7 analyze the theoretical and experimental performance of SECO, respectively; finally, Section 8 concludes the whole paper.

2. Related work

Identity-based encryption (IBE) is an encryption choice in cloud computing (Li et al., 2013a; Guo et al., 2013). The concept

of IBE is proposed by Shamir (1985), and the first fully functional IBE schemes are described by Boneh and Franklin (2001) and Cocks (2001). In IBE, the public key for a unique user can be set to any value (such as one's identity) and the corresponding private key is generated by a trusted third party called private key generator (PKG). Relatively speaking, the IBE scheme is a public key cryptosystem (PKC) and can eliminate the search for recipient's public key. To reduce the workload on the PKG, Horwitz and Lynn (2002) introduced a HIBE scheme with collusion-resistance. Gentry and Silverberg (2002) presented a HIBE scheme with total collusion resistance and chosen ciphertext security (CCA) in the random oracle model. Later on, Boneh et al. (2005) introduced an efficient HIBE scheme with selective-ID security without random oracle model under BDH assumption. However, these HIBE schemes are all one-to-one encryption paradigms.

An attribute-based encryption (ABE) system is actually a simplified IBE system, with only one attribute in the system. ABE was first proposed by Sahai and Waters (2005). In an ABE scheme, the sender encrypts the message with a set of attributes and specifies a number d. The recipient who has at least d attributes of the given attributes can decrypt the encrypted message. Based on these, Goyal et al. (2006) proposed a fine-grained data access control ABE scheme that supports any monotonic access structure, i.e., AND, OR, or other threshold gates. Later on, Ostrovsky et al. (2007) proposed an enhanced scheme that supports non-monotonic access structure which includes NOT gate that was not allowed in Goyal et al. (2006). There are two classes of ABE named key-policy attribute-based encryption (KP-ABE) and ciphertext policy attribute-based encryption (CP-ABE). In KP-ABE (Goyal et al., 2006), the access structure is used to encrypt the secret key, while the attributes are used to describe the ciphertext. CP-ABE was first introduced by Bethencourt et al. (2007). In a CP-ABE scheme, the access structure is used to encrypt the ciphertext and the secret key is generated based on an attribute set. Thus, the roles of the secret key and the ciphertext in CP-ABE are opposite to what they are in KP-ABE. ABE is a one-to-many encryption paradigm. However, it is not suitable for data collaboration services due to the key management.

Identity-based broadcast encryption is also a one-to-many encryption paradigm. The concept of broadcast encryption (BE) was first proposed by Fiat and Naor (1994). In BE schemes, a broadcast center encrypts messages and broadcasts them to a group of authorized users who are listening on a broadcast channel. Moreover, Mu et al. (2003) is the first to introduce the concept called "Identity-Based Broadcasting Encryption", which can be applied to dynamic key management in secure broadcasting. Later on, Baek et al. (2005) constructs an efficient "multi-receiver identity-based encryption scheme", which only needs one pairing computation to encrypt a single message for n receivers. Based on these, Delerablée (2007) describes an identity-based broadcast encryption with constant size ciphertexts and private keys. However, in these identity-based broadcasting encryption schemes, only the broadcast center can encrypt messages, and each authorized user just reads the message. That is to say, identity-based broadcasting encryption schemes cannot support data writing operation in data collaboration services. Moreover, identity-based broadcasting encryption schemes cannot achieve fine-grained access control in a group of authorized users.

Functional encryption is an emerging paradigm for public-key encryption that enables fine-grained control access to encrypted data (Agrawal et al., 2013). It extends several precious notions, mostly notably IBE encryption system, and provides the ability to generate and release secret keys associated with a keyword that can decrypt only those documents that contain the keyword. More generally, functional encryption allows the owner of a "master" secret key to release restricted secret keys that reveal a specific function of encrypted data. Based on these, Goldwasser et al. (2014) introduces the problem of multi-input functional encryption, where a secret key sk_f can correspond to an n-ary function f that takes multiple ciphertexts as input. Later on, Boneh et al. (2013) develops an approach for designing function-private identity-based encryption schemes. The authors first put forward a new notion, function privacy, in IBE encryption and functional encryption. In addition to function privacy, Boneh et al. (2013) proposed the first public-key searchable encryption scheme that are provably keyword private. In their schemes, a search key sk_w enables to identify encryptions of an underlying keyword w, while not revealing any additional information about keyword w. Therefore, the keyword w is sufficiently unpredictable. Functional encryption is also a one-to-many encryption paradigm. However, functional encryption also cannot support data writing operation in data collaboration services.

Furthermore, existing works can be found in the areas of secure outsourced data storage and sharing services. Adya et al. (2002) used symmetric keys to encrypt data and provided a secure, scalable data system that logically functions as a centralized data server but is physically distributed among a set of untrusted servers. However, every user used their public key to encrypt the symmetric keys and thus bring high overhead on key management. In Kallahalla et al. (2003), the authors proposed a cryptographic data system and used verify and sign keys to determine whether or not a user can read or write data respectively. Since the key generation procedure is proportional to the total number of data-groups, the above schemes are not suitable for the case of data collaboration in cloud computing, in which the number of data-groups could be enormous. In addition, the above schemes are one-to-one encryption paradigms and only support coarse-grained access control.

Goh et al. (2003) proposed SIRIUS that adopted a complicated structure and provided end-to-end security. However, the complexity of the scheme depends on each meta data size and thus is not scalable. Wang et al. (2009) proposed a mechanism in owner-write-users-read applications that assigned every data block with a different key to achieve flexible cryptography based access control. However, the users only can read the data but not write data, and thus are not suitable for data collaboration in cloud computing. Li et al. (2013b) proposed a novel patient-centric framework and a suit of mechanisms for data access control to personal health records (PHRs) stored in semi-trusted servers. To achieve fine-grained and scalable data access control for PHRs, they leverage ABE techniques to encrypt each patient's PHR file. However, the scheme is a data sharing services and can not support data writing operation in the stored PHR files. Wang

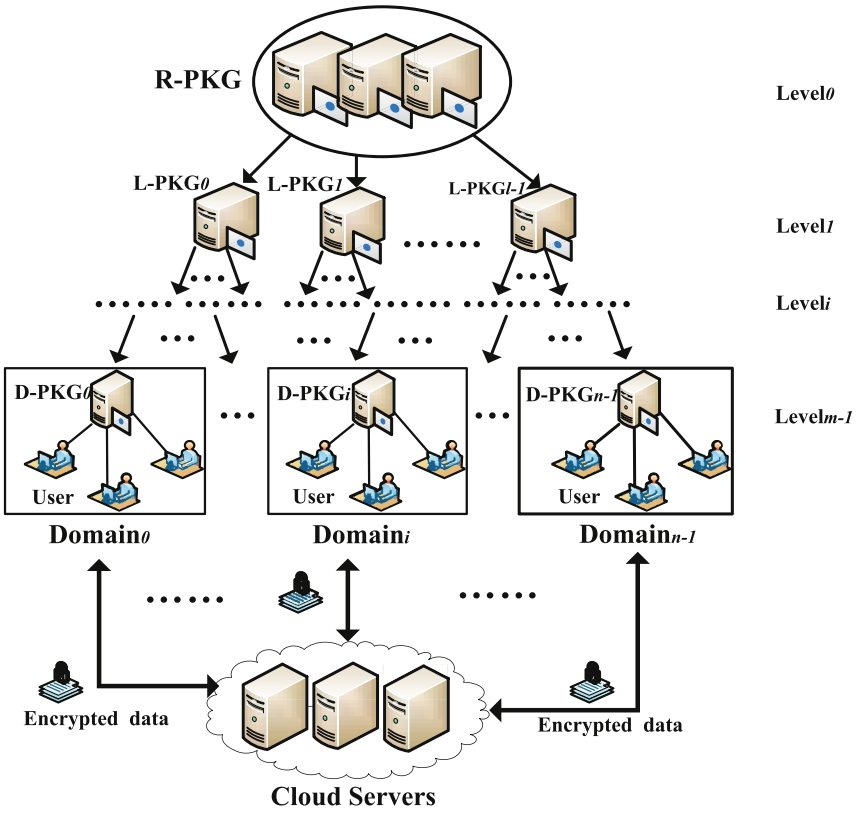

Fig. 1 – System model.

et al. (2012) proposed a flexible distributed storage integrity auditing mechanism, utilizing the homomorphic token and distributed erasure-coded data. However, the scheme also cannot support data writing operation and the complexity of the homomorphic encryption is high. Moreover, the above schemes cannot support data writing operation and are not suitable in data collaboration services.

3. Problem statement

3.1. System model

Generally, a cloud data collaboration system has five different parties in network: *Cloud Server* provides high-quality services utilizing a number of servers with significant storage space and computation power; *Root Private Key Generator* (R-PKG) possesses a master key and generates corresponding private keys for lower-level PKGs; *Level Private Key Generators* (L-PKGs) request private keys from upper-level PKGs and generate private keys for lower-level PKGs; *Domain Private Key Generators* (D-PKGs) request private keys from upper-level PKGs and generate private keys for their domain entities; *Users* cooperate with each other to complete a project, receive their private keys from D-PKG and store their data in the Cloud Server.

Fig. 1 depicts the system model, which is characterized by a multi-level HIBE scheme. From the system model, we can see that it consists of a R-PKG, a series of L-PKGs, D-PKGs and individual users. In this hierarchical architecture, the R-PKG generates system public parameters for all system entities and private keys for lower-level L-PKGs. Then, L-PKG in turn generates private keys for the entities in the next level. L-PKGs share the workload of private key generation and identity authentication for R-PKG. Thus, secret key transmission and authentication can be achieved locally. A domain consists of a D-PKG and a number of individual users who cooperate to complete a project. In each domain, the D-PKG keep a user list UL_{dom} which records public keys of all the valid users in the domain. The D-PKG will send the latest domain user list UL_{dom} to all valid users in a domain. All entities in a domain store their data into a set of cloud servers that are running in a cooperated and distributed manner. Users use their keys to decrypt the data stored in the Cloud Server. All entities in the domain can interact with the Cloud Server to access (read, write, update, etc.) the stored data dynamically. Furthermore, PKGs and users do not have to be online all the time, whereas the Cloud Server is always online.

3.2. Threat model

The adversary model considers most threats toward cloud data confidentiality. In the system model, Cloud Server is semi-trusted. Namely, it behaves properly most of the time, but for some benefits the Cloud Server might try to find out as much secret information as possible. In fact, there are several types of threats: Both inner threats (CSP and users who might obtain the unauthorized data) and outer threats (external adversaries beyond the domain of this system, e.g., unauthorized attackers) might be present; Attacks can either be active (unauthorized users who may inject malicious data into

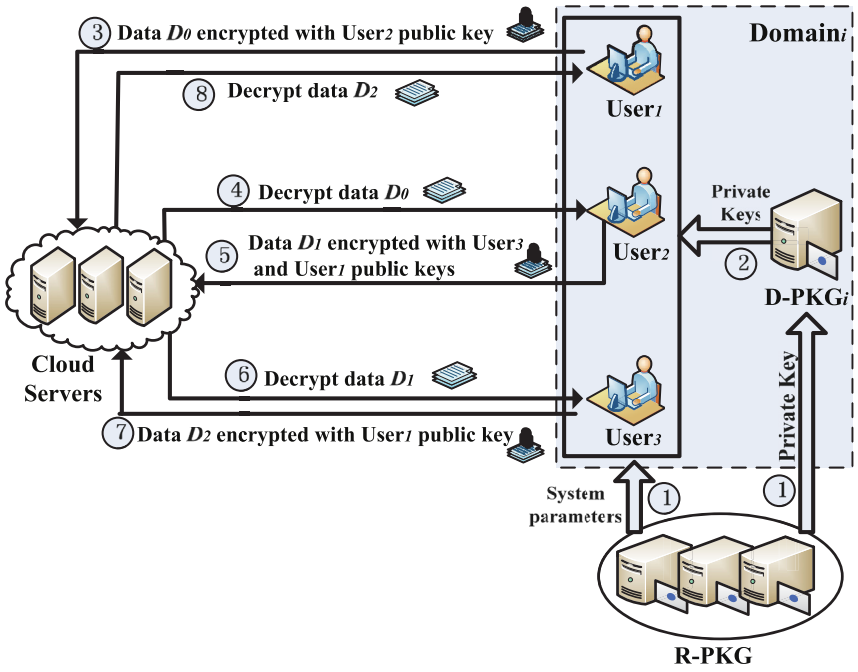

Fig. 2 – A simplified workflow of SECO in a domain.

the cloud) or be passive (unauthorized users eavesdropping on conversations between users and the cloud); For the purpose of harvesting data contents, CSP and users may collude and try to access unauthorized data.

For the purpose of secure data collaboration in cloud computing, the main goal of this paper is to protect the contents of domain data from being learned by the cloud and attackers, including inner intruders and unauthorized outer users. All these attacks can be active or passive. With respect to data access control in the cloud, we have the following requirements: 1) ***Fine-grained access control***: Each user should only access the data he is allowed and should not access the data he is not authorized to; 2) ***Collusion resistance***: As described in the adversary model, users cannot collude and share their secret key to access the data they are not allowed; 3) ***Backward secrecy***: Users should not access the decrypted domain data after they have been revoked from the domain. Note that, in the adversary model, the communication channels between users and Cloud Server are secured under existing protocols, such as SSL.

4. The design of SECO

In order to achieve secure cloud data collaboration, we propose a multi-level HIBE scheme SECO. SECO realizes a one-to-many encryption paradigm in which encrypted domain data can be decrypted by multiple authorized users, writing operation and fine-grained access control can be done simultaneously without any information leakage.

4.1. *Overview*

SECO employs a multi-level hierarchical architecture to embody the users' role in data collaboration in cloud computing. In SECO, R-PKG generates private keys for lower-level PKGs, and D-PKGs request private keys from upper-level PKGs. In a domain, a user encrypts data with multiple recipients' public keys and stores it into the Cloud Server. So only those intended recipients and the D-PKG can decrypt the data using their own secret keys. A user only takes public keys of the recipients and system parameters as inputs to encrypt data. Any other users outside the recipient list cannot obtain any data information even if all of them collude. Therefore, users in the same domain can cooperate to complete work without worrying about their data security.

SECO elegantly integrates five randomized algorithms: ***Root setup***, ***Lower-level setup***, ***Key generation***, ***Encryption***, ***Decryption*** to achieve secure cloud data collaboration. Fig. 2 describes a simplified workflow of SECO in a domain. At the initialization phase, R-PKG uses the *Root setup* algorithm to generate systems parameters for all entities. L-PKGs and D-PKGs then use *Lower-level setup* algorithm to pick some seeds for themselves. By the *Key generation* algorithm, A PKG generates private keys for all his children by using the system parameters and his private key. Suppose there are three users to work collaboratively in Domain_i. User_1 has confidential data D_0 and need User_2 and User_3 to operate it. User_1 first uses the *Encryption* algorithm to encrypt data D_0 with User_2 and User_3 public keys and stores it to the cloud. So that User_2 and User_3 can access data D_0. User_2 downloads data D_0 and uses the *Decryption* algorithm to decrypt it. After modifying D_0, User_2 renames it to data D_1 and uses *Encryption* algorithm to encrypt with User_1 and User_3 public keys. User_3 now can decrypt data D_1 and makes a final modification. User_3 encrypts the final version D_2 with User_1 public key and stores it. In the end, User_1 uses his private key to obtain the final data D_2. Thus, User_1, User_2 and User_3 modify the confidential data collaboratively without leaking any information to unauthorized users. In the following subsections, we elaborate the design details. Table 1 shows the symbols and their meanings as used in SECO.

4.2. Preliminaries

We give some related definitions and assumptions similar to those given in Boneh and Franklin (2001) and Gentry and Silverberg (2002), which are used in SECO.

Bilinear Diffie-Hellman (BDH) parameter generator: As in Gentry and Silverberg (2002), a randomized algorithm $\mathcal{IG}$ is a BDH parameter generator which takes a security parameter $K>0$ as input, and outputs the description of two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of the prime order q and a bilinear map $\hat{e}:\mathbb{G}_1\times\mathbb{G}_1\to\mathbb{G}_2$ in polynomial time.

Bilinear map: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order q, and g_1 is the generator of group $\mathbb{G}_1$. $\hat{e}$ is a bilinear map if $\hat{e}:\mathbb{G}_1\times\mathbb{G}_1\to\mathbb{G}_2$ satisfies the following properties:

- Bilinearity: for all $Q,R,S\in\mathbb{G}_1$ and $a,b\in\mathbb{Z}_q$ where $\mathbb{Z}_q=\{0,1,2,\ldots q-1\}$, have $\hat{e}(aQ,bR)=\hat{e}(bQ,aR)=\hat{e}(Q,R)^{ab}$, $\hat{e}(Q+R,S)=\hat{e}(Q,S)\hat{e}(R,S)$ and $\hat{e}(Q,R+S)=\hat{e}(Q,R)\hat{e}(Q,S)$.
- Computability: for any $Q,R\in\mathbb{G}_1$, there is a polynomial time algorithm to compute $\hat{e}(Q,R)\in\mathbb{G}_2$.
- Non-degeneracy: $\hat{e}(g_1,g_1)\neq 1$.

BDH problem: Randomly choose P as well as aP, bP and cP where $P\in\mathbb{G}_1$ and $a,b,c\in\mathbb{Z}_q$, compute $\hat{e}(P,P)^{abc}$.

BDH assumption: As in Gentry and Silverberg (2002), the advantage $Adv_{\mathcal{IG}}(\mathcal{B})$that an algorithm $\mathcal{B}$ has in solving the BDH problem is defined to be the probability that the algorithm $\mathcal{B}$ takes $\mathbb{G}_1,\mathbb{G}_2,\hat{e},P,aP,bP,cP$ as inputs and outputs $\hat{e}(P,P)^{abc}$, where Break is the output of BDH parameter generator $\mathcal{IG}$ for large security parameter $K>0$, P is a random generator of group $\mathbb{G}_1$, and a,b,c are random elements of $\mathbb{Z}_q$. The BDH assumption is that $Adv_{\mathcal{IG}}(\mathcal{B})$ is negligible for all efficient algorithm $\mathcal{B}$.

4.3. Construction of SECO

In this section, we construct SECO using the bilinear map. We first introduce the form of keys. Then, the detailed algorithms of SECO are presented.

Table 1 – Symbols and their meanings.

Symbols	Meanings
M	Message
C	Ciphertext
S_0	Identity element of group $\mathbb{G}_1$
t	The level of a entity
dom	The level of each D-PKG
UL_{dom}	The user list in a domain
E^i_{dom+1}	A domain user at $Level_{dom+1}$
$(ID-tuple_{dom}, ID^i_{dom+1})$	User E^i_{dom+1}'s public key (ID-tuple)
$s_0, s_{dom}, s^i_{dom+1}$	Master key for R-PKG, D-PKG and User E^i_{dom+1}
SK_{dom}, SK^i_{dom+1}	Private key for D-PKG and User E^i_{dom+1}
P_0,Q_0	System parameters generated by R-PKG
P_{dom},Q_{dom}	D-PKG's parameters
P^i_{dom+1}, Q^i_{dom+1}	User E^i_{dom+1}'s parameters
H_1,H_2,H_3,H_4	Hash function, example SHA-1

Let $Level_t$ be the set of entities at level t, and $Level_0=\{\text{R-PKG}\}$. In SECO, Each L-PKG and D-PKG has two secret keys: a *private key* and a *master key*. The private key is obtained from the upper-level PKG while the master key is a random seed picked by the PKG itself. A D-PKG manages a number of domain users. In a domain, the D-PKG uses the two keys to generate private keys for all users in this domain. In this hierarchy, each L-PKG, D-PKG and user has a primitive ID, which is an arbitrary string, such as ID number and email address. A user's public key is an ID-tuple consisting of his ancestor L-PKG's ID, D-PKG's ID and his own ID, i.e., $(ID_1,\ldots,ID_{m-1})$, where m is the depth of the hierarchy. For example, in a two-level hierarchy, Bob is a D-PKG which requests the private key from R-PKG and generates the private key for Alice which is a user in Bob's domain. Suppose the email addresses are used as their IDs, then the public keys of Bob and Alice are ("Bob@email") and ("Bob@email||Alice@email", "||" denotes string concatenation), respectively. In addition, the R-PKG also publishes several system parameters used to encrypt and decrypt the cloud data.

SECO is specified by the following five randomized algorithms.

Root Setup: Let K be the security parameter used by a BDH parameter generator $\mathcal{IG}$. The R-PKG takes K as input, and outputs *params* (system parameters) and a root master key. The system parameters which contain the description of plaintext space $\mathcal{M}$, ciphertext space $\mathbb{C}$ and some other parameters are published, while the root master key is only known to the R-PKG.

The R-PKG takes as input a security parameter K and runs the BDH parameter generator $\mathcal{IG}$ to generate two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order q. It generates a bilinear map $\hat{e}:\mathbb{G}_1\times\mathbb{G}_1\to\mathbb{G}_2$ which has the properties of bilinearity, computability and non-degeneracy. The R-PKG then picks an arbitrary generator $P_0\in\mathbb{G}_1$ and a seed $s_0\in\mathbb{Z}_q$ randomly, where $\mathbb{Z}_q=\{0,1,2,\ldots q-1\}$, and it sets $Q_0=s_0P_0$. Finally, the R-PKG defines four cryptographic hash functions $H_1:\{0,1\}^*\to\mathbb{G}_1$, $H_2:\mathbb{G}_2\to\{0,1\}^n$, $H_3:\{0,1\}^n\times\{0,1\}^n\to\mathbb{G}_1$ and H_4:$\{0,1\}^n\to\{0,1\}^n$ for some n, and the four hash functions will be treated as random oracles.

The plaintext space is $\mathcal{M}=\{0,1\}^n$, while the ciphertext space is $\mathbb{C}=\mathbb{G}_1^{N+L}\times\{0,1\}^{2n}$ where N is the number of the intended recipients and L is the level of the intended recipients. The parameters of the system are $params=<\mathbb{G}_1,\mathbb{G}_2,\hat{e},P_0,Q_0,H_1,H_2,H_3,H_4>$. The master key of R-PKG is $s_0\in\mathbb{Z}_q$.

Lower-level setup: Each PKG $E_t\in Level_t$ obtains the system parameters (*params*) from the R-PKG. Each PKG randomly picks a $s_t\in\mathbb{Z}_q$ as his master key which will be used to issue private keys to his children. Except for s_t, each PKG is not permitted to generate any other parameters.

Key generation: A PKG (whether the root one or a lower-level one) uses its private key (and any other secret information) and system parameters to generate private keys for all of his children. The private keys of domain users are generated by the D-PKG in the same domain. Let S_0 be the identity element of group $\mathbb{G}_1$.

For each PKG $E_t\in Level_t$ (L-PKG or D-PKG) with ID-tuple $(ID_1,\ldots,ID_t)$, where $(ID_1,\ldots,ID_s)$ for $1\le s<t$ is the ID-tuple of E_t's ancestor at $Level_s$, E_t's parent generates the private keys SK_t for each E_t. It first calculates $P_t=H_1(ID_1,\ldots,ID_t)\in\mathbb{G}_1$; then E_t's parent computes private key for E_t as:

$SK_t = S_{t-1} + s_{t-1}P_t = \Sigma_{j=1}^{t} s_{j-1}P_j$

and sends E_t the value $Q_j = s_jP_0$ for $1 \le j \le t-1$.

For a D-PKG $E_{dom} \in Level_{dom}$ in domain A, it there is a polynomial time algorithm s_{dom} and a private key SK_{dom}. Private key SK_{dom} is used to decrypt all domain data stored in the Cloud Server. Each D-PKG uses his private key SK_{dom} and master key s_{dom} to generate private keys for all users belonging to this domain.

For each user in domain A whose D-PKG/parent is E_{dom}, the ID-tuple for user E^i_{dom+1} is (ID-tuple$_{dom}$, ID^i_{dom+1}). Therefore, the level of these users in domain A is $dom+1$. E^i_{dom+1} randomly picks an element $s^i_{dom+1} \in \mathbb{Z}_q$ as his master key. E_{dom} generates the private key SK^i_{dom+1} for E^i_{dom+1}.

For each user E^i_{dom+1}, the D-PKG E_{dom} first calculates $P^i_{dom+1} = H_1(ID-tuple_{dom}, ID^i_{dom+1}) \in \mathbb{G}_1$; then it computes private key for E^i_{dom+1} as:

$SK_i = SK_{dom} + s_{dom}P^i_{dom+1} = \Sigma_{j=1}^{dom} s_{j-1}P_j + s_{dom}P^i_{dom+1}$

and sends to E^i_{dom+1} the value Q_j for $1 \le j \le dom$ and Q^i_{dom+1} which $Q^i_{dom+1} = s^i_{dom+1}P_0$. E^i_{dom+1} has two secret keys: a master key s^i_{dom+1} and a private key SK^i_{dom+1}. E^i_{dom+1} uses s^i_{dom+1} and SK^i_{dom+1} to decrypt the authorized data in the Cloud Server. Each D-PKG has a secret just like the root PKG. In addition, the D-PKGs need not always use the same s_{dom} for each private key extraction. That is to say, s_{dom} could be generated randomly for each of the D-PKG's children. It is worth noticing that E^i_{dom+1} public key (ID-tuple$_{dom}$, ID^i_{dom+1}) is equal to (ID_1, …,ID_{dom}, ID^i_{dom+1}) where the D-PKG $E_{dom} \in Level_{dom}$ is the parent of E^i_{dom+1}.

Encryption: When a user wants to encrypt a file for N recipients in domain A, the data D is encrypted in the following form: *Encrypt* (*paremters*, $ID-tuple_1$,…, $ID-tuple_N$, M), where parameters are the system public parameters, N is the number of the intended recipients, $ID-tuple_1$,…, $ID-tuple_N$ are the public key (ID-tuples) of N recipients E^1_{dom+1}, …, E^N_{dom+1}, respectively, and M is the data. The user inputs system parameters *params*, plaintext $M \in \mathcal{M}$ and the ID-tuples of the N intended data recipients, and then calculates a ciphertext $C \in \mathbb{C}$. After modifying data M, the user encrypts it with N recipients' ID-tuple (ID-tuple$_{dom}$, ID^i_{dom+1}) for $1 \le i \le N$ in the same domain.

The user first calculates $P^i_{dom+1} = H_1(ID-tuple_{dom}, ID^i_{dom+1}) \in \mathbb{G}_1$ for every $1 \le i \le N$ and $P_t = H_1(ID_1,…,ID_t)$ for $1 \le t \le dom$. Here P^i_{dom+1} means the hash value for the i-th recipient at $Level_{dom+1}$. Then the user picks a random $\sigma \in \{0,1\}^n$ and sets $r = H_3(\sigma, M)$. Therefore, the ciphertext is set as:

$$C = \left[\left\{rP^i_{dom+1}\right\}, \{rP_t\}, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma)\right]$$

for $1 \le i \le N$ and $0 \le t \le dom$ where $g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2$. The user encrypts the data M with N intended recipients in the same domain, and sends the ciphertext C to the Cloud Server.

Decryption: A user or D-PKG in domain A inputs system parameters *params*, ciphertext $C \in \mathbb{C}$, and its private key SK, and then recovers the data $M \in \mathcal{M}$. The D-PKG can decrypt all the encrypted data belonging to the domain, whereas the users only can decrypt the authorized data.

Given $C = [\{U^i_{dom+1}\}, \{U_t\}, V, W]$ be the ciphertext encrypted using the N recipients' ID-tuple (ID-tuple$_{dom}$,ID_i), for $i \in (1,2,…,N)$. Here $U^i_{dom+1} = rP^i_{dom+1}$, $U_t = rP_t$, $V = \sigma \oplus H_2(g^r)$ and $W = M \oplus H_4(\sigma)$. If $(U_0, U_1, U_2, …, U_{dom}) \notin \mathbb{G}_1^{dom+1}$, E_{dom} rejects this ciphertext. To decrypt C, the D-PKG E_{dom} computes

$$V \oplus H_2\left(\hat{e}(U_0, SK_{dom}) \Big/ \prod_{j=2}^{dom} \hat{e}\left(Q_{j-1}, U_j\right)\right).$$

We observe that:

$$\begin{aligned}
&V \oplus H_2\left(\hat{e}(U_0, SK_{dom}) \Big/ \prod_{j=2}^{dom} \hat{e}\left(Q_{j-1}, U_j\right)\right)\\
&= V \oplus H_2\left(\hat{e}\left(rP_0, \Sigma_{i=1}^{dom} s_{i-1}P_i\right) \Big/ \prod_{j=2}^{dom} \hat{e}\left(s_{j-1}P_0, rP_j\right)\right)\\
&= V \oplus H_2\left(\hat{e}(rP_0, s_0P_1)\hat{e}\left(rP_0, \Sigma_{i=2}^{dom} s_{i-1}P_i\right) \Big/ \hat{e}\left(rP_0, \Sigma_{j=2}^{dom} s_{j-1}P_j\right)\right)\\
&= V \oplus H_2\left(\hat{e}(Q_0, P_1)^r\right) = \sigma.
\end{aligned}$$

After calculating the value of σ, E_{dom} then computes $W \oplus H_4(\sigma) = M$.

Given the ciphertext $C = [\{U^i_{dom+1}\}, \{U_t\}, V, W]$ to each intended recipient E^i_{dom+1} of $i \in \{1,2,…,N\}$. If $(U_0, U_1, U_2, …, U_{dom}) \notin \mathbb{G}_1^{dom+1}$, E^i_{dom+1} rejects this ciphertext. To decrypt C, the recipient E^i_{dom+1} executes the following setups:

- computes $P^i_{dom+1} = H_1(ID-tuple_{dom}, ID^i_{dom+1})$;
- computes $V \oplus H_2(\hat{e}(U_0, SK^i_{dom+1}) / \prod_{j=2}^{dom} \hat{e}(Q_{j-1}, U_j)\hat{e}(Q_{dom}, U^i_{dom+1}))$ to recover σ;
- computes $W \oplus H_4(\sigma) = M$.
- sets $r = H_3(\sigma, M)$, tests that $U^i_{dom+1} = rP^i_{dom+1}$. If not, rejects the ciphertext. Otherwise, outputs M as the decryption of C.

Observe that:

$$\begin{aligned}
&V \oplus H_2\left(\hat{e}\left(U_0, SK^i_{dom+1}\right) \Big/ \prod_{j=2}^{dom} \hat{e}\left(Q_{j-1}, U_j\right)\hat{e}\left(Q_{dom}, U^i_{dom+1}\right)\right)\\
&= V \oplus H_2\left(\hat{e}\left(rP_0, SK_{dom} + s_{dom}P^i_{dom+1}\right) \Big/ \prod_{j=2}^{dom} \hat{e}\left(s_{j-1}P_0, rP_j\right)\hat{e}\left(s_{dom}P_0, rP^i_{dom+1}\right)\right)\\
&= V \oplus H_2\left(\hat{e}(rP_0, s_0P_1)\hat{e}\left(rP_0, \Sigma_{j=2}^{dom} s_{j-1}P_j\right)\hat{e}\left(rP_0, s_{dom}P^i_{dom+1}\right) \Big/ \hat{e}\left(rP_0, \Sigma_{j=2}^{dom} s_{j-1}P_j\right)\hat{e}\left(s_{dom}P_0, rP^i_{dom+1}\right)\right)\\
&= V \oplus H_2\left(\hat{e}(s_0P_0, P_1)^r\right) = \sigma.
\end{aligned}$$

The domain users cooperate to complete a project and store their project data into the Cloud Server. The domain PKG can decrypt all domain data while any user in this domain only can access the data that he is allowed.

4.4. Dynamic operations

In this section, we present the detail dynamic data and user operation processes in SECO. Since domain users do not physically possess their data but store them into the Cloud Server instead, the dynamic data and user operations are quite challenging. When SECO deals with these dynamic requests, it needs to satisfy the following requirements: firstly, secret keys cannot be disclosed to the Cloud Server; secondly, in order to manage keys efficiently, D-PKG should not redistribute secret keys for domain users; finally, domain users need to guarantee that all operations should be processed faithfully.

4.4.1. Data operation

From data perspective, the domain users are about to create and delete the domain data.

Data creation: In a domain, to achieve data collaboration, any user in the domain can create new data for his project and store the data into the Cloud Server. When new data is created, SECO first chooses a unique *ID* for the new data, and then the data creator decides the intended recipients. Finally, the creator encrypts the data with recipients' public keys and uploads the ciphertext with his signature to the Cloud Server. If verifying the signature correctly, the Cloud Server stores the new data. Otherwise, the Cloud Server rejects the data. Upon completion of the current work, the user can go offline as he likes.

Data deletion: In a domain, SECO also provides data deletion operation. The delete operation we consider here is straightforward. Only D-PKG can delete the domain data in SECO. When the D-PKG is ready to delete a data, he sends the data *ID* and his signature to the Cloud Server. After verifying the signature on this data *ID* correctly, the Cloud Server deletes the data.

4.4.2. User operations

From users perspective, to preserve domain data security, the D-PKG is about to add new users into the domain and revoke outdated users from the domain.

User addition: Sometimes, some new users need to join a domain for working. In SECO, secret key transmission can be implemented locally. So when a new user applies for joining the domain, the D-PKG first verifies identity of the user, if correct, the D-PKG first gives the user a new ID and calculates a private key for the user using the key generation algorithm in Section 4.3. Then, the D-PKG sends the ID-tuple and the private key to the user. Finally, the D-PKG adds the new user to the domain user list UL_{dom} and sends the new domain user list UL_{dom} to all valid users in the domain. The new user picks a random seed as his master key. After receiving the public/private key from D-PKG, the user can access the domain data correctly. In addition, the user can use other users' public key to encrypt.

User revocation: In a domain, the D-PKG may revoke some users' access privileges to preserve data security. The users are not allowed to access (read, write, update, etc.) the domain data anymore after revoking. In SECO, users encrypt data with recipients' public keys. When there is a user to be revoked from the domain, D-PKG first cancels his public key. The D-PKG removes the revoked user from the domain user list UL_{dom} and sends the new domain user list UL_{dom} to all valid users in the domain. Then, the data that is encrypted with this public key will be re-encrypted by D-PKG. From then on, the domain users encrypt data without the canceled public key. After revoking from the domain, the users cannot access the domain data anymore, even if he colludes with other unauthorized users. In some related works, the D-PKG need to update the secret keys for the non-revoked users (Yu et al., 2010) when there exist user revocation operations. However, SECO does not need to update the keys for non-revoked users because the private key of each user is independent.

4.4.3. Domain operations

From domains perspective, SECO is about to add a new domain and revoke a outdated domain from the system.

Domain addition: Sometimes, some new domains need to join in a system for economic reason. In SECO, each domain is independent. That is to say, secret keys among each domain are independent. So when a new domain applies for joining the system, the system first verifies identity of the D-PKG. If correct, the system gives an ID-tuple and calculates a secret key for the joining D-PKG. Finally, the new D-PKG generates secret keys for domain users using the key generation algorithm in Section 4.3. The domain users can access their domain data correctly.

Domain revocation: As we know, the domains are independent. When there is a domain which wants to leave the system, SECO just removes all the secret keys and cloud files belonging to this domain. This operation will not affect other domains.

4.5. Discussion

Based on the current research, two issues remain to be addressed in SECO.

4.5.1. Data consistency

SECO supports multi-user reading and writing operations. Suppose a user *A* downloads and modifies a data file. Before *A* uploading the modified data, another user *B* also downloads the same date and modifies it. If user *A* and user *B* both upload their modified data to the Cloud Server, the data uploaded by user *A* does not include user *B*'s modification, vice versa. Thus, data conflict will occur. It results in the problem of data consistency. SECO can utilize the Cloud Server to solve the problem of data consistency. When more than one user downloads and modify a same data from the Cloud Server at the same time, the Cloud Server detects the data conflict if these users upload their data. Consequently, the Cloud Server just keeps the latest one as the final version. Meanwhile, the Cloud Server informs other users that their modifications are not successful.

4.5.2. Signature

SECO also has the ability to support signature. Compared to traditional public key infrastructure (PKI), IBE scheme does

not require online public key lookup. Indeed, we can transform any PKI signature scheme to an ID-based signature scheme using certificates. When a user E^i_{dom+1} wants to sign M with his public key/ID-tuple (ID-tuple$_{dom}$, ID^i_{dom+1}), the following setups are executed:

- calculates $P_M = H_3((ID-tuple_{dom}, ID^i_{dom+1}), M) \in \mathbb{G}_1$;
- calculates $Sig_M = Sig((ID-tuple_{dom}, ID^i_{dom+1}), M) = SK^i_{dom+1} + s^i_{dom+1} P_M$;
- sends $[Sig_M, Q_1, \ldots, Q_{dom}, Q^i_{dom+1}]$ as the signature for ((ID-tuple$_{dom}$, ID^i_{dom+1}),M) where $Q_j = s_j P_0$ for $1 \le j \le dom$ and $Q^i_{dom+1} = s^i_{dom+1} P_0$.

When the recipients receive the signature, they confirm the following equation:

$$\hat{e}(P_0, Sig_M) = \hat{e}(Q_0, P_1)\hat{e}\left(Q^i_{dom+1}, P_M\right)\prod_{t=2}^{dom}\hat{e}(Q_{t-1}, P_t)\hat{e}\left(Q_{dom}, P^i_{dom+1}\right).$$

In addition, key management in SECO is straightforward because all D-PKGs only need to keep track of two keys: private key and master key. Therefore, in a domain, the key transmission and signature authentication can be executed locally. There is no out-of-band communication of the key management.

5. Security analysis

In the previous section, we show that our secure data collaboration scheme SECO can realize one-to-many encryption paradigm and writing operation simultaneously. In this section, we first provide a rigorous security proof about the proposed scheme. Then, we analyze the fulfillment of the security requirements discussed in Section 3.2.

5.1. *Security of SECO*

We follow the security definition of the standard IBE (Boneh and Franklin, 2001) and show our scheme is IND-ID-CCA security. We first define the security of our scheme using a game which reflects the notion of IND-ID-CCA security and then present the proof in this section.

5.1.1. *Security definition*

We say that the proposed scheme is semantically secure against an adaptive chosen ciphertext attack (IND-ID-CCA) if no polynomial bounded adversary $\mathscr{A}$ has a non-negligible advantage against the challenger in the following game:

Setup: The challenger runs the *Root Setup* algorithm taking a security parameter K as input. It gives $\mathscr{A}$ the system parameters *params* and keeps the root master key to itself.

Phase 1: The adversary $\mathscr{A}$ can issue queries $q_1,\ldots, q_m$ where q_i is one of: 1)H_1-*query* (ID-tuple$_i$): the adversary obtains H(ID-tuple$_i$) corresponding to ID-tuple$_i$; 2) *Private key query* (ID-tuple$_i$): the challenger runs the *Key Generation* algorithm to generate the private key SK_i corresponding to ID-tuple$_i$ and sends it to $\mathscr{A}$; 3) *Decryption query* (ID-tuple$_i$, C_i): the challenger runs the *Key Generation* algorithm to generate the private key SK_i and *Decryption* algorithm to decrypt C_i using SK_i and then gives the resulting plaintext to $\mathscr{A}$. These queries can be asked adaptively by $\mathscr{A}$, that is, each query q_i may depend on the replies to $q_1,\ldots,q_{i-1}$.

Challenge: Once the challenger decides Phase 1 is complete, it outputs N recipients's ID-tuples: ID-tuples$_1$, … ID-tuples$_N$, and two equal length plaintexts $M_0, M_1 \in \mathscr{M}$ on which it wishes to be challenged. The only constraint is that none of the ID-tuples and his ancestors appear in any private key query in Phase 1. These ID-tuples may correspond to positions at the same level in the hierarchy. The challenger picks a random bit $d \in \{0,1\}$ and uses the *Encryption* algorithm to encrypt M_d as $C =$ Encryption (*params*, ID-tuples$_1$, … ID-tuples$_N$, M_d). It sends the challenge C to the adversary $\mathscr{A}$.

Phase 2: The adversary $\mathscr{A}$ issues more queries $q_{m+1},\ldots, q_n$ where q_i is one of: 1)H_1-*query* (ID-tuple$_i$): Challenger responds as in Phase 1; 2) *Private key query* (ID-tuple$_i$ ≠ ID-tuple or ancestor): Challenger responds as in Phase 1; 3) *Decryption query* ((ID-tuple$_i$, C_i) ≠ (ID-tuple or ancestor, C)): Challenger responds as in Phase 1. These queries may be asked adaptively as in Phase 1.

Guess: The adversary $\mathscr{A}$ outputs a guess $d' \in 0,1$. $\mathscr{A}$ wins the game if $d = d'$. The advantage of $\mathscr{A}$ in this game is defined as $\left|Pr[d' = d] - \frac{1}{2}\right|$.

5.1.2. *Proof of security*

We prove the security with the following theorem:

Theorem 5.1. Suppose there is an IND-ID-CCA adversary $\mathscr{A}$ which has the advantage $\varepsilon(k)$ of successfully attacking the scheme SECO. Suppose $\mathscr{A}$ specifies N recipients at level$_t$, and makes at most q_E private key queries, at most q_D decryption queries, and at most $q_{H_2}, q_{H_3}, q_{H_4}$ queries to the hash functions H_2,H_3,H_4 in Level$_t$ respectively. Then there is an algorithm $\mathscr{B}$ for $\mathscr{IG}$ that solves the BDH problem with the advantage at least $2FO_{adv}\left(\frac{\varepsilon(k)(N+t)^{N+t}}{(e(N+t+q_E+q_D))^{N+t}}, q_{H_3}, q_{H_4}, q_D\right) \Big/ q_{H_2}$ where the function FO_{adv} is defined in Theorem 14 in Fujisaki and Okamoto (1999). Here $e \approx 2.71$ is the base of the natural logarithm.

The proof of Theorem 5.1 will use the result of Theorem 14 in Fujisaki and Okamoto (1999). According to Theorem 14 in Fujisaki and Okamoto (1999), we need the following Lemma 5.1 to translate between an IND-ID-CCA on SECO and an IND-CCA on BasicPub^{hy} that is a related public key encryption scheme used in Boneh and Franklin (2001). BasicPub^{hy} is the result of applying the Fujisaki-Okamoto transformation (Fujisaki and Okamoto, 1999) to a public key encryption scheme (not an identity based scheme) called BasicPub. The details of BasicPub and BasicPub^{hy} are presented in Boneh and Franklin (2001). The difference here is that the message is encrypted with multiple recipients in BasicPub^{hy}.

Lemma 5. 1. Let $\mathscr{A}$ be an IND-ID-CCA adversary that has advantage $\varepsilon(k)$ against SECO. Suppose $\mathscr{A}$ makes at most q_E private key queries and at most q_D decryption queries. Then there is an IND-CCA adversary $\mathscr{B}$ that solves BasicPub^{hy} with the advantage at least $\frac{\varepsilon(k)(N+t)^{N+t}}{(e(N+t+q_E+q_D))^{N+t}}$.

Proof: We construct an IND-CCA adversary $\mathscr{B}$ that uses $\mathscr{A}$ to obtain advantage $\frac{\varepsilon(k)(N+t)^{N+t}}{(e(N+t+q_E+q_D))^{N+t}}$ against BasicPub^{hy}. The challenger and the adversary $\mathscr{B}$ start with the challenger by

running key generation algorithm of BasicPub^{hy}. The result is $param = <q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, N, P_0, Q_0, P_{\mathscr{B}}, H_2, H_3, H_4>$ and a private key $SK_{\mathscr{B}} = s_0 P_{\mathscr{B}}$. Here $Q_0 = s_0 P_0$. The challenger gives *param* to $\mathscr{B}$. $\mathscr{B}$ mounts an IND-CCA attack on *param* with the help of adversary $\mathscr{A}$. $\mathscr{B}$ interacts with $\mathscr{A}$ as the above game:

Setup: $\mathscr{B}$ gives $\mathscr{A}$ $<q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, N, P_0, Q_0, H_1, H_2, H_3, H_4>$ as the system parameters. Here H_1 is controlled by $\mathscr{B}$ as described below.

H_1-queries: At any time, $\mathscr{A}$ can query the hash H_1 which will be used to determine the P-tuple$_i$ $= (T_{i1}, \ldots, T_{it_i})$, with $T_{ik} = H_1(ID_{i1},\ldots,ID_{ik})$ for $1 \le k \le t_i$, corresponding to the ID-tuple$_i$ $= (ID_{i1}, \ldots, ID_{it_i})$. $\mathscr{B}$ maintains a list of tuples (ID-tuple$_i$, P-tuple$_i$, b-tuple$_i$, s-tuple$_i$, c-tuple$_i$) called H_1^{list}. The list is initially empty. When $\mathscr{A}$ queries H_1 at a point ID-tuple$_i$ $\mathscr{B}$ responds as follows: Let y be maximal such that $(ID_{i1},\ldots ID_{iy}) = (ID_{j1},\ldots,ID_{jy})$ for the tuple $((ID_{j1},\ldots ID_{jt_j}), (T_{j1}, \ldots, T_{jt_j}), (b_{j1}, \ldots, b_{jt_j}), (s_{j1}, \ldots, s_{jt_j}), (c_{j1}, \ldots, c_{jt_j}))$ which is in H_1^{list}.

- If ID-tuple$_i$ already appears on H_1^{list} for $1 \le k \le y$, then $\mathscr{B}$ responds with $T_{ik} = T_{jk}, s_{ik} = s_{jk}, b_{ik} = b_{jk}$, and $c_{ik} = c_{jk}$. (Note that this is independent of j.)
- Otherwise ($y \le k < t_i$), $\mathscr{B}$ picks two random seeds s_{ik} and $b_{ik} \in \mathbb{Z}_q$, set $c_{i0} = 0$ and generates a random coin $c_{ik} \in \{0,1\}$ so that $Pr[c_{ik} = 0] = \delta$ for some δ that will be determined later.
- If $c_{ik} = c_{i(k-1)}$, $\mathscr{B}$ computes $T_{ik} = b_{ik} P_0$. If $c_{ik} = 1$ and $c_{i(k-1)} = 0$, computes $T_{ik} = b_{ik} P_{\mathscr{B}}$. If $c_{ik} = 0$ and $c_{i(k-1)} = 1$, computes $P_{ik} = b_{ik} P_0 - s_{i(k-1)}^{-1} b_{if(k)} P_{\mathscr{B}}$ *where* $s_{i(k-1)}^{-1}$ is the inverse of $s_{i(k-1)}$ modulo q. Here, $f(k) < k$ is the largest subscript which satisfies $c_{if(k)} = 1$ and $c_{i(f(k)-1)} = 0$.
- $\mathscr{B}$ adds the tuple $((ID_{i1}, \ldots, ID_{it_i}), (T_{i1}, \ldots, T_{it_i}), (b_{i1}, \ldots, b_{it_i}), (s_{i1}, \ldots, s_{it_i}), (c_{i1}, \ldots, c_{it_i}))$ to the H_1^{list} and responds to $\mathscr{A}$ with P-tuple$_1$ $= (T_{i1}, \ldots, T_{it_i})$ to $\mathscr{A}$.

Note that these values (T_{ik}) are uniform in $\mathbb{G}_1$ and independent of $\mathscr{A}$'s current view as required.

Phase 1: Private key queries. Let ID-tuple$_i$ be a private key query issued by adversary $\mathscr{A}$. $\mathscr{B}$ responds to this query as follows:

- $\mathscr{B}$ runs the H_1-queries algorithm to obtain the corresponding tuples (ID-tuple$_i$, P-tuple$_i$, b-tuple$_i$, s-tuple$_i$, c-tuple$_i$) on the H_1^{list}. If $c_{it_i} = 1$, $\mathscr{B}$ reports failure and terminates the interaction.
- Therefore $c_{it_i} = 0$, $\mathscr{B}$ computes $SK_{it_i} = \Sigma_{k=1}^{t_i} p_{i(k-1)} T_{ik}$ where $p_{i(k-1)} = s_{i(k-1)} s_{i(f(k)-1)} s_{i(f(f(k))-1)} \ldots$ with $s_{i0} = s_0$ and $s_{i(f(j)-1)} = 1$ if $f(j)$ does not exist.

$\mathscr{B}$ does not know the values of s_0 or $s_0 P_{\mathscr{B}}$, but it still can output the private key for ID-tuple$_i$.

Phase 1: Decryption queries. Let (ID-tuple$_i$, C_i) be a decryption query issued by $\mathscr{A}$. $\mathscr{B}$ responds to this query as follows:

- $\mathscr{B}$ runs the H_1-queries algorithm to obtain the corresponding tuples (ID-tuple$_i$, P-tuple$_i$, b-tuple$_i$, s-tuple$_i$, c-tuple$_i$) on the H_1^{list}.
- Suppose $c_{it_i} = 0$, $\mathscr{B}$ runs the private key queries to obtain the private key for ID-tuple$_i$. Then $\mathscr{B}$ uses the private key to respond to the decryption query.
- Suppose $c_{it_i} = 1$, $\mathscr{B}$ relays the decryption query C_i to the challenger and relays the challenger's response back to $\mathscr{A}$.

Challenge: Once $\mathscr{A}$ decides that Phase 1 is over, it outputs N ID-tuples: ID-tuples$_1$, … ID-tuples$_N$ $\in$ Level$_t$, and two plaintext M_0,M_1 on which it wishes to be challenged. $\mathscr{B}$ responds as follows:

- $\mathscr{B}$ runs the H_1-queries algorithm to obtain the corresponding tuples $((ID_{j1},\ldots,ID_{jt}), (T_{j1},\ldots,T_{jt}), (b_{j1},\ldots,b_{jt}), (s_{j1},\ldots,s_{jt}), (c_{j1},\ldots,c_{jt}))$ for each ID-tuple$_j$ on the H_1^{list}, where $j \in \{1,2,\ldots,N\}$.
- If $c_{jk} = 0$ for some $1 \le k \le t$, then $\mathscr{B}$ reports failure and terminates. Otherwise, we know $T_{j1} = b_{j1} P_{\mathscr{B}}$ and $T_{jk} = b_{jk} P_0$, for $2 \le k \le t$.
- $\mathscr{B}$ gives the challenger M_0, M_1. Let $C = [U,V,W]$. The challenger responds with a ciphertext $C' = [b_{11}^{-1} b_{1t} U, b_{21}^{-1} b_{2t} U, \ldots, b_{N1}^{-1} b_{Nt} U, b_{j1}^{-1} U, b_{j1}^{-1} b_{j1} U, \ldots, b_{j1}^{-1} b_{j(t-1)} U, V, W]$ for $j \in \{1,2,\ldots,N\}$ such that C' is the encryption of M_d for a random $d \in \{0,1\}$ and gives C' to $\mathscr{A}$.

In this challenge, the private key for ID-tuple$_j$ is $SK'_{jt} = s_0 T_{j1} + \Sigma_{k=1}^{t-1} s'_{jk} T_{j(k+1)}$ with the additional information $\{s'_{jk} P_0 : 1 \le k < t\}$ for some $(s'_{j1},\ldots,s'_{j(t-1)}) \in (\mathbb{Z}_q)^{t-1}$. Observe that:

$$\hat{e}\left(b_{j1}^{-1} U, SK'_{jt}\right) \Big/ \prod_{k=2}^{t} \hat{e}\left(b_{j1}^{-1} b_{jk} U, s'_{j(k-1)} P_0\right) = \hat{e}\left(b_{j1}^{-1} U, s_0 T_{j1}\right) = \hat{e}(U, s_0 P_{\mathscr{B}}).$$

Therefore, C' is a valid ciphertext for M_d.

Phase 2: Adversary $\mathscr{B}$ responds to queries at a point ID-tuple$_i$ in the same way it did in Phase 1. The constraint for $\mathscr{A}$ is listed in the definition of security model.

Guess: Eventually adversary A outputs a guess d' for d. $\mathscr{B}$ outputs d' as its guess for d.

The responses to H_1-queries are uniformly and independently distributed in $\mathbb{G}_1$. All responses to private key and decryption queries are valid. The challenge ciphertext C' given to $\mathscr{A}$ is the encryption of M_d for random $d \in \{0,1\}$. So, we have $\left|Pr[d' = d] - \frac{1}{2}\right| \ge \varepsilon(k)$. Then, calculate the probability that $\mathscr{B}$ aborts during the simulation. Let ε_1 be the event that $\mathscr{A}$ issues a bad private key query during phase 1 or 2, ε_2 be the event $\mathscr{A}$ chooses a bad ID-tuple$_i$ to be challenge and ε_3 be the event $\mathscr{A}$ issues a bad decryption query during phase 2. We have: $Pr[\neg\varepsilon_1 \wedge \neg\varepsilon_2 \wedge \neg\varepsilon_3] \ge \delta^{q_E+q_D}(1-\delta)^{N+t}$. Please refer to Boneh and Franklin (2001) for the proof of the above formula. Here N is the number of recipients for a ciphertext and t is the level of the recipients. We now optimize the choice of δ. Since $Pr[\neg\varepsilon_1 \wedge \neg\varepsilon_2 \wedge \neg\varepsilon_3] \ge \delta^{q_E+q_D}(1-\delta)^{N+t}$, the probability that $\mathscr{B}$ does not abort during the simulation is $\delta^{q_E+q_D}(1-\delta)^{N+t}$. Therefore, the success probability is maximized at $\delta_{opt} = 1 - (N+t)/(q_E + q_d + N + t)$. Using δ_{opt}, the probability that $\mathscr{B}$ does not abort is at least $\left(\frac{N+t}{(e(N+t+q_E+q_D))}\right)^{N+t}$. This shows that $\mathscr{B}$'s success probability is at least $\frac{\varepsilon(k)(N+t)^{N+t}}{(e(N+t+q_E+q_D))^{N+t}}$ as required.

We give the proof of Theorem 5.1 as follows:

Proof: By Lemma 5. 1 an IND-ID-CCA adversary on SECO implies an IND-CCA adversary on BasicPub^{hy}. From the proof of Theorem 4.4 in Boneh and Franklin (2001), this can imply an algorithm against BHD assumption. All these give the required bounds in Theorem 5.1.

According to Theorem 5.1, we can conclude that SECO is IND-ID-CCA security.

5.2. Security requirements

For the purpose of secure data collaboration in cloud computing, SECO should achieve the following security properties:

5.2.1. Fine-grained of access control

In SECO, the user who modifies data is able to define and enforce who can access this data and encrypt with multiple recipients' public keys. Each user has secret keys from the D-PKG. Suppose a user E_i download the encrypted data. We recall that the ciphertext is: $C = [\{rP^i_{dom+1}\}, \{rP_t\}, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma)]$. If this data is encrypted with E^i_{dom+1} public key, E^i_{dom+1} can obtain the corresponding $U^i_{dom+1} = rP^i_{dom+1}$, and then decrypts this data by calculating: $W \oplus H_4(V \oplus H_2(\hat{e}(U_0, SK^i_{dom+1}) / \prod_{j=2}^{dom} \hat{e}(Q_{j-1}, U_j)\hat{e}(Q_{dom}, U^i_{dom+1})))$ to obtain the plaintext. However, if a user is not in the encryption list, then he cannot obtain U_i in the ciphertext text. So the decryption algorithm will fail. Specifically, only those intended recipients can decrypt this data. Therefore, users only can access the data they are allowed and not access the data they are not authorized to.

5.2.2. Fully collusion secure

In SECO, the data $\mathcal{M}$ is encrypted in the form of $C = [\{U^i_{dom+1}\}, \{U_t\}, V, W]$ where $V = \sigma \oplus H_2(g^r)$ and $W = M \oplus H_4(\sigma)$. Obviously, unauthorized users must construct $H_2(g^r)$ where $g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2$ to decrypt ciphertext C. Although unauthorized users can obtain Q_0 and P_1, they are unaware of the random seed r, so $\hat{e}(Q_0, rP_1)$ cannot to be constructed directly. Beside, unauthorized users observe that: $\hat{e}(Q_0, rP_1) = \hat{e}(rP_0, s_0P_1) = \hat{e}(U_0, SK_0)$. To recover plaintext, unauthorized users may recover $\hat{e}(Q_0, rP_1)$ instead of $\hat{e}(U_0, SK_0)$. However, since SK_0 is only known to R-PKG, unauthorized users also cannot recover $\hat{e}(U_0, SK_0)$. Therefore, colluded users cannot recover plaintext.

5.2.3. Backward secrecy

As described in Section 4.4, SECO will re-encrypt the related data after some legitimate users are revoked. The D-PKG will cancel the revoked users' public key and the non-revoked users encrypt data without this public key afterward. Therefore, the revoked users cannot access the encrypted data anymore.

According to the above analysis, we can conclude that SECO can realize fine-grained access control, collusion resistance and backward secrecy.

6. Theoretical analysis

In this section, we provide the theoretical analysis of SECO. We first analyze the computation and communication overhead. Then we analyze the user revocation and storage cost.

6.1. Computation complexity

In SECO, the R-PKG generates two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order q and a bilinear map to achieve the five randomized algorithms. In all computations, pairing computation, i.e., bilinear map computation, is the most expensive operation. In SECO, *Root Setup* generates the system parameters and a master key for R-PKG, and *Lower-level Setup* picks a master key for each lower-level PKG. In *Key Generation*, A PKG generates private keys for all his children. These three algorithms have no pairing computations and need to run only once at initialization time. Moreover, the size of system parameters and keys are fixed in length. Therefore, the computation complexity of these three algorithms is negligible.

Table 2 summarizes the operation numbers required in key generation, encryption and decryption algorithms. In *Key generation*, the D-PKG $E_{dom} \in$ Level$_{dom}$ needs $dom + 1$ scalar multiplications to calculate SK^i_{dom+1} for each domain user E^i_{dom+1}. In *Encryption*, a user encrypts data with N recipients' public keys. He needs one pairing computation to calculate $\hat{e}(Q_0, P_1)$, N scalar multiplications to compute rP^i_{dom+1} for $1 \le i \le N$, and $dom + 1$ scalar multiplications to compute rP_t for $0 \le t \le dom$. Since the pairing computation is independent with data encryption and Q_0, P_{dom} are the same in a domain, for each different data, pairing computation is calculated only once for all domain users. In *Decryption*, the D-PKG needs one pairing computation to calculate $\hat{e}(U_0, SK_{dom})$ and $dom - 1$ pairing computations to calculate $\hat{e}(Q_{j-1}, U_j)$ for $2 \le j \le dom$. Each user E^i_{dom+1} needs $dom + 1$ pairing computations to calculate $\hat{e}(U_0, SK^i_{dom+1})$, $\hat{e}(Q_{j-1}, U_j)$ for $2 \le j \le dom$ and $\hat{e}(Q_{dom}, U^i_{dom+1})$. Since U_0, SK_{dom} and SK^i_{dom+1} are fixed, the D-PKG calculates $\hat{e}(U_0, SK_{dom})$ once, and E^i_{dom+1} calculates $\hat{e}(U_0, SK^i_{dom+1})$ once. Table 3 shows the computation complexity comparison among SECO, ABE-based schemes and identity-based broadcast encryption (BE) schemes (Delerablée, 2007; Gentry and Waters, 2009). Here $|I|$ is the number of attributes in ABE-based schemes and N is the number of users. In practice, N is larger than dom. Therefore, we can see SECO takes fewer computation complexities than BE-based schemes and ABE-based schemes. From the above analysis, SECO only needs a few pairing computations to achieve secure data collaboration in cloud computing, so the computation complexity of SECO is acceptable.

6.2. Communication cost

In SECO, the communication cost is mainly attributable to the encrypted data transmission. After encryption, the following information is sent by users along with the encrypted data to the cloud: Value of U^i_{dom+1} for every intended recipient which requires $Nlog|\mathbb{G}_1|$ bits, value of U_t which requires

Table 2 – Operation numbers required in encryption and decryption algorithm.

Algorithm	Scalar multiplication	Pairing
Key generation	dom + 1	0
Encryption	N + dom + 1	1
Decryption (D-PKG/User)	0/0	dom/dom + 1

Table 3 – Computation complexity required in ABE-based schemes, BE-based schemes and SECO.

Algorithm	Encryption	Decryption
ABE-based schemes	$\mathcal{O}(\|I\|)$	$\mathcal{O}(max(\|I\|, N))$
BE-based schemes	$\mathcal{O}(N)$	$\mathcal{O}(N)$
SECO	$\mathcal{O}(max(N, dom)$	$\mathcal{O}(dom)$

$(dom+1)log|\mathbb{G}_1|$ bits, value of V which requires n bits, and value of W which requires n bits. Thus, the communication cost is given by $(N+dom+1)log|\mathbb{G}_1|+2n$ bits. Table 4 shows the communication expenses comparison among SECO, ABE-based schemes, and symmetric key cryptosystem (SKC) schemes. Here n is the length of the plaintext, t is the number of the users, I is the number of attributes used in ABE-based schemes (Yu et al., 2010) and k is the length of keys used in SKC-based schemes (Goh et al., 2003). Since the data size is fixed (n), N, k, dom and I are varying but have the same order of magnitude as n. From Table 4, we can see that SECO takes fewer communication cost than ABE-based schemes and SKC-based schemes. The reason is that every data block is bind with t users KeyID and two secret keys in SKC-based schemes, while in ABE-based schemes, the data owner needs transfer the access structure of the data and other parameters to the cloud. From the above analysis, SECO takes small communication overhead to achieve secure and efficient data collaboration services in cloud computing.

6.3. Cost of revocation operation

When user revocation is necessary, the related ciphertext needs to be re-encrypted without the revoked user public key in SECO. We first evaluate the computation cost of the revocation operation. The user who encrypts the data will choose a new σ randomly and recalculate U^i_{dom+1}, V and W. To update V and W, the user only needs Boolean XOR operator. For each corresponding recipient public key, there is one scalar multiplication to update U^i_{dom+1}. The system does not need to update the secret keys for non-revoked users because the secret keys of each user is independent. Therefore, there are N scalar multiplications in total to re-encrypt the ciphertext by the user. For the non-revoked users, they do not need to do any computation. Next we give the communication cost of the user revocation. After re-encrypting the ciphertext, the user sends the new ciphertext to the cloud, while the cloud just replaces the outdated ciphertext and does not need to transfer it to the non-revoked users, so the additional communication costs is $Nlog|\mathbb{G}_1|+2n$.

In the existing works, when revocation happens, the data owner needs to re-encrypt the related ciphertext and issue the new keys to those non-revoked users. Table 5 shows the user revocation costs comparison between SECO and ABE-based schemes. Here, I is the number of attributes which the revoked user possessed and k is the depth of the access structure. From Table 5, ABE-based schemes bring an abundance of additional computation overhead. SECO can accomplish this dynamic request with lightweight computation complexity and communication cost.

Table 4 – Communication cost in ABE-based schemes, SKC-based schemes BE-based schemes and SECO.

Scheme	Communication costs
ABE-based	$\|I\|+2log\|I\|+(\|I\|+1)log\|\mathbb{G}_1\|+log\|\mathbb{G}_2\|+n$
SKC-based	$3tk+n$
SECO	$(N+dom+1)log\|\mathbb{G}_1\|+2n$

Table 5 – User revocation cost in ABE-based schemes and SECO.

Scheme	Scalar multiplication	Pairing
ABE-based	Ik	1
SECO	N	0

6.4. Storage cost

The storage cost is one of the most significant aspects of the data access control scheme in cloud storage services. We analyze the storage overhead of SECO and compare it with SKC-based schemes and ABE-based schemes. The storage cost is assessed in terms of ciphertext storage overhead and key storage overhead (secret keys and system parameters stored on the users and D-PKG). Table 6 presents the comparative results.

Ciphertext storage overhead: In ABE-based schemes, the size of ciphertext is $\mathcal{O}(max(|I|, n))$, with $|I|$ as the number of attributes the ciphertext issued. For SKC-based schemes, to achieve read and write permission, each data is binding with each user access privilege. The size of ciphertext depends on the numbers of users and the size of key. Thus, the size is $\mathcal{O}(n^2)$. For BE-based schemes, the size of ciphertext is $\mathcal{O}(N)$, with N is the number of recipients. In SECO, as depicted in Section 4, the bit-length of the ciphertext grows only linearly with the level of the message recipient. The ciphertext is composed of N intended recipients' information, $dom+1$ hierarchy information and a body. The body is just the encrypted message. The length of the ciphertext is linear with the recipient quantity. The length will increase an element on $\mathbb{G}_1$ when adding a recipient. Thus the message size is $\mathcal{O}(max(N, dom))$. From Table 6, we can see SECO and BE-based schemes take the least ciphertext storage cost. The analysis indicates that SECO achieves efficient data collaboration with light weighted ciphertext storage.

Key storage overhead: Compared with ABE-based schemes, SKC-based schemes and BE-based schemes, SECO greatly reduced the key storage overhead of the D-PKG (data owner). In ABE-based schemes, SKC-based schemes and BE-based

Table 6 – Storage cost in ABE-based schemes, SKC-based schemes, BE-based schemes and SECO.

Scheme	Ciphertext storage	Key storage	
		D-PKG (Data owner)	User
ABE-based	$\mathcal{O}(max(\|I\|, n))$	$\mathcal{O}(n)$	$\mathcal{O}(logn)$
SKC-based	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
BE-based	$\mathcal{O}(N)$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$
SECO	$\mathcal{O}(max(N, dom)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

schemes, the data owner needs to store every user's access privilege. While in SECO, the D-PKG just stores his own secret keys and system parameters. Users only need store their own secret keys and system parameters in SCK-based schemes and SECO. However, users in ABE-based schemes have to store their own access structures with there corresponding secret keys. Therefore, SECO also takes small key storage overhead to achieves data collaboration in cloud computing.

7. Experimental evaluation

In this section, we evaluate the performance of the algorithms used in SECO. In our experiments, we utilize a three-level HIBE scheme, where $\text{Level}_0 = \{\text{Root PKG}\}$ and $\text{Level}_1 = \{\text{D-PKGs}\}$. All the domain users are lying at Level_2. We calculate the time cost and report the average of 100 trials of each algorithm. We first compare the overhead of SECO with ABE-based schemes and BE-based schemes, and then examine the scalability of our scheme. Our implementation was done in Python, and all experiments were performed on an Intel Core 2 Duo 2.0 GHz PC with 2 GB RAM.

In the first set of experiments, we evaluate the overhead of encryption and decryption algorithms in SECO, ABE-based schemes and BE-based schemes. We assume the number of recipient users in SECO and BE-based schemes, and attributes in ABE-based schemes both are 500. Figs. 3 and 4 plots the overhead as the data size varies in SECO, ABE-based schemes and BE-based schemes. Fig. 3 plots the time cost of encryption algorithms in SECO, ABE-based schemes and BE-based schemes. From Fig. 3, we can see the encryption cost increases linearly with the file size in all the three schemes. This is consistent with the above computation complexity analysis. However, with the increase of file size, SECO takes less time cost than ABE-based schemes and BE-based schemes. Fig. 4 plots time cost of encryption algorithms in SECO, ABE-based schemes and BE-based schemes. In Fig. 4, we notice the cost in all the three schemes is nearly linearly proportional to the number of the data size. Meanwhile, users take more time than the D-PKG in decryption as the above analysis in SECO. To decrypt 64 KB data, the D-PKG takes 1 s and user E^i_{dom+1} takes

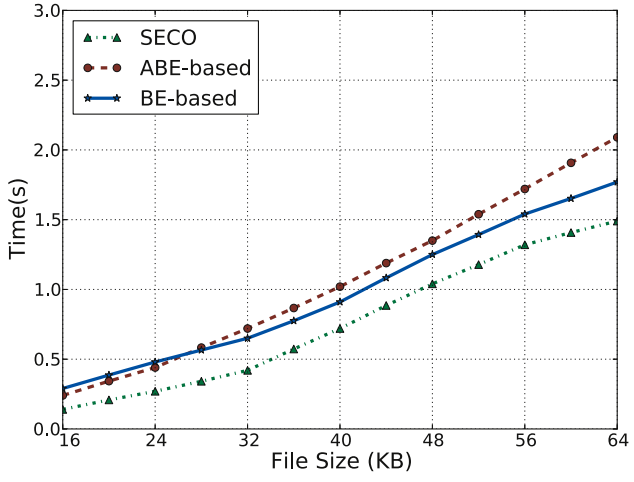

Fig. 3 – The overhead of encryption algorithms in SECO, ABE-based schemes and BE-based schemes.

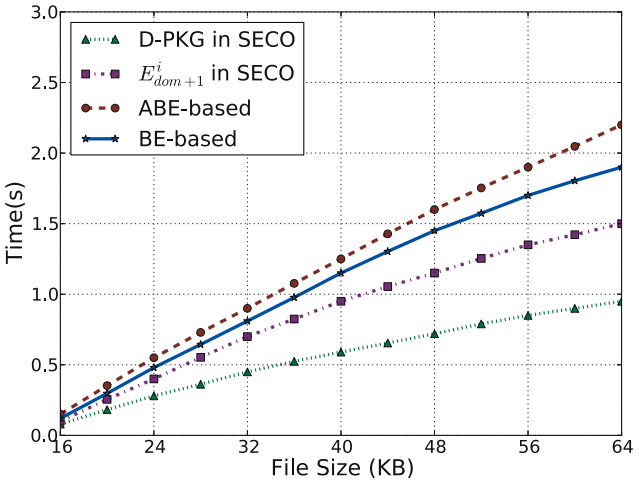

Fig. 4 – The overhead of decryption algorithms in SECO, ABE-based schemes and BE-based schemes.

1.5 s. The algorithm does the pairing computation for each data, but the pairing computation can be done once at the beginning as the above analysis. However, both the D-PKG and users in SECO take less time cost than ABE-based schemes and BE-based schemes as well. The results of this experiments show SECO is light weighted and efficient to be applied in practice.

In the second set of experiments, we evaluate the scalability of SECO. According to the analysis in Section 6, the computation complexity of encryption algorithm is $\mathcal{O}(max(N, dom))$. In our experiment setting, we have $N > dom$. Therefore, we study the overhead under different data size and different number of users. Fig. 5 plots the encryption overhead for preparing three kinds of data size as the number of recipients varies. From Fig. 5, we see the encryption time grows linearly with the number of the recipients. The time to encrypt 64 KB data with 800 recipients approaches to 1.6 s, which is an ideal result. Moreover, the time cost is relatively stable versus the number of users. Thus, our scheme is scalable in cloud computing. Fig. 6 plots the re-encryption cost for preparing three kinds of data size as the number of revoked users varies. The total number of domain users in Fig. 6 is

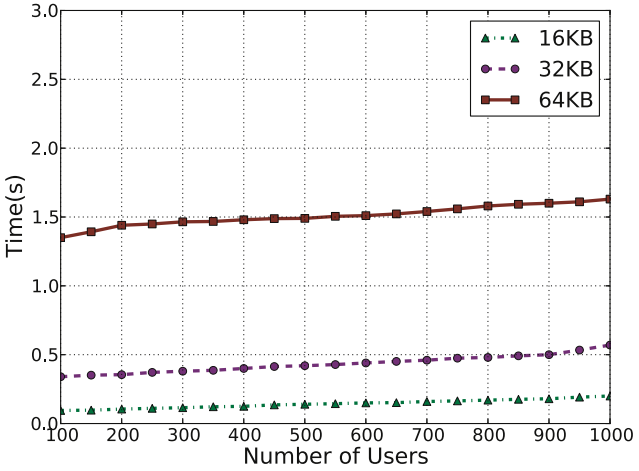

Fig. 5 – The overhead of encryption algorithm.

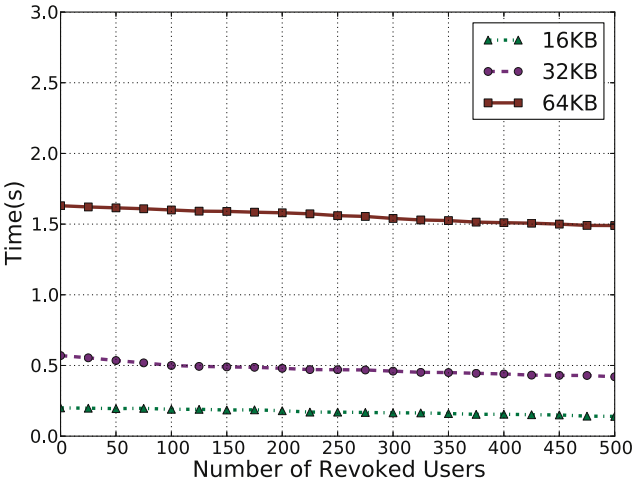

Fig. 6 – The overhead of re-encryption algorithm.

1000. From Fig. 6, we can see the re-encryption cost decreases linearly with the number of revoked users. When the system revokes 100 users, the time to re-encrypt 32 KB data approaches to 0.5 s. The result shows our user revocation scheme is efficient. As a summary, our scheme is scalable to large number of users.

8. Conclusion

In this paper, we address the one-to-many encryption paradigm, writing operation and fine-grained access control issue, and propose a secure cloud data collaboration scheme SECO with explicit dynamic data/user. SECO employs a multi-level HIBE scheme to guarantee data security against the cloud. SECO realizes a one-to-many encryption paradigm and data writing operation simultaneously to achieve secure data collaboration in cloud computing. Moreover, SECO provides dynamic operations such as data creation/deletion and user addition/revocation. Security analyses show that SECO is IND-CCA security under the BDH assumption and can realize fine-grained access control, collusion resistance and backward secrecy. In addition, we evaluate the performance of SECO about computation complexity, communication cost, user revocation cost and storage cost. The result shows that SECO has low overhead and is highly efficient. Following the current research, we will implement the proposed secure data collaboration services in a real CSP platform, address the privacy issues and work on the data synchronization in SECO for future work.

Acknowledgment

This work is supported in part by the National High Technology Research and Development Program of China (863 Program) under Grant 2015AA011403, the Research Fund for the Doctoral Program of Higher Education of China (NO. 20120073120034), the National Natural Science Foundation of China under Grant 61271222 and the National Basic Research Program of China under Grant 2013CB338004.

Appendix A. Supplementary data

Supplementary data related to this article can be found at http://dx.doi.org/10.1016/j.cose.2015.01.003.

REFERENCES

Adya A, Bolosky WJ, Castro M, Cermak G, Chaiken R, Douceur JR, et al. Farsite: federated, available, and reliable storage for an incompletely trusted environment. ACM SIGOPS Oper Syst Rev 2002;36:1–14.

Agrawal S, Gorbunov S, Vaikuntanathan V, Wee H. Functional encryption: new perspectives and lower bounds. In: Advances in cryptology–CRYPTO 2013. Springer; 2013. p. 500–18.

Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, et al. Above the clouds: a Berkeley view of cloud computing. Technical Report. Berkeley: EECS Department, University of California; 2009. Tech. Rep. UCB/EECS-2009-28.

Arora R, Parashar A, Transforming CCI. Secure user data in cloud computing using encryption algorithms. Int J Eng Res Appl (IJERA) 2013;3(4):1922–6.

Arrington M. Gmail disaster: reports of mass email deletions. 2006. Retrieved online on 28 December 2006 from, http://www.techcrunch.com/2006/12/28/gmail-disasterreports-ofmass-email-deletions/.

Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on computer and communications security; 2007. p. 598–609.

Baek J, Safavi-Naini R, Susilo W. Efficient multi-receiver identity-based encryption and its application to broadcast encryption. In: Public key cryptography–PKC. Springer; 2005. p. 380–97.

Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: IEEE symposium on security and privacy; 2007. p. 321–34.

Boneh D, Boyen X, Goh EJ. Hierarchical identity based encryption with constant size ciphertext. In: Advances in cryptology–EUROCRYPT. Springer; 2005. p. 440–56.

Boneh D, Franklin M. Identity-based encryption from the Weil pairing. In: Advances in cryptology–CRYPTO. Springer; 2001. p. 213–29.

Boneh D, Raghunathan A, Segev G. Function-private identity-based encryption: hiding the function in functional encryption. In: Advances in cryptology–CRYPTO 2013. Springer; 2013. p. 461–78.

Cocks C. An identity based encryption scheme based on quadratic residues. In: Cryptography and coding. Springer; 2001. p. 360–3.

De Capitani di Vimercati S, Foresti S, Jajodia S, Paraboschi S, Samarati P. Encryption policies for regulating access to outsourced data. ACM Trans Database Syst (TODS) 2010;35(2):12:1–12:46.

Delerablée C. Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Advances in cryptology–ASIACRYPT. Springer; 2007. p. 200–15.

Dong X, Yu J, Luo Y, Chen Y, Xue G, Li M. Achieving secure and efficient data collaboration in cloud computing. In: Proceedings of the 2013 IEEE/ACM international symposium on quality of service; 2013. p. 195–200.

Dong X, Yu J, Luo Y, Chen Y, Xue G, Li M. Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing. Comput Secur 2014;42(0):151–64.

Fiat A, Naor M. Broadcast encryption. In: Advances in cryptology–CRYPTO. Springer; 1994. p. 480–91.

Fujisaki E, Okamoto T. Secure integration of asymmetric and symmetric encryption schemes. In: Advances in cryptology–CRYPTO. Springer; 1999. p. 537–54.
Gentry C, Silverberg A. Hierarchical ID-based cryptography. In: Advances in cryptology–ASIACRYPT. Springer; 2002. p. 548–66.
Gentry C, Waters B. Adaptive security in broadcast encryption systems (with short ciphertexts). In: Advances in cryptology–EUROCRYPT. Springer; 2009. p. 171–88.
Goh EJ, Shacham H, Modadugu N, Boneh D. Sirius: securing remote untrusted storage. In: Proceedings of 10th ISOC network and distributed system security symposium; 2003. p. 40–55.
Goldwasser S, Gordon S, Goyal V, Jain A, Katz J, Liu FH, et al. Multi-input functional encryption. In: Advances in cryptology–EUROCRYPT. Springer; 2014. p. 578–602.
Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security; 2006. p. 89–98.
Guo SC, Liu Y, Ling J. A quasi IBE identity authentication scheme in a cloud computing environment. Adv Mater Res 2013;756:837–40.
Horwitz J, Lynn B. Toward hierarchical identity-based encryption. In: Advances in cryptology–EUROCRYPT. Springer; 2002. p. 466–81.
Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K. Plutus: scalable secure file sharing on untrusted storage. In: Proceedings of the 2nd USENIX conference on file and storage technologies; 2003. p. 29–42.
Li J, Li J, Chen X, Jia C, Lou W. Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans Comput 2013a;99(1):121–32.
Li M, Yu S, Zheng Y, Ren K, Lou W. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. Parallel and distributed Systems. IEEE Trans 2013b;24(1):131–43.
Mu Y, Susilo W, Lin YX. Identity-based broadcasting. In: Progress in cryptology–INDOCRYPT. Springer; 2003. p. 177–90.
Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 14th ACM conference on computer and communications security; 2007. p. 195–203.
Ren K, Wang C, Wang Q. Security challenges for the public cloud. IEEE Internet Comput 2012;16(1):69–73.
Sahai A, Waters B. Fuzzy identity-based encryption. In: Advances in cryptology–EUROCRYPT. Springer; 2005. p. 557–73.
Samarati P, De Capitani di Vimercati S. Data protection in outsourcing scenarios: issues and directions. In: Proceedings of the 5th ACM symposium on information, computer and communications security (ASIACCS). ACM; 2010. p. 1–14.
Shamir A. Identity-based cryptosystems and signature schemes. In: Advances in cryptology–CRYPTO. Springer; 1985. p. 47–53.
Wang C, Wang Q, Ren K, Cao N, Lou W. Toward secure and dependable storage services in cloud computing. Services computing. IEEE Trans 2012;5(2):220–32.
Wang C, Wang Q, Ren K, Lou W. Privacy-preserving public auditing for data storage security in cloud computing. In: Proceedings of the 31rd IEEE international conference on computer communications; 2010. p. 525–33.
Wang G, Liu Q, Wu J. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In: Proceedings of the 17th ACM conference on computer and communications security; 2010. p. 735–7.
Wang W, Li Z, Owens R, Bhargava B. Secure and efficient access to outsourced data. In: Proceedings of the 2009 ACM workshop on cloud computing security; 2009. p. 55–66.
Wilson S. Appengine outage. 2008. Online at, http://www.cio-weblog.com/50226711/appengineoutage.php.
Yu S, Wang C, Ren K, Lou W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of the 31rd IEEE international conference on computer communications; 2010. p. 534–42.

Xin Dong is a Ph.D. candidate in Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include networking, information security and privacy, mobile computing and cloud computing. He received his Bachelor degree in computer science and engineering from South China University of Technology (SCUT), Guangzhou, China, in 2010.

Jiadi Yu is an assistant professor in Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. He obtained the PhD degree in Computer Science from Shanghai Jiao Tong University, Shanghai, China, in 2007 and the MS degree in computer science from Xi'an Technological University, Xi'an, China, in 2003. In the past, he has worked as a postdoc at Stevens Institute of Technology, USA, from 2009 to 2011. His research interests include networking, mobile computing, cloud computing and wireless sensor networks.

Yanmin Zhu received the BEng degree in computer science from the Xi'an Jiao Tong University in 2002 and the PhD degree in computer science from Hong Kong University of Science and Technology in 2007. He was a research associate in the Department of Computing, Imperial College London. Now, he is an associate professor in the Department of Computer Science and Engineering at the Shanghai Jiao Tong University. His research interests include ad-hoc sensor networks, mobile computing, grid computing, and resource management in distributed systems. He is a member of the IEEE and the IEEE Communication Society.

Yingying Chen received the PhD degree in computer science from Rutgers University. She is working as an assistant professor in the Department of Electrical and Computer Engineering at Stevens Institute of Technology. Her research interests include cyber security and privacy, wireless embedded systems, wireless and sensor networks, mobile social networks, and pervasive computing. She was the recipient of the US National Science Foundation CAREER award in 2010. She was the recipient of the Google Research Award in 2010 and the Best Paper Award from the ACM International Conference on Mobile Computing and Networking (MobiCom) in 2011.

Yuan Luo received the B.S., M.S., and Ph.D. degrees in applied mathematics from Nankai University, Tianjin, China, in 1993, 1996, and 1999, respectively. From July 1999 to April 2001, he held a postdoctoral position at the Institute of Systems Science, Chinese Academy of Sciences, Beijing, China. From May 2001 to April 2003, he held a postdoctoral position at the Institute for Experimental Mathematics, University of Duisburg-Essen, Essen, Germany. Since June 2003, he has been with the Computer Science and Engineering Department, Shanghai Jiao Tong University, Shanghai, China. His current research interests include coding theory and information theory.

Minglu Li received his PHD in Computer Software from Shanghai Jiao Tong University in 1996. He is a Full Professor at the Department of Computer Science and Engineering of Shanghai Jiao Tong University. Currently, his research interests include grid computing, services computing, and cloud computing. He has published over 100 papers in important academic journals and international conferences.