# Achieving Secure and Efficient Data Collaboration in Cloud Computing

Xin Dong[†], Jiadi Yu[†], Yuan Luo[†], Yingying Chen[‡], Guangtao Xue[†] and Minglu Li[†]
[†]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, P.R.China
Email: {xindong, jiadiyu, yuanluo, gt_xue, mlli}@sjtu.edu.cn
[‡]Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken NJ 07030, USA
Email: yingying.chen@stevens.edu

***Abstract*—Cloud storage services enable users to remotely store their data and eliminate excessive local installation of software and hardware. 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 and efficient data collaboration scheme SECO. In SECO, we employ a two-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 service 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 enforces fine-grained access control and collusion resistant. Extensive performance analysis and experiment results demonstrate that SECO is highly efficient and low overhead on computation and communication.**

## I. Introduction

Cloud computing [1], the long-held dream of computing as a utility, is rapidly evolving to revolutionize the way data is stored/used. Cloud computing benefits the data users in that it allows convenient access and use of storage resources offered by a cloud server provided (CSP). Challenges in security, however, posed by outsourcing data to cloud, come along with benefits. Outsourcing users' data to the cloud initiates a series of problems about security and privacy. Examples of security breach never stop showing up [2]. Therefore, maintaining data confidentiality becomes critical to enable wide deployment of CSP-based data service with high quality.

Recently, the notion of secure cloud storage services has been proposed in the content of ensuring remotely stored data under different systems. These existing works addressed secure cloud storage issue either by introducing attribute-based encryption (ABE) [3] for fine-grained access control [4][5], or by utilizing owner-write-user-read mechanism [6] to achieve cryptography-based access control and only support course grained access control. The ABE-based schemes are data-read sharing services, while owner-write-user-read mechanism is a one-to-one encryption paradigms 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 single user. None of works takes into account that multi-users operate data (read/write) collaboratively, i.e, data collaboration service.

A ***Data Collaboration*** service is to support the availability and consistency of the shared data among multi-users. Let's consider a typical data collaboration service scenario. Companies outsource their data to the cloud and then authorize employees to access these data. The untrusted cloud servers, however, may disclose confidential information about an enterprise to their business competitors or even hide data loss to maintain their reputations [7][8]. In order to ensure data security, companies and enterprises usually have to encrypt the data before outsourcing it to the cloud, a fact that challenges the convenience when employees need to access data. To avoid information leakage, data have to be restrained within the reach of authorized users. Thus the access policy is: user can only access the authorized data; the CSP and other unauthorized users knows nothing about data. In addition, whoever updates data can determine the access privilege of the data.

To satisfy the above policies in data collaboration service, we face the following challenges. Firstly, the encryption paradigm should be one-to-many that indicates multiple recipients can decrypt the encrypted data to achieve data collaboration. Secondly, authorized users have the privilege to operate the cloud data, so the encryption paradigm should support data writing operation. Thirdly, the system should provide fine-grained access control to the team members. Thus, realizing secure data collaboration service will be essential in achieving robust and secure cloud storage systems. However, there is no existing solution, to the best of our knowledge, to tackling the problem of secure data collaboration service in cloud computing.

In this paper, we propose a scalable scheme (**SECO**) to enable secure cloud data collaboration with explicit dynamic data/user. For cloud data security, we employ a two-level hierarchical identity-based encryption (HIBE) scheme, which contains a root private key generator (PKG) and a number of independent cooperative domains. Each domain has a domain PKG that requests a private key from the root PKG and generates secret keys for each domain user. During data collaboration, to achieve one-to-many encryption paradigm, a user encrypts data with 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) the decrypted data, and then send it into the cloud to share with other domain users. The data writing operation does not in-

Research was sponsored by the Doctoral Program of Higher Education of China (No. 20100073110016)

978-1-4799-0590-4/13/$31.00 ©2013 IEEE

troduce security problems. Specifically, the main contributions of this paper can be summarized as following: we propose SECO that enables secure and efficient data collaboration in cloud computing, which realized one-to-many encryption paradigm, writing operation, fine-grainedness and collusion resistant simultaneously without any information leakage. Our work is the first attempt to explore secure data collaboration in cloud computing. 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 and realizes the effectiveness and efficiency.

The rest of the paper is organized as follows. Section II discusses related work. In Section III, we introduce the system model and thread model. Section IV presents details of SECO. Section V and VI analyze the security and performance of SECO respectively. Finally, Section VII concludes the whole paper.

## II. Related work

With enterprises outsourcing their data into the cloud, a number of cryptosystems have been used to encrypt these data. Identity-based encryption (IBE) is one of the popular choices. The concept of IBE is proposed by Shamir [9], and the first fully functional IBE schemes are described by Boneh [10] and Cocks [11]. 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 searching for recipient's public key. To reduce the workload on the PKG, Horwitz *et al.* [12] introduced a HIBE scheme with collusion-resistance. Gentry *et al.* [13] presented a HIBE scheme with total collusion resistance and chosen ciphertext security (CCA) in the random oracle model. Later on, Boneh *et al.* [14] introduced an efficient HIBE scheme with selective-ID security without random oracles model under BDH assumption. In recent works, Gentry *et al.* [15] presented a fully secure HIBE scheme that has full security for more than a constant number of levels. However, these HIBE schemes are all one-to-one encryption paradigms.

Furthermore, existing works can be found in the areas of secure outsourced data storage and sharing services. Adya *et al.* [16] 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 [17], Kallahalla *et al.* 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.* proposed SIRIUS [18] 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.* [6] 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.

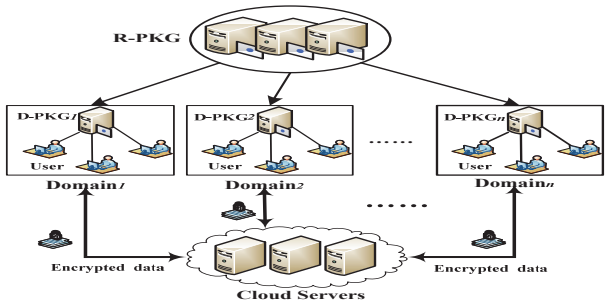


Fig. 1. System model.

## III. Problem Statement

### A. System Model

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

Fig.1 depicts the system model, which is characterized by a two-level HIBE scheme. From the system model, we can see that it consists of a R-PKG and a number of domains. A domain consists of a D-PKG and a number of users who cooperate to complete a project. In practice, R-PKG is a trusted third party which assigns keys, and D-PKG is a team leader who manages all users. 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 CS. All entities in the domain can interact with the CS to dynamically access (read, write, update, etc.) the data so that they can work collaboratively. The R-PKG generates keys for all D-PKGs and system public parameters for all system entities. Furthermore, PKGs and users do not have to be online all the time, whereas the cloud server is always online.

### B. Thread 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 time, but for some benefits the cloud server might try to find out as much secret information as possible.In fact, there are three 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 presented; Attacks can either be active (unauthorized

users who may inject malicious data into 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. Note that, in the adversary model, the communication channels between users and CS are secured under existing protocols, such as SSL.

## IV. The Design of SECO

In order to achieve secure cloud data collaboration, we propose a two-level HIBE scheme SECO. SECO realizes one-to-many encryption paradigm such that an encrypted domain data can be decrypted by many authorized users.

### A. Overview

To embody the users' role in data collaboration, SECO employs a two-level hierarchical architecture in cloud computing. The P-PKG manages a number of D-PKGs while each D-PKG manages a number of domain users. In a domain, a user encrypts data with multiple recipients' public key and stores it to the CS after modifying the data. 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 recipients 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.

### B. Preliminaries

We give some related definitions and assumptions similar to those given in [10][13], which are used in SECO.

***Bilinear Diffie-Hellman (BDH) Parameter Generator:*** As in [13], 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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ 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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfies the following properties:

- Bilinearity: for all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$ where $\mathbb{Z}_q = \{0, 1, 2, ....q-1\}$, have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- Computability: for any $u, v \in \mathbb{G}_1$, there is a polynomial time algorithm to compute $\hat{e}(u, v) \in \mathbb{G}_2$.
- 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 [13], 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 $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ 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 $\mathcal{B}$.

### C. Construction of SECO

SECO is a two-level HIBE system, where $\text{Level}_0$ = {R-PKG} and $\text{Level}_1$ = {D-PKGs}. R-PKG generates private keys for the D-PKGs, and then D-PKG generates private keys for the domain users. The D-PKG has two secret keys: a *private key* and a *master key*. D-PKG uses the two keys to generate private keys for all his domain users. Each user then picks a random seed as his master key. In a domain, each user and D-PKG has a primitive ID, which is an arbitrary string, such as users ID card number and email address. An user's public key is an ID-tuple consisting of the D-PKG's ID and his own ID, i.e, (D-PKG's ID, User's ID) [12]. In addition, the R-PKG also publishes several system parameters used to encrypt and decrypt the cloud data.

Let $K$ be the security parameter used by a BDH parameter generator $\mathcal{IG}$. The ID-tuple for user $E_i$ is ($ID_{dom}$, $ID_i$) where $ID_{dom}$ is the ID of the D-PKG. SECO is specified by the following five randomized algorithms.

***Root Setup:*** The R-PKG takes a security parameter K as input, and outputs *params* (system parameters) and a root master key $s_0$. The system parameters which contain the description of plaintext space $\mathcal{M}$, ciphertext space $C$ and some other parameters are published, while the root master key $s_0$ only is 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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ 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 $\mathbb{Z}_q = \{0, 1, 2, ....q-1\}$, and it sets $Q_0 = s_0 P_0$. Finally, the R-PKG defines four cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_q$ and $H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$ 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 $\mathbb{C} = \mathbb{G}_1^t \times \{0,1\}^n$ where $t$ is the number 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 $s_0 \in \mathbb{Z}_q$.

***The Domain Setup:*** Each D-PKG obtains the system parameters (*params*) from the R-PKG. Each D-PKG randomly picks a $s_{dom} \in \mathbb{Z}_q$ as his master key which will be used to issue private keys to the domain users. Except for $s_{dom}$, each D-PKG is not permitted to generate any other parameters.

***Key Generation:*** The R-PKG uses its master key to generate private keys for D-PKGs while D-PKG uses the system parameters *params* and their secret keys to compute private keys for all the domain users. Let $S_0$ be the identity element of group $\mathbb{G}_1$.

For each D-PKG $E_{dom} \in \text{Level}_1$, it picks a random $s_{dom} \in \mathbb{Z}_q$ as its master key. Given the public $ID_{dom}$, the R-PKG generates the private keys $SK_{dom}$ for each $E_{dom}$. It first calculates $P_{dom} = H_1(ID_{dom}) \in \mathbb{G}_1$; then the R-PKG computes private key for D-PKG as:

$$SK_{dom} = S_0 + s_0 P_{dom}$$

and sends the value $Q_{dom} = s_{dom}P_0$ to $E_{dom}$.

For each D-PKG, it has two secret keys: a master key $s_{dom}$ and a private key $SK_{dom}$. Private key $SK_{dom}$ is used to decrypt all domain data stored in the CS. 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 whose D-PKG is $E_{dom}$, the ID-tuple for user $E_i$ is ($ID_{dom}$, $ID_i$). $E_i$ randomly picks an element $s_i \in \mathbb{Z}_q$ as his master key. $E_{dom}$ generates the private key $SK_i$ for $E_i$.

For each user $E_i$, the D-PKG $E_{dom}$ first calculates $P_i = H_1(ID_{dom}, ID_i) \in \mathbb{G}_1$; then it computes private key for $E_i$ as:

$$SK_i = SK_{dom} + s_{dom}P_i$$

and sends to $E_i$ the value $Q_{dom}$ and $Q_i$ which $Q_i = s_iP_0$. $E_i$ has two secret keys: a master key $s_i$ and a private key $SK_i$. $E_i$ uses $s_i$ and $SK_i$ to decrypt the authorized data in the CS.

***Encryption:*** A user inputs system parameters *params*, plaintext $M \in \mathcal{M}$ and the ID-tuples of the intended data recipients, and then calculates a ciphertext $C \in \mathcal{C}$. After modifying data $D$, the user encrypts it with $t$ recipients' ID-tuple ($ID_{dom}$, $ID_i$) for $1 \le i \le t$ in the same domain.

The user first calculates $P_i = H_1(ID_{dom}, ID_i) \in \mathbb{G}_1$ for $1 \le i \le t$ and $P_{dom} = H_1(ID_{dom})$. Then the user picks a random $\sigma \in \{0,1\}^n$ and sets $r = H_3(\sigma, M)$. Therefore, the ciphertext is set as:

$$C = [rP_0, rP_1, ..., rP_t, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma)]$$

where $g = \hat{e}(Q_0, P_{dom}) \in \mathbb{G}_2$ as before. The user encrypts the data $D$ with $t$ intended recipients in the same domain, and sends the ciphertext $C$ to the CS. Note here, as the D-PKG manages all the domain users, a user can get the recipients' public keys from the D-PKG or other users.

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

Given $C = [U_0, U_1, ..., U_t, V, W]$ be the ciphertext encrypted using the $t$ recipients' ID-tuple ($ID_{dom}$, $ID_i$). Here $U_i = rP_i$, $V = \sigma \oplus H_2(g^r)$ and $W = M \oplus H_4(\sigma)$. If $(U_1, ..., U_t) \notin \mathbb{G}_1^t$, $E_{dom}$ rejects this ciphertext. To decrypt $C$, the D-PKG $E_{dom}$ computes $V \oplus H_2(\hat{e}(U_0, SK_p))$. We observe that:

$$\begin{aligned} & V \oplus H_2(\hat{e}(U_0, SK_{dom})) \\ = \; & V \oplus H_2(\hat{e}(rP_0, S_0 + s_0P_{dom})) \\ = \; & V \oplus H_2(\hat{e}(s_0P_0, rP_{dom})) \\ = \; & V \oplus H_2(\hat{e}(Q_0, P_{dom})^r) = \sigma. \end{aligned}$$

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

Given the ciphertext $C = [U_0, U_1, ..., U_t, V, W]$ to each intended recipient $E_i$ of $1 \le i \le t$. If $(U_1, ..., U_t) \notin \mathbb{G}_1^t$, $E_i$ rejects this ciphertext. To decrypt $C$, the recipient $E_i$ executes the following setups:

- computes $P_i = H_1(ID_{dom}, ID_i)$;
- computes $V \oplus H_2(\hat{e}(U_0, SK_i)/\hat{e}(Q_p, U_i))$ to recover $\sigma$;
- computes $W \oplus H_4(\sigma) = M$.
- sets $r = H_3(\sigma, M)$, tests that $U_i = rP_i$. If not, rejects the ciphertext. Otherwise, outputs $M$ as the decryption of $C$.

Observe that:

$$\begin{aligned} & V \oplus H_2(\hat{e}(U_0, SK_i)/\hat{e}(Q_{dom}, U_i)) \\ = \; & V \oplus H_2(\hat{e}(rP_0, SK_{dom} + s_{dom}P_i)/\hat{e}(Q_{dom}, rP_i)) \\ = \; & V \oplus H_2(\hat{e}(rP_0, s_0P_{dom})\hat{e}(rP_0, s_{dom}P_i)/\hat{e}(Q_{dom}, rP_i)) \\ = \; & V \oplus H_2(\hat{e}(s_0P_0, rP_{dom})\hat{e}(s_{dom}P_0, rP_i)/\hat{e}(Q_{dom}, rP_i)) \\ = \; & V \oplus H_2(\hat{e}(s_0P_0, P_{dom})^r) = \sigma. \end{aligned}$$

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

### D. Signature scheme

SECO also has 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 $E_j$ wants to sign $M$ with his public key ($ID_{dom}$, $ID_j$), he first calculates $P_M = H_1(ID_{dom}, ID_i, M) \in \mathbb{G}_1$ and $Sig(ID_{dom}, ID_i, M) = SK_j + s_jP_M$. Then, $E_j$ sends $[Sig, Q_j]$ as the signature for ($ID_{dom}$, $ID_j$, $M$) where $Q_j = s_jP_0$. When the recipients receive the signature, they confirm the following equation:

$$\hat{e}(P_0, Sig) = \hat{e}(Q_0, P_1)\hat{e}(Q_{dom}, P_j)\hat{e}(Q_j, P_M).$$

In practice, we can use the signature and the aforementioned proposed scheme SECO in a PKI system together.

## V. 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 discuss the security about SECO. By lack of space, we omit the rigorous security proof about the proposed scheme. Then, we provide the realization of fine-grained of access control and collusion resistant.

### A. Security of SECO

In SECO, the message $M$ is encrypted in the form of $C = [rP_0, rP_1, ..., rP_t, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma)]$. Obviously, the adversary need to construct $\sigma$. To obtain $\sigma$, the adversary can recovery $\hat{e}(Q_0, P_{dom})^r$. Although the adversary can obtain some public parameters available. i.e, $Q_0$ and $P_{dom}$, he is unaware of the value of random seed $r$. Therefore, $\hat{e}(Q_0, P_{dom})^r$ cannot be constructed directly. We know that $\hat{e}(Q_0, P_{dom})^r = \hat{e}(U_0, SK_{dom})$. To construct $\hat{e}(Q_0, P_{dom})^r$, the adversary can obtain $\hat{e}(U_0, SK_{dom})$ instead. We recall that, the occurrence of $SK_{dom}$ is in the D-PKG secret key, the adversary cannot obtain the private keys. For this reason, outside adversaries cannot compromise the ciphertext and SECO is secure.

### B. 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. If this data is encrypted with $E_i$ public key, $E_i$ can obtain the corresponding $U_i = rP_i$, and then decrypts this data by calculating: $W \oplus H_4(V \oplus H_2(\hat{e}(U_0, SK_i)/\hat{e}(Q_{dom}, U_i)))$ 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.

### C. Fully collusion secure

In SECO, the data $\mathcal{M}$ is encrypted in the form of $C = [U_0, U_1, ..., U_t, V, W]$ 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 $g = \hat{e}(Q_0, P_{dom}) \in \mathbb{G}_2$ to decrypt ciphertext $C$. Although unauthorized users can obtain $Q_0$ and $P_{dom}$, they are unaware of the random seed $r$, so $\hat{e}(Q_0, rP_{dom})$ cannot to be constructed directly. Beside, unauthorized users observe that: $\hat{e}(Q_0, rP_{dom}) = \hat{e}(U_0, SK_{dom})$. To recover plaintext, unauthorized users may recover $\hat{e}(Q_0, rP_{dom})$ instead of $\hat{e}(U_0, SK_{dom})$. However, since $SK_{dom}$ is only known to R-PKG and D-PKG, unauthorized users also cannot recover $\hat{e}(U_0, SK_{dom})$. Therefore, colluded users cannot recover plaintext. In addition, for an unauthorized ciphertext, there does not exist the corresponding $U_i$ for these unauthorized users. Unauthorized users cannot use decryption algorithm to recover plaintext. Therefore, any of these unauthorized outside the intended recipients will have no idea of the plaintext, even if all of them collude.

## VI. Performance Analysis

In this section, we first evaluate the computation complexity. Then we analyze the communication overhead. At last, we present the storage cost.

### A. 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 *Domain Setup* picks a master key for D-PKG. In *Key Generation*, PKGs generate keys for users. 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. In *key generation*, the R-PKG needs two scalar multiplications to compute $SK_{dom}$ and $Q_{dom}$ for each D-PKG $E_{dom}$, and the D-PKG needs two scalar multiplications to calculate $SK_i$ and $Q_i$ for each domain user $E_i$. In *Encryption*, a user encrypts data with $t$ recipients' public keys. He needs one pairing computation to calculate $\hat{e}(Q_0, P_{dom})$, and $t + 1$ scalar multiplications to compute $rP_i$ for $0 \le i \le t$. 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 user $E_i$ needs two pairing computations to calculate $\hat{e}(U_0, SK_i)$ and $\hat{e}(Q_{dom}, U_i)$. Since $U_0$, $SK_{dom}$ and $SK_i$ are fixed, the D-PKG calculates $\hat{e}(U_0, SK_i)$ once, and $E_i$ calculates $\hat{e}(U_0, SK_i)$ once. From the above analysis, the computation complexity of SECO is acceptable.

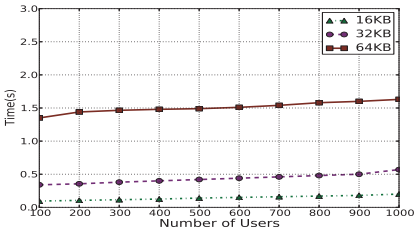


Fig. 2. The cost of encryption algorithm.

We also conduct a thorough experimental evaluation about the time cost of SECO. We calculate the total computing to gain the time cost. The whole experiment system is implemented by Python language on a Windows 7 machine with Core 2 Duo CPU running at 2.0 GHz. We report the average of 100 trials. Fig.2 plots the encryption cost for preparing three kinds of data size as the number of recipients various. From Fig. 2, we see the encryption time grows linearly with the number of the recipients. The time to encrypt 64KB data with 800 recipients approaches to 1.6 seconds, which is an ideal result. Fig.3 plots the D-PKG and user decryption cost as the data size various. In Fig. 3, we notice the cost both by the D-PKG and user 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. To decrypt 64KB data, the D-PKG takes 1 second and user $E_i$ takes 1.5 seconds. The algorithm does the pairing computation for each data, but the pairing computation can be done once at the beginning as the above analysis. The results of our experiments show SECO is light weighted and efficient to be applied in practice.

### B. 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 $U_i$ for every intended data recipient which requires $(t + 1)log|\mathbb{G}_1|$ bits, value of $V$ which requires $n$ bits, and value of $W$ which requires $n$ bits. Thus, the communication cost is given by $(t+1)log|\mathbb{G}_1|+2n$ bits. Table I 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 numbers of the users, $i$ is the number of attributes used in ABE-based scheme [5] and $k$ is the length of keys used in SKC-based scheme [18]. Since the data size is fixed (n), $t$, $k$ and $i$ are varying but have the same order of magnitude as $n$. From Table I, we can see that SECO takes little communication cost. The reason is that every data block is bind with $t$ users KeyID and two secret keys in SKC-based scheme, while in ABE-based scheme, the data owner needs transfer the access structure of the data and other parameters to the cloud. From the above analysis, SECO takes little communication overhead

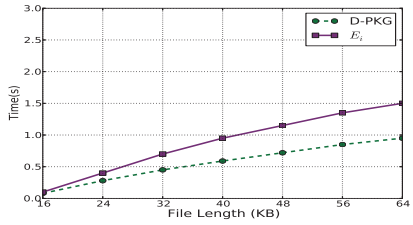


Fig. 3. The cost of decryption algorithm.

TABLE I. COMMUNICATION COST IN ABE-BASED SCHEME, SKC-BASED SCHEME 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 | $(t + 1)log\|\mathbb{G}_1\| + 2n$ |

to achieve secure and efficient data collaboration service in cloud computing.

### C. 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 scheme and ABE-based scheme. 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 II presents the comparative results.

***Ciphertext storage overhead:*** In ABE-based scheme, the size of ciphertext is $O(max(|I|, n))$, with $|I|$ as the number of attributes the ciphertext issued. For SKC-based scheme, 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 $O(n^2)$. In SECO, as depicted in Section IV, the ciphertext is composed of $t$ intended recipients' 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 $O(n)$. Note here, when joining more recipients, it just have one ciphertext which contains more intended recipients' information. From Table II, we can see SECO takes the least ciphertext storage cost.

***Key storage overhead:*** Compared with ABE-based scheme and SKC-based scheme, SECO greatly reduced the key storage overhead of the D-PKG(data owner). In ABE-based scheme and SKC-based scheme, the data owner needs to store every users 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 scheme and SECO. However, users in ABE-based scheme have to store their own access structures with there corresponding secret keys. Therefore, SECO also takes little key storage overhead to achieves data collaboration in cloud computing.

## VII. 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

TABLE II. STORAGE COST IN ABE-BASED SCHEME, SKC-BASED SCHEME AND SECO

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

SECO. SECO employs a two-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. Security analysis show that SECO is secure and can realize fine-grained access control and collusion resistance. In addition, we evaluate the performance of SECO about computation complexity, communication cost and storage cost. The result shows that SECO is low overhead and highly efficient.

## REFERENCES

[1] A. Fox *et al.*, "Above the clouds: A berkeley view of cloud computing," *University of California, Berkeley, Rep. UCB/EECS*, vol. 28, 2009.

[2] M. Arrington, "Gmail disaster: Reports of mass email deletions," *Online at http://www. techcrunch. com/2006/12/28/gmail-disasterreports-ofmass-email-deletions*, 2006.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS'2006*. Alexandria, USA: ACM, pp. 89–98.

[4] G. Wang, Q. Liu, and J. Wu, "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*. Chicago, USA: ACM, 2010, pp. 735–737.

[5] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *INFOCOM'2010*, San Diego, USA, pp. 1–9.

[6] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. Chicago, USA: ACM, 2009, pp. 55–66.

[7] M. Shah, M. Baker, J. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, San Diego, USA, 2007.

[8] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*. Alexandria,USA: ACM, 2007, pp. 598–609.

[9] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1985, pp. 47–53.

[10] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology‹CRYPTO 2001*, 2001, pp. 213–229.

[11] C. Cocks, "An identity based encryption scheme based on quadratic residues," *Cryptography and Coding*, pp. 360–363, 2001.

[12] J. Horwitz and B. Lynn, "Toward hierarchical identity-based encryption," in *Advances in Cryptology‹EUROCRYPT 2002*, pp. 466–481.

[13] C. Gentry and A. Silverberg, "Hierarchical id-based cryptography," *Advances in Cryptology‹ASIACRYPT 2002*, pp. 149–155, 2002.

[14] D. Boneh, X. Boyen, and E. Goh, "Hierarchical identity based encryption with constant size ciphertext," *Advances in Cryptology–EUROCRYPT 2005*, pp. 562–562, 2005.

[15] C. Gentry and S. Halevi, "Hierarchical identity based encryption with polynomially many levels," in *Theory of Cryptography*, 2009.

[16] A. Adya, *et al.*, "Farsite: Federated, available, and reliable storage for an incompletely trusted environment," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 1–14, 2002.

[17] M. Kallahalla, E. Riedel, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, 2003, pp. 29–42.

[18] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing remote untrusted storage." NDSS, 2003.