Fine-Grained Two-Factor Protection Mechanism for Data Sharing in Cloud Storage

Cong Zuo, Jun Shao*, Joseph K. Liu, Guiyi Wei and Yun Ling

***Abstract*—Data sharing in cloud storage is receiving substantial attention in Information Communications Technology, since it can provide users with efficient and effective storage services. To protect the confidentiality of the shared sensitive data, the cryptographic techniques are usually applied. However, the data protection is still posing significant challenges in cloud storage for data sharing. Among them, how to protect and revoke the cryptographic key is the fundamental challenge. To tackle this, we propose a new data protection mechanism for cloud storage, which holds the following properties. 1) The cryptographic key is protected by the two factors. Only if one of the two factors works, the secrecy of the cryptographic key is held. 2) The cryptographic key can be revoked efficiently by integrating the proxy re-encryption and key separation techniques. 3) The data is protected in a fine-grained way by adopting the attribute-based encryption technique. Furthermore, the security analysis and performance evaluation show that our proposal is secure and efficient, respectively.**

***Index Terms*—two-factor, revocability, fine-grained, attribute-based encryption, proxy re-encryption, cloud storage**

I. Introduction

DRIVEN by the attractive on-demand features and advantages, the development and deployment of cloud-based applications have gained tremendous impetus in the industry and research community in recent years. Cloud storage is one of the most successful cloud-based applications [1]–[7], since it matches the huge data sharing demand quite well. Sharing huge data with several data sharers is a cost-consuming task, and the cost on the data owner side is usually proportional to the number of data sharers. While this cost could be reduced to the size of shared data with the help of cloud storage. The only thing the data sharer needs to do is to upload the data to the cloud and grant the access right to the data sharer. After that, data sharers can obtain the data from the cloud instead of the data owner.

Despite the benefits of data sharing in cloud storage, it also introduces many chances to the adversary to access the shared data without authorization. To protect the confidentiality of the shared data, the cryptographic schemes are usually applied. The security of cryptographic schemes stem from the security of underlying cryptographic key. Currently, the cryptographic key is simply stored in the computer in most of existing cryptographic schemes. While it has been reported that the stored keys can be revealed by some viruses [8]. To deal with the key exposure problem, many techniques have been proposed, such as key-insulated public key technique [9], [10], and parallel key insulated public key technique [11], [12].

To the best of our knowledge, the cryptographic key exposure and revocation problems in cloud storage are unrevealed till the work by Liu et al. [13] (named $\mathtt{LLS}^{+}15$ afterwards). In [13], they proposed a novel two-factor data protection mechanism. The cryptographic key is divided into two parts. One is kept in user's computer and the other is stored in a security device (e.g., smart card), which is similar to the e-banking. Only if one of these two parts are kept secret from the adversary, the confidentiality of the cryptographic key is held. Hence, the "two-factor" is named. Furthermore, once the user's security device was either lost or stolen, it could be revoked by using the proxy re-encryption technique. While $\mathtt{LLS}^{+}15$ aims to solve the security problem of the data storage but not the data sharing scenario in cloud computing. Especially, one ciphertext in $\mathtt{LLS}^{+}15$ is essentially an identity-based ciphertext that can be decrypted by only one user but not a group of users as in data sharing scenario. Recently, the data sharing is rising a heated concern. While privacy is still the key concern and an equally striking challenge that reduce the growth of data sharing in cloud [14].

Naive Solution: At first glance, it seems that we can solve the key exposure and revocation problem in the data sharing scenario by simply replacing the IBE scheme in $\mathtt{LLS}^{+}15$ with the ABE scheme. However, it cannot work well for data sharing in cloud storage due to the contradiction between the inefficiency of $\mathtt{LLS}^{+}15$ and the "pay-per-use" nature of cloud storage.

In $\mathtt{LLS}^{+}15$, once the cloud server receives the encrypted data, it encrypts the data by using a public key encryption (PKE) with a public key corresponding to the user's security device. While when it comes to the data sharing scenario where the ciphertext intends for many users, the cloud server would encrypt the data into many ciphertexts under many public keys. Furthermore, even the data is shared with one user, it should be decrypted twice. In particular, one is for the IBE encryption, the other is for the PKE encryption. This makes the solution inefficient.

There is another potential shortcoming for $\mathtt{LLS}^{+}15$. To realize the key revocation, the key generation center in $\mathtt{LLS}^{+}15$ needs to store every security device's secret. When the IBE scheme is directly replaced by the ABE scheme, the size of secret will increase. It would be a burden for the key

C. Zuo, J. Shao, G. Wei and Y. Ling are with the School of Computer and Information Engineering, Zhejiang Gongshang University, Zhejiang, 310018, P.R. China. e-mail: zuocong10@gmail.com, chn.junshao@gmail.com, weigy@zjgsu.edu.cn and yling@zjgsu.edu.cn

* J. Shao is the corresponding author.

J. K. Liu is with Faculty of Information Technology, Clayton Campus, Monash University, VIC 3800, Australia. email: joseph.liu@monash.edu

Manuscript received XX XX, XXXX; revised XX XX, XXXX.

generation center.

Our Technique: To solve the shortcomings of the naive solution, we integrate the attribute-based encryption technique, proxy re-encryption technique, and the key separation technique to remove the use of PKE and the storage of security device's secret in the key generation center while solving key exposure and revocation problems and supporting fine-grained access control. In LLS$^+$15, the ciphertexts are of two formats. One is the IBE ciphertext, the other is the PKE ciphertext. However, all the ciphertexts in our proposed framework are ABE ciphertexts. The main difficulties to make our framework work well are how the old security device is revoked and how the new security device can do decryption properly. To revoke the old security device, we need that the cloud updates the old ciphertexts before sending them to the user by using proxy re-encryption technique. When the user requests the new security device, the user should give a secret to the key generation center to generate a new secret which can be used to decrypt the updated ciphertexts.

Compared to LLS$^+$15, our proposed solution has the following properties.

- LLS$^+$15 is mainly targeted for secure data storage while our main focus is for secure data sharing. These are two different functionalities provided by different types of cryptographic solutions.
- We use a different approach to realize the two-factor technique to solve the key exposure and key revocation problems. As a result, only one kind of ciphertexts exists in our solution, which makes our solution easier to understand and implement. Furthermore, the key generation center in our proposal does not need to store any other secrets except its own private key.
- We explicitly show that how the decryption is proceeded without revealing the secret stored in the security device. While this part is not mentioned in LLS$^+$15.
- When we make use of ABE as IBE, our proposal is more efficient than LLS$^+$15 in terms of computational cost and storage cost. See the details in Section VI.

A. Related Work

In this section, we briefly review the cryptographic schemes with similar functionalities needed in the data sharing scenario and explain why they cannot fully achieve our goals.

1) Cryptographic schemes dealing with the key exposure problem: In 2002, Dodis et al. [9] proposed a key-insulated public key scheme which is the first paper dealing with the private key exposure problem. In such a system, there are two keys. One, named master secret key, is stored in a physically-secure but computationally limited device (security device); and the other is stored in an insecure device (e.g., computer) while can be updated periodically by using the master secret key. The master secret key and the public key remain the same for all the time. Later on, Dodis et al. [10] introduced the key insulated technique into digital signatures. However, the more frequently the private key updates, the more risk of master secret key exposes. To address this problem, in 2006, Hanaoka et al. [11] introduced the parallel key insulated public key encryption. In [11], there are two independent master secret keys, which significantly reduces the risk of master secret key exposure. But the security proof of [11] is obtained in the random oracle model. In 2007, Libert et al. [12] proposed a parallel key insulated public key encryption secure in the standard model. In 2008, Liu et al. [15] applied the key insulated methodology to solve the key exposure problem in ring signature.

In all the above schemes, the security device is used to update every user's private key periodically. Moreover, once the private key is updated, the security device is not involved in the decryption phase. However, according to the requirements in the data sharing scenario for cloud computing, it is desired that the user's private key does not updated in every time period, and the security device should be involved in every decryption phase.

2) Cryptographic schemes with the fine-grained access control: In 2005, Sahai et al. [16] first introduced the notion of attribute based encryption (ABE) and further discussed in [17]. Later, in 2006, Goyal et al. [18] formalized two complementary flavors of ABE: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In a KP-ABE, the private key is associated with a policy (a boolean formula) and the ciphertext is associated with a set of attributes. While the CP-ABE is the opposite case of KP-ABE. In [18], the proposed ABE scheme only supports monotonic access structure. Later on, a new KP-ABE scheme supporting non-monotonic access structure is proposed by Ostrovsky et al. [19]. They also gave a CP-ABE construction based on the technical of [20] where a CP-ABE secure in the random oracle model is proposed. In 2007, Cheung et al. [21] proposed a CP-ABE scheme with non-monotonic access structure in the standard model. Later, many efficient, secure and expressive ABE schemes were proposed [22]–[25]. However, by leveraging these schemes can only achieve the fine-grained access control but not revocability and two-factor protection which are required in the data sharing scenario for cloud computing.

In 2016, Liu et al. [26] proposed a fine-grained two-factor authentication access control system for web-based cloud computing services by using the two-factor and attribute-based signature techniques. They focused on building a user authentication with a privacy-preserving, fine-grained and key-exposure-resisting way. As a result, Liu et al.'s scheme cannot provide fine-grained access control on ciphertexts like we do in this paper, and the key revocation is neither in the scope of [26]. Furthermore, the cost-consuming zero-knowledge proof technique is applied in [26], which is against the "pay-per-use" nature of cloud storage. Hence, the methods used in [26] cannot be applied in our proposal, and new methods realizing the two-factor technique are desired.

3) Cryptographic schemes with revocability: Since the proposed solution in this paper is based on ABE, we would like to review the ABE schemes with revocability. In 2008, Boldyreva et al. [27] introduced a revocable ABE scheme which is extended from their main contribution, a revocable IBE. In their scheme, the non-revoked users need to update

their private keys periodically, which is quite inconvenient. To solve this problem, Attrapadung et al. [28] proposed a new ABE scheme with revocability, where the non-revoked users do not need to update their private keys periodically any more, but the sender needs to know the revocation list. To address this problem, in the same year, Attrapadung et al. [29] proposed another ABE scheme, where the authors conclude that there are two methods to let ABE with revocability, namely, direct and indirect methods. The direct revocation requires the sender to know the revocation list during the encryption process, while the indirect revocation requires the non-revoked users to update their private keys periodically. The advantage of direct revocation over the indirect revocation is that the non-revoked users do not need to update their private keys periodically. On the opposite, the advantage of indirect over the direct revocation is that the sender does not need to know the revocation list. They combined both the advantages of direct and indirect revocation methods and proposed the first hybrid revocable ABE scheme. Later, the fully secure revocable ABE were proposed in [30], [31]. They used composite order bilinear groups to achieve fully secure which is less efficient than the prime order bilinear groups. However, in our proposal, we only need to issue a new security device to revoke the old key without updating all other security devices, and we do not need to maintain a revocation list for all the revoked security devices. As a result, we believe that our scheme provides an alternative approach to tackle the long-existing revocability problem of ABE.

Another cryptographic privilege supporting revocability is proxy re-encryption (PRE) which was proposed by Blaze et al. in [32]. In a proxy re-encryption scheme, a proxy (e.g., the semi-trusted cloud) can transform a ciphertext for a user into another ciphertext that another user can decrypt while the proxy can learn nothing but the length of the ciphertext. PRE can be formalized into two categories in terms of the direction of transformation: bidirectional and unidirectional. In bidirectional PRE, the proxy can transform the ciphertext from one user into another user and vice versa. In unidirectional PRE, the proxy can only convert in one direction. To increase the efficiency and security of PRE, many schemes were proposed [33]–[37]. We state that combining the ABE and PRE techniques, the resulting solution still cannot satisfy the desired requirements in the data sharing scenario for cloud computing. In particular, it cannot support the two-factor protection.

B. Organization

The remaining paper is organized as follows. In Section II, we give the models and design goals for our scheme. In Section III, we give the necessary background for our complexity assumption. After that, we present our fine-grained two-factor data protection mechanism for cloud storage in Section IV and its security analysis in Section V. We also give the performance evaluation for our scheme in VI. Finally, we give the conclusion in Section VII.

II. Framework and Design Goal

In this section, we formalize the system model and attack models considered in this paper, and identify the design goals.

A. Framework Architecture

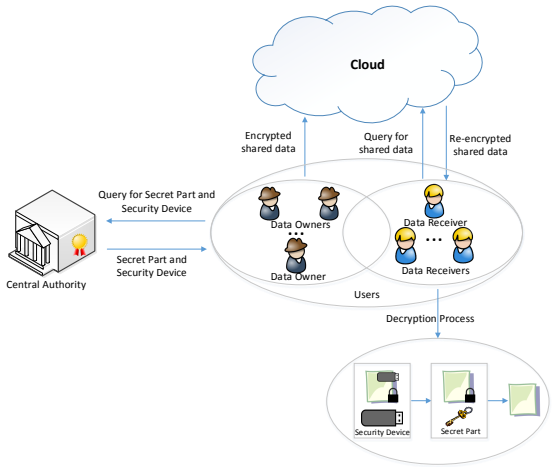


Fig. 1: Architecture of our framework

Fig. 1 shows the architecture of our CP-ABE based fine-grained two-factor data protection framework. There are four main entities in our framework: Central Authority (CA), Cloud, Data Owners (DOs) and Data Receivers (DRs)[1].

- Central Authority (CA): The CA is a trusted party which is responsible for issuing the cryptographic key for every user according to their attribute set and then splitting it into two parts (two-factor): One, called as Secret Part Key (SPK), is assumed to be stored in a potential-insecure place (e.g., computer). The other, named as Security Device Key (SDK) is stored in a physically-secure but computationally limited device (security device). Furthermore, the CA is also responsible for updating every user's security device (and the corresponding SDK). Specially, in the SDK update phase, the CA generates a new SDK that is stored in a security device and the corresponding re-encryption key that will be sent to the cloud. Fig. 2 shows the process of SDK update. We will give more details in Section IV.

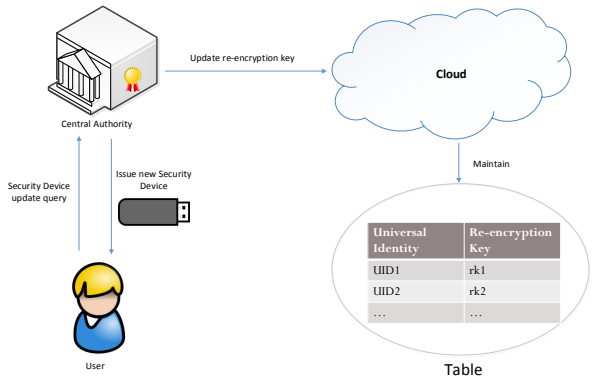


Fig. 2: The process of SDK update

[1]Both DOs and DRs are users.

Note that the re-encryption key is used to update the ciphertexts to make the new SDK work, while the generation of the re-encryption key requires the information of the old SDK. As mentioned before, one of the advantages of our proposal is that the CA does not need to store any secrets for users. In this case, the way that the CA simply issue an update key to update the old SDK cannot work due to the absent of the old SDK (the security device may be stolen or lost). To solve the problem, we use SPK to retrieve SDK instead. (See Section IV for more details.)

- Cloud: The cloud is a semi-trust party that stores all encrypted shared data and maintains a table $Table$ containing the users' universal identity (UID) and corresponding re-encryption key. When a DR queries for the shared data, the cloud acts as a proxy to re-encrypt the encrypted shared data by using DR's corresponding re-encryption key and returns the re-encrypted shared data to DR.
- Data Owner (DO): A DO is a user who wants to share data with other users (DRs). All the shared data are encrypted by using CP-ABE according to the access policy.
- Data Receiver (DR): A DR is a user who can receive the shared data from the cloud. When a DR wants to retrieve the shared data, the cloud firstly does re-encryption and then returns the resulting re-encrypted ciphertext. The re-encrypted ciphertext can be decrypted by using DR's own SPK and SDK, if DR's attribute set satisfies the access policy of the shared data. Note that SDK is never revealed out of the security device during the decryption, while a partial decryption process using SDK would be executed in the security device. Once the security device is lost or stolen, DR can revoke it and obtain a new security device through interacting with CA.

B. Definition of FGTFDP: Fine-Grained Two-Factor Data Protection Mechanism for Cloud Storage System

The Fine-Grained Two-Factor Data Protection mechanism for cloud storage system (FGTFDP) consists of the following algorithms:

- $\text{INIT}(\lambda) \longrightarrow (param, msk)$: On inputting a security parameter λ, the algorithm (run by the CA) outputs the public parameter $param$ and the master secret key msk.
- $\text{KEYGEN}(param, msk, S_i) \longrightarrow (\texttt{SPK}_i, \texttt{SDK}_i)$: On inputting the public parameters $param$, the master secret key msk and a user $\texttt{UID}_i$ with attribute set S_i, the algorithm (run by the CA) outputs the secret part key $\texttt{SPK}_i$ and security device key $\texttt{SDK}_i$ for the user $\texttt{UID}_i$.
- $\text{REVOCATION}(param, msk, \texttt{UID}_i) \longrightarrow (\texttt{SDK}_i, \texttt{rk}_i)$: It is an interactive algorithm performed between the user who wants to revoke his/her security device and the CA. At the end of this algorithm, the CA outputs the new security device key $\texttt{SDK}_i$ for the user and the corresponding re-encryption key $\texttt{rk}_i$ for the cloud.
- $\text{DATAUPLOAD}(param, policy, m) \longrightarrow \texttt{CT}$: On inputting the public parameters $param$, the access structure $policy$ and the message m, the algorithm (run by the DO) outputs the ciphertext CT and uploads it to the cloud.
- $\text{DATADOWNLOAD}(\texttt{CT}, \texttt{rk}_i) \longrightarrow \texttt{CT}$ or $\texttt{CT}_{r,i}$: On inputting the ciphertext CT and a DR $\texttt{UID}_i$'s corresponding re-encryption key $\texttt{rk}_i$, the algorithm (run by the cloud) first checks if $\texttt{rk}_i$ exists. If not (the DR's security device has never been updated), the algorithm outputs the CT to the DR. Otherwise, the algorithm returns re-encrypted ciphertext $\texttt{CT}_{r,i}$ to the DR.
- $\text{DATAREVEAL}(\texttt{CT}_{r,i}, \texttt{SPK}_i, \texttt{SDK}_i) \longrightarrow m$: On inputting $\texttt{CT}_{r,i}$[2], the secret part key $\texttt{SPK}_i$ and the security device key $\texttt{SDK}_i$. The algorithm (run by the DR) outputs m. Note that $\texttt{SDK}_i$ is never revealed out of the security device in this algorithm.

Correctness: We require that an FGTFDP scheme is correct, e.g. DATAREVEAL algorithm correctly reveals $\texttt{CT}_{r,i}$ of an access structure $policy$ with a security device $\texttt{SDK}_i$ on S_i, when S_i satisfies the access structure $policy$ and the generation $\texttt{gen}_i$ of $\texttt{SDK}_i$ and $\texttt{CT}_{r,i}$ are the same.

More specially, for all messages m, and any attribute set S_i and access structures $policy$ with $S_i \in policy$, any pair $(param, msk)$ output from $\text{INIT}(\lambda)$, any secret pair $(\texttt{SPK}_i, \texttt{SDK}_i)$ output from $\text{KEYGEN}(param, msk, S_i)$ or updated pair $(\texttt{SDK}_i, \texttt{rk}_i)$ output from $\text{REVOCATION}(param, msk, \texttt{UID}_i)$, any ciphertext CT output from $\text{DATAUPLOAD}(param, policy, m)$, any $\texttt{CT}_{r,i}$ output from $\text{DATADOWNLOAD}(\texttt{CT}, \texttt{rk}_i)$, and the generation $\texttt{gen}_i$ of $\texttt{SDK}_i$ and $\texttt{CT}_{r,i}$ are the same, it is true that: $\text{DATAREVEAL}(\text{DATADOWNLOAD}(\texttt{CT}, \texttt{rk}_i), \texttt{SPK}_i, \texttt{SDK}_i) = m$.

C. Security Model

In our system, we assume that the CA is a trusted party. As the existing literatures dealing with the security in cloud computing [13], [38], we assume that the cloud is honest-but-curious. In particular, the cloud could only launch passive attacks and would not collude with other users. Furthermore, it will definitely follow the proposed protocol. In our proposal, it will update the ciphertexts before sending them to users. We also assume that the security device is tamper-resistant.

In this paper, we consider the following attack model, which is mainly due to the two-factor requirement.

- **Type-1**: The adversary can obtain SPK, revoked SDK's of the target user, while it cannot have the corresponding current SDK.
- **Type-2**: The adversary can obtain all SDK's for all the time of the target user, while it cannot have the corresponding SPK.

We will give more details related to the security model in Section V.

D. Design Goal

Considering the aforementioned framework architecture and security model, our design goal is to present fine-grained two-factor data protection mechanism for cloud storage with revocability. Specifically, the following design goals should be satisfied:

[2] $\texttt{CT}_{r,i}$ and CT are of the same format, hence we only use $\texttt{CT}_{r,i}$ for simplicity.

- *Fine-grained.* The fine-grained access control is always a desired requirement for the data sharing scenario.
- *Revocable.* The secrets used to decrypt the ciphertexts can be revoked.
- *Secure.* The proposed system should be resilient to **Type-1** and **Type-2** attacks.
- *Practical.* The underlying encryption, decryption and SDK update procedures should be effective and efficient.

III. Preliminaries

In this section, we introduce the cryptographic technique of bilinear pairing and related complexity assumption, which will be used in the security proof of our system.

A. Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two cyclic groups of the same big prime order p, and g be a generator of $\mathbb{G}$. let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a pairing, i.e. a map satisfies the following properties:

1) Bilinearity. $e(g^a, g^b) = e(g, g)^{ab}$ for any $a, b \in \mathbb{Z}_p^*$.
2) Non-degeneracy. $e(g, g)$ is a generator of group $\mathbb{G}_1$.
3) Computability. e can be computed efficiently.

We denote BSetup as an algorithm that takes as input the security parameter λ and outputs the parameters for a bilinear map as $(p, g, \mathbb{G}, \mathbb{G}_1, e)$.

B. Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map, both $\mathbb{G}$ and $\mathbb{G}_1$ are cyclic groups of prime order p. Choose a random generator g of $\mathbb{G}$ and random a, b, c, z from $\mathbb{Z}_p^*$. The decisional Bilinear Diffie-Hellman(DBDH) Assumption is to distinguish between the tuples of the form $(g, g^a, g^b, g^c, e(g, g)^{abc})$ and $(g, g^a, g^b, g^c, e(g, g)^z)$. An algorithm $\mathcal{A}$ has non-negligible advantage ϵ in solving DBDH if

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^z) = 1]| \geq \epsilon.$$

IV. FGTFDP: Fine-Grained Two-Factor Data Protection Mechanism for Cloud Storage System

In this section, we present our construction of FGTFDP by using CP-ABE and PRE. There exist six algorithms in our proposed solution. The processes related to CP-ABE is almost the same as that in [21]. Other ABE schemes, likes that in [22], [24], [25] with more expressive access structure, could also be adapted to our framework.

- Init$(1^\lambda) \longrightarrow (param, msk)$: On inputting the security parameter λ, the CA generates all public parameters and master secret key used throughout the execution of the system. The public parameters are shared with all parties participating into the system, while the master secret key is kept secret to the CA.
  Let the set of attributes in our solution be $N = \{1, ..., n\}$ for some natural number n. We refer to attributes j and their negations $\bar{j}$ as literals. As [21], we consider the access structures that consist of a single **AND** gate whose inputs are literals. This is denoted as $\bigwedge_{j \in I} \tilde{j}$, where $I \subseteq N$ and every $\tilde{j}$ is a literal (e.g., j or $\bar{j}$). For the more expressive access structure, we may apply other CP-ABE schemes.
  The CA runs $\texttt{BSetup}(1^\lambda)$ to obtain $(p, g, \mathbb{G}, \mathbb{G}_1, e)$, where λ is the security parameter. It also defines $\mathbf{t}_1 = (t_{1,1}, \cdots, t_{1,3n})$ and $\mathbf{T}_1 = (T_{1,1}, \cdots, T_{1,3n})$, where $t_{1,j}$ for $j \in [1, 3n]$ are randomly selected from $\mathbb{Z}_p^*$, and $T_{1,j} = g^{t_{1,j}}$ for each $j \in [1, 3n]$. In order to achieve revocation property, we need an additional $\mathbf{t}$, named $\mathbf{t}_{\texttt{gen}}$. At the beginning, we have that $\mathbf{t}_{\texttt{gen}} = \mathbf{t}_1$, where $\texttt{gen} = 1$. It then picks $\alpha \in \mathbb{Z}_p^*$ at random and computes $A = e(g, g)^\alpha$. The resulting master secret key is $msk = (\alpha, \mathbf{t}_1, \mathbf{t}_{\texttt{gen}})$ and the public parameters $param$ to be $param = (e, g, p, \mathbb{G}, \mathbb{G}_1, A, \mathbf{T}_1)$. (We refer readers to [21] for more details.)
- KeyGen$(param, msk, S_i) \longrightarrow (\texttt{SPK}_i, \texttt{SDK}_i)$: In this algorithm, the CA will generate the secrets (including $\texttt{SPK}_i$ and $\texttt{SDK}_i$) for a registered user $\texttt{UID}_i$ according to his/her attribute set S_i. As [21], we consider a negative attribute if $j \notin S_i$.
  First, the CA selects random r_i and $r_{i,j}$ from $\mathbb{Z}_p^*$ for $j \in [1, n-1]$ and computes $r_{i,n} = r_i - \sum_{j=1}^{n-1} r_{i,j} \bmod p$. Since r_i and $r_{i,j}$ for $j \in [1, n-1]$ are random elements in $\mathbb{Z}_p^*$, we have that $r_{i,n}$ is random in $\mathbb{Z}_p^*$. Second, the CA computes the following values.
  - $D_i = g^{\alpha - r_i}$
  - For each $j \in N$, $D_{i,j} = \begin{cases} g^{\frac{r_{i,j}}{t_{1,j}}}, & \text{if } j \in S_i; \\ g^{\frac{r_{i,j}}{t_{1,n+j}}}, & \text{if } j \notin S_i. \end{cases}$
  - For every $j \in N$, $F_{i,j} = g^{\frac{r_{i,j}}{t_{1,2n+j}}}$.

  The secret key for $\texttt{UID}_i$ is defined as $\texttt{sk}_i = (D_i, \{D_{i,j}, F_{i,j}\})$ for $j \in N$. The CA further splits $\texttt{sk}_i$ into two parts: $\texttt{SPK}_i = D_i$ that is given to the user directly and $\texttt{SDK}_i = \{D_{i,j}, F_{i,j}\}$ that is embedded into a security device and sent to the user.
- Revocation$(param, msk, \texttt{UID}_i) \longrightarrow (\texttt{SDK}_i, \texttt{rk}_i)$: The user will obtain a new SDK from the CA, when the old security device is either lost or stolen. The process is specified as follows.
  - The user $\texttt{UID}_i$ with attribute set S_i issues the revocation query to the CA via a secure and authenticated channel with a $list$ $\boxed{\texttt{UID}_i \mid S_i \mid \texttt{SPK}_i \mid \texttt{gen}_i}$, where $\texttt{SPK}_i$ is the secret part key (D_i) directly sent to the user $\texttt{UID}_i$, and $\texttt{gen}_i$ is the generation of the security device.
  - Upon receiving the query from the user $\texttt{UID}_i$, the CA checks whether $\texttt{gen}_i < \texttt{gen}$ where gen is the current generation of $\mathbf{t}_{\texttt{gen}}$. If not, the CA chooses random $\mathbf{t}_{\texttt{gen}} = (t_{\texttt{gen},1}, \cdots, t_{\texttt{gen},3n})$ from $\mathbb{Z}_p^{*3n}$ and increases gen by one. Otherwise, the CA continues to do next steps.
  - The CA recovers g^{r_i} for the user $\texttt{UID}_i$ from $g^\alpha / \texttt{SPK}_i = g^\alpha / D_i = g^\alpha / g^{\alpha - r_i}$.
  - The CA chooses new random $r_{i,j}$ from $\mathbb{Z}_p^*$ for $j \in [1, n-1]$, then sets $g^{r_{i,n}} = g^{r_i} / \prod_{j=1}^{n-1} g^{r_{i,j}}$.
  - The CA computes new $\{D_{i,j}, F_{i,j}\}$ as follows.

∗ For each $j \in N$,

$$D_{i,j} = \begin{cases} (g^{r_{i,j}})^{\frac{1}{t_{\mathtt{gen},j}}} = g^{\frac{r_{i,j}}{t_{\mathtt{gen},j}}}, & \text{if } j \in S_i; \\ (g^{r_{i,j}})^{\frac{1}{t_{\mathtt{gen},n+j}}} = g^{\frac{r_{i,j}}{t_{\mathtt{gen},n+j}}}, & \text{if } j \notin S_i. \end{cases}$$

∗ For every $j \in N$,

$$F_{i,j} = (g^{r_{i,j}})^{\frac{1}{t_{\mathtt{gen},2n+j}}} = g^{\frac{r_{i,j}}{t_{\mathtt{gen},2n+j}}}.$$

– The CA embeds $\{D_{i,j}, F_{i,j}\}$ into a new security device as the new $\mathtt{SDK}_i$ and gives it to the user together with the current generation gen as his/her new generation $\mathtt{gen}_i$.
– The CA also generates the re-encryption key

$$\begin{aligned} \mathtt{rk}_i &= (rk_{i,1}, \cdots, rk_{i,3n}) \\ &= \left(\frac{t_{\mathtt{gen},1}}{t_{1,1}}, \cdots, \frac{t_{\mathtt{gen},3n}}{t_{1,3n}}\right) \end{aligned}$$

and gives it to Cloud together with $\mathtt{UID}_i$ via a secure and authenticated channel. The cloud will update the table $Table$ with $\mathtt{UID}_i$ and $\mathtt{rk}_i$.

- DataUpload$(param, policy, m) \longrightarrow \mathtt{CT}$: In this algorithm, the DO will encrypt the shared data according to his/her sharing policy, and upload the resulting ciphertexts to the cloud. For simplicity, we assume that the shared data m belonging to $\mathbb{G}_1$. For other message spaces, we can simply apply the hybrid encryption method. We also assume $policy$ (the access structure) is an **AND** gate $W = \bigwedge_{j \in I} \tilde{j}$, where $I \subseteq N$. Again, to support more expressive access structure, we can apply other CP-ABE schemes instead of the one in [21].
The DO works as follows:
  – Select a random $s \in \mathbb{Z}_p^*$ and set $C = m \cdot A^s$ and $C' = g^s$.
  – For each $j \in I$, computes

$$C_j = \begin{cases} T_{1,j}^s, & \text{if } \tilde{j} = j \\ T_{1,n+j}^s, & \text{if } \tilde{j} = \bar{j} \end{cases}$$

  – For each $j \in N \backslash I$, computes $C_j = T_{1,2n+j}^s$.

  The resulting ciphertext is $\mathtt{CT} = (W, C, C', \{C_j\}_{j=1}^n)$. At last, the DO sends CT to the cloud.
- DataDownload$(\mathtt{CT}, \mathtt{rk}_i) \longrightarrow \mathtt{CT}$ or $\mathtt{CT}_{r,i}$: In this algorithm, the DR will download the shared data from the cloud in this algorithm. The details are as follows.
  – The DR queries the cloud for the ciphertext with universal identity $\mathtt{UID}_i$.
  – The cloud first checks if this DR's security device has been updated. If not, the cloud returns CT to the DR directly.
  – Otherwise, the cloud will transform the ciphertext $\mathtt{CT} = (W, C, C', \{C_j\}_{j=1}^n)$ under the DR's re-encryption key $\mathtt{rk}_i$ as follows. Note that $W = \bigwedge_{j \in I} \tilde{j}$.
    ∗ It chooses a random s' from $\mathbb{Z}_p^*$.
    ∗ It computes $C_r = C \cdot A^{s'} (= m \cdot A^{s+s'})$ and $C_r' = C' \cdot g^{s'} (= g^{s+s'})$.
    ∗ For each $j \in I$, it computes [3]

$$C_{r,j} = \begin{cases} (C_j \cdot T_{1,j}^{s'})^{\mathtt{rk}_{i,j}} = \left(g^{t_{1,j} \cdot (s+s')}\right)^{\frac{t_{\mathtt{gen}_i,j}}{t_{1,j}}} \\ = g^{t_{\mathtt{gen}_i,j} \cdot (s+s')}, \quad \text{if } \tilde{j} = j; \\ (C_j \cdot T_{1,n+j}^{s'})^{rk_{i,n+j}} = \left(g^{t_{1,n+j} \cdot (s+s')}\right)^{\frac{t_{\mathtt{gen}_i,n+j}}{t_{1,n+j}}} \\ = g^{t_{\mathtt{gen}_i,n+j} \cdot (s+s')}, \quad \text{if } \tilde{j} = \bar{j}. \end{cases}$$

    ∗ For each $j \in N \backslash I$, it computes

$$\begin{aligned} C_{r,j} &= \left(C_j \cdot T_{1,2n+j}^{s'}\right)^{\mathtt{rk}_{i,2n+j}} \\ &= \left(g^{t_{1,2n+j} \cdot (s+s')}\right)^{\frac{t_{\mathtt{gen}_i,2n+j}}{t_{1,2n+j}}} \\ &= g^{t_{\mathtt{gen}_i,2n+j} \cdot (s+s')}. \end{aligned}$$

    The re-encrypted ciphertext for the DR with universal $\mathtt{UID}_i$ is $\mathtt{CT}_{r,i} = (W, C_r, C_r', \{C_{r,j}\}_{j=1}^n)$. Then, the cloud returns $\mathtt{CT}_{r,i}$ to DR.
    Note that the original ciphertext and re-encrypted ciphertext have the same format, hence, we simply consider $s + s' \bmod q$ as a new s.
- DataReveal$(\mathtt{CT}_{r,i}, \mathtt{SPK}_i, \mathtt{SDK}_i) \longrightarrow m$: In this algorithm, the DR will decrypt the received (re-encrypted) ciphertext $\mathtt{CT}_{r,i} = (W, C_r, C_r', \{C_{r,j}\}_{j=1}^n)$ [4] by using his/her secret part $\mathtt{SPK}_i$ and security device $\mathtt{SDK}_i$. Firstly, the DR parses $\mathtt{SPK}_i$ and $\mathtt{SDK}_i$ as $(D_i, \{D_{i,j}, F_{i,j}\}_{j=1}^n)$ and $\mathtt{CT}_{r,i}$ as $(W, C_r, C_r', \{C_{r,j}\}_{j=1}^n)$ where $W = \bigwedge_{j \in I} \tilde{j}$.
  – Security Device Decryption Phase: On inputting the ciphertext $\mathtt{CT}_{r,i}$, the security device does the following steps.
    ∗ For each $j \in I$, it computes the pairing $e(C_{r,j}, D_{i,j})$.
      · If $\tilde{j} = j$ and $j \in S_i$ then

$$\begin{aligned} e(C_{r,j}, D_{i,j}) &= e(g^{t_{\mathtt{gen}_i,j} \cdot s}, g^{r_{i,j}/t_{\mathtt{gen}_i,j}}) \\ &= e(g,g)^{r_{i,j} \cdot s}. \end{aligned}$$

      · Similarly, if $\tilde{j} = \bar{j}$ and $j \notin S_i$, then

$$\begin{aligned} e(C_{r,j}, D_{i,j}) &= e(g^{t_{\mathtt{gen}_i,n+j} \cdot s}, g^{r_{i,j}/t_{\mathtt{gen}_i,n+j}}) \\ &= e(g,g)^{r_{i,j} \cdot s}. \end{aligned}$$

    ∗ For each $j \notin I$, it computes the paring

$$\begin{aligned} e(C_{r,j}, F_{i,j}) &= e(g^{t_{\mathtt{gen}_i,2n+j} \cdot s}, g^{r_{i,j}/t_{\mathtt{gen}_i,2n+j}}) \\ &= e(g,g)^{r_{i,j} \cdot s}. \end{aligned}$$

    ∗ At last, it outputs $R = \prod_{j=1}^n e(g,g)^{r_{i,j} \cdot s}$.
  – Secret Part Decryption Phase: With the output from the security device, DR can obtain the shared data m by using $\mathtt{SPK}(= D_i)$ as follows:
    ∗ It computes $B = e(C_r', D_i) \cdot R$,
    ∗ Then, it outputs $m = C_r / B$.

  Correctness: If the attribute set S_i of $\mathtt{SDK}_i$ satisfies the access structure $policy$ and $\mathtt{SDK}_i$ and $\mathtt{CT}_{r,i}$ are in the same generation $\mathtt{gen}_i$ (Note that the revoked $\mathtt{SDK}_i$ cannot

[3] We use $\mathtt{gen}_i$ (rather than gen) to distinguish different users.
[4] If $\mathtt{CT}_{r,i} = \mathtt{CT}$, then $\mathbf{t}_{\mathtt{gen}_i} = \mathbf{t}_1$ which means it is the original ciphertext.

decrypt properly.), then DR can recover every $e(g,g)^{r_{i,j}\cdot s}$ for $j \in N$ as stated in DATAREVEAL. Therefore, we have that

$$\begin{aligned} B &= e(C'_r, D_i) \cdot \prod_{j=1}^{n} e(g,g)^{r_{i,j}\cdot s} \\ &= e(g^s, g^{\alpha - r_i}) \cdot e(g,g)^{r_i \cdot s} \\ &= e(g,g)^{\alpha \cdot s} \end{aligned}$$

V. SECURITY ANALYSIS

Before giving the security analysis of our proposed solution, we would like to give the details of the security models.

A. Security models

We say our proposed solution is secure against **Type-1** attacks if no probabilistic polynomial-time adversary having the non-negligible advantage in the following game played a challenger $\mathcal{C}$.

- **Init:** $\mathcal{A}$ chooses the challenge access structure W^* and challenge generation $\mathtt{gen}^*$, and sends them to $\mathcal{C}$.
- **Setup:** $\mathcal{C}$ runs algorithm INIT and gives $\mathcal{A}$ the public parameters. $\mathcal{C}$ also sets $\mathtt{gen} = 1$.
- **Phase 1:** $\mathcal{A}$ is allowed to query the following oracles adaptively.
  - $\mathcal{O}_{spk}$: On inputting $(S_i, \mathtt{gen}_i)$, $\mathcal{C}$ returns the corresponding $\mathtt{SPK}_i$ if $\mathtt{gen}_i \leq \mathtt{gen}$; otherwise, $\mathcal{C}$ returns $\perp$. This oracle simulates that the secret part stored in computer is revealed.
  - $\mathcal{O}_{sdk}$: If $\mathtt{gen}_i \leq \mathtt{gen} < \mathtt{gen}^*$, or $\mathtt{gen}_i = \mathtt{gen} = \mathtt{gen}^*$ and S_i that does not satisfy W^*, $\mathcal{C}$ returns the corresponding $\mathtt{SDK}_i$; otherwise, $\mathcal{C}$ returns $\perp$.
  - $\mathcal{O}_{update}$: $\mathcal{C}$ increases $\mathtt{gen}$ by one and updates the corresponding $\mathbf{t}_{\mathtt{gen}}$. This oracle simulates that $param$ is updated in case that the revocation request is queried, and this oracle is allowed to be queried for $\mathtt{gen}^* - 1$ times at most.
  - $\mathcal{O}_{cipher}$: On inputting $(\mathtt{CT}_i, S_i, \mathtt{gen}_i)$, $\mathcal{C}$ returns the corresponding ciphertext $\mathtt{CT}'_i$ if $\mathtt{gen}_i \leq \mathtt{gen}$; otherwise, $\mathcal{C}$ returns $\perp$. Note that $\mathtt{CT}_i$ should be always corresponding to generation $\mathtt{gen} = 1$. This oracle simulates that the ciphertext the user (DR) receives is revealed to the adversary.
- **Challenge:** $\mathcal{A}$ sends two messages m_0^*, m_1^* of equal length to $\mathcal{C}$. $\mathcal{C}$ firstly queries $\mathcal{O}_{update}$ for $\mathtt{gen}^* - \mathtt{gen}$ times, and then encrypts m_b^* as the challenge ciphertext $\mathtt{CT}^*$ to $\mathcal{A}$, where b is chosen randomly from $\{0,1\}$, and $\mathtt{CT}^*$ is corresponding to the newest $\mathbf{t}_{\mathtt{gen}^*}$.
- **Phase 2:** Identical to that in Phase 1.
- **Guess:** $\mathcal{A}$ outputs a guess b' of b.

Similarly, we can define the security model for **Type-2** attacks.

- **Init:** Identical to that in **Type-1** case.
- **Setup:** Identical to that in **Type-1** case.
- **Phase 1:** $\mathcal{A}$ is allowed to query the following oracles adaptively.
  - $\mathcal{O}_{spk}$: On inputting $(S_i, \mathtt{gen}_i)$, $\mathcal{C}$ returns the corresponding $\mathtt{SPK}_i$ if $\mathtt{gen}_i \leq \mathtt{gen}$ and S_i does not satisfy W^*; otherwise, $\mathcal{C}$ returns $\perp$.
  - $\mathcal{O}_{sdk}$: If $\mathtt{gen}_i \leq \mathtt{gen}$, $\mathcal{C}$ returns the corresponding $\mathtt{SDK}_i$; otherwise, $\mathcal{C}$ returns $\perp$.
  - $\mathcal{O}_{update}$: Identical to that in **Type-1** case.
  - $\mathcal{O}_{cipher}$: Identical to that in **Type-1** case.
- **Challenge:** $\mathcal{A}$ sends two messages m_0^*, m_1^* of equal length to $\mathcal{C}$. $\mathcal{C}$ firstly queries $\mathcal{O}_{update}$ for $\mathtt{gen}^* - \mathtt{gen}$ times, and then encrypts m_b^* as the challenge ciphertext $\mathtt{CT}^*$ to $\mathcal{A}$, where b is chosen randomly from $\{0,1\}$, and $\mathtt{CT}^*$ is corresponding to the newest $\mathbf{t}_{\mathtt{gen}^*}$.
- **Phase 2:** Identical to that in Phase 1.
- **Guess:** $\mathcal{A}$ outputs a guess b' of b

B. Security Proofs

Theorem 1: Our proposal is secure against the **Type-1** attacks.

Proof: If there is an adversary $\mathcal{A}$ that can successfully launch **Type-1** attacks on our proposed solution, we can build an algorithm $\mathcal{B}$ breaking the DBDH assumption.

- **Init:** $\mathcal{B}$ receives the challenge gate $W^* = \bigwedge_{j \in I} \tilde{j}$ and $\mathtt{gen}^*$ from $\mathcal{A}$.
- **Setup:** $\mathcal{B}$ is given an instance of the DBDH tuple (g, g^a, g^b, g^c, Z), where g is a random generator of $\mathbb{G}$ and Z is either $e(g,g)^{abc}$ or $e(g,g)^z$ $(a,b,c,z \xleftarrow{R} \mathbb{Z}_p^*)$. Then $\mathcal{B}$ sets $A = e(g,g)^{ab}$. For each $j \in N$, $\mathcal{B}$ chooses random $\eta_{\mathtt{gen}^*,j}, \theta_{\mathtt{gen}^*,j}, \vartheta_{\mathtt{gen}^*,j} \in \mathbb{Z}_p^*$. Now $\mathcal{B}$ constructs $T_{\mathtt{gen}^*,j}, T_{\mathtt{gen}^*,n+j}$ and $T_{\mathtt{gen}^*,2n+j}$ as in Table I.

TABLE I: $\mathbf{T}_{\mathtt{gen}^*}$'s simulation in the secure game

	$j \in I$		$j \notin I$
	$\tilde{j} = j$	$\tilde{j} = \bar{j}$	
$T_{\mathtt{gen}^*,j}$	$g^{\eta_{\mathtt{gen}^*,j}}$	$g^{b\cdot\eta_{\mathtt{gen}^*,j}}$	$g^{b\cdot\eta_{\mathtt{gen}^*,j}}$
$T_{\mathtt{gen}^*,n+j}$	$g^{b\cdot\theta_{\mathtt{gen}^*,j}}$	$g^{\theta_{\mathtt{gen}^*,j}}$	$g^{b\cdot\theta_{\mathtt{gen}^*,j}}$
$T_{\mathtt{gen}^*,2n+j}$	$g^{b\cdot\vartheta_{\mathtt{gen}^*,j}}$	$g^{b\cdot\vartheta_{\mathtt{gen}^*,j}}$	$g^{\vartheta_{\mathtt{gen}^*,j}}$

If $\mathtt{gen}^* \neq 1$, $\mathcal{B}$ further generates the public parameters $param$ except A and the corresponding master secret keys as that in the real execution. Note that in this case $\mathcal{B}$ knows $\mathbf{t}_1$.

- **Phase 1:** $\mathcal{A}$ is allowed to query the following oracles adaptively.
  - $\mathcal{O}_{spk}$: There are two cases in this oracle. Note that $\mathtt{gen}_i$ should follow the restrictions as described in the security model.
    - If S_i does not satisfy W^*, there must exist $k \in I$ satisfying that: either $k \in S_i$ and $\tilde{k} = \bar{k}$, or $k \notin S_i$ and $\tilde{k} = k$. $\mathcal{B}$ chooses such k. Without loss of generality, we assume that $k \notin S_i$ and $\tilde{k} = k$.
      $\mathcal{B}$ randomly chooses $r'_{i,j}$ from $\mathbb{Z}_p^*$ for every $j \in N$, and sets $r_{i,j} = r'_{i,j} \cdot b \bmod p$ for $j \neq k$ and $r_{i,k} = ab + r'_{i,k} \cdot b \bmod p$. After that, $\mathcal{B}$ computes $\mathtt{SPK}_i = D_i = \prod_{j=1}^{n} \frac{1}{(g^b)^{r'_{i,j}}} = g^{-\sum_{j=1}^{n} r'_{i,j} \cdot b} = g^{ab - r_i}$ with implicitly setting $r_i = \sum_{j=1}^{n} r_{i,j} = ab + \sum_{j=1}^{n} r'_{i,j} \cdot b \bmod p$. Note that $r'_{i,j}$ should be recorded by $\mathcal{B}$, since it will be used in $\mathcal{O}_{sdk}$.

∗ If S_i satisfies W^*, then $\mathcal{B}$ chooses a random r'_i from $\mathbb{Z}_p^*$ and computes $\mathtt{SPK}_i = D_i = g^{r'_i}$ with implicitly setting $r_i = ab - r'_i \bmod p$.

– $\mathcal{O}_{sdk}$: There are also two cases in this oracle. Note that $\mathtt{gen}_i$ should follow the restrictions as described in the security model.

∗ If $\mathtt{gen}_i \neq \mathtt{gen}^*$, then $\mathcal{B}$ computes $g^{\beta_i} = A/\mathtt{SPK}_i$, where $\mathtt{SPK}_i$ is from $\mathcal{O}_{spk}$ for the same user. Note that $\mathcal{B}$ has no idea about the value of β_i. $\mathcal{B}$ further chooses $r_{i,j}$ from $\mathbb{Z}_p^*$ for $j \in N$, and they satisfy $\sum_{j=1}^{n} r_{i,j} = 1 \bmod p$.

· For each $j \in N$,
$$D_{i,j} = \begin{cases} g^{\frac{r_{i,j}\cdot\beta_i}{t_{\mathtt{gen}_i,j}}}, & \text{if } j \in S_i; \\ g^{\frac{r_{i,j}\cdot\beta_i}{t_{\mathtt{gen}_i,n+j}}}, & \text{if } j \notin S_i. \end{cases}$$

· For every $j \in N$, $F_{i,j} = g^{\frac{r_{i,j}\cdot\beta_i}{t_{\mathtt{gen}_i,2n+j}}}$.

Note that $t_{\mathtt{gen}_i,j}$'s are from $\mathcal{O}_{update}$ corresponding to generation $\mathtt{gen}_i$.

∗ If $\mathtt{gen}_i = \mathtt{gen}^*$, then S_i cannot satisfy W^*. By using $r'_{i,j}$ generated in $\mathcal{O}_{spk}$ for the same user, $\mathcal{B}$ can compute $D_{i,k} = (g^a)^{1/\theta_{i,k}} \cdot g^{r'_{i,k}/\theta_{\mathtt{gen}^*,k}} = g^{(ab+r'_{i,k}\cdot b)/(b\cdot\theta_{\mathtt{gen}^*,k})} = g^{r_{i,k}/(b\cdot\theta_{\mathtt{gen}^*,k})}$.

For $j \neq k$, we have the followings.

· Case $j \in S_i \wedge j \in I \wedge \tilde{j} = j$, $\mathcal{B}$ computes $D_{i,j} = (g^b)^{r'_{i,j}/\eta_{\mathtt{gen}^*,j}} = g^{r_{i,j}/\eta_{\mathtt{gen}^*,j}}$.

· Case $j \in S_i \wedge ((j \in I \wedge \tilde{j} = \bar{j}) \vee j \notin I)$, $\mathcal{B}$ computes $D_{i,j} = g^{r'_{i,j}/\eta_{\mathtt{gen}^*,j}} = g^{r_{i,j}/(b\cdot\eta_{\mathtt{gen}^*,j})}$.

· Case $j \notin S_i \wedge ((j \in I \wedge \tilde{j} = j) \vee j \notin I)$, $\mathcal{B}$ computes $D_{i,j} = g^{r'_{i,j}/\theta_{\mathtt{gen}^*,j}} = g^{r_{i,j}/(b\cdot\theta_{\mathtt{gen}^*,j})}$.

· Case $j \notin S_i \wedge (j \in I \wedge \tilde{j} = \bar{j})$, $\mathcal{B}$ computes $D_{i,j} = (g^b)^{r'_{i,j}/\theta_{\mathtt{gen}^*,j}} = g^{r_{i,j}/\theta_{\mathtt{gen}^*,j}}$.

$F_{i,j}$ can be computed similarly. In particular, $F_{i,k} = (g^a)^{1/\vartheta_{\mathtt{gen}^*,k}} \cdot g^{r'_{i,k}/\vartheta_{\mathtt{gen}^*,k}} = g^{(ab+r'_{i,k}\cdot b)/(b\cdot\vartheta_{\mathtt{gen}^*,k})} = g^{r_{i,k}/(b\cdot\vartheta_{\mathtt{gen}^*,k})}$.

For $j \neq k$, we have the followings.

· Case $j \in I$, $\mathcal{B}$ computes $F_{i,j} = g^{r'_{i,j}/\vartheta_{\mathtt{gen}^*,j}} = g^{r_{i,j}/(b\cdot\vartheta_{\mathtt{gen}^*,j})}$.

· Case $j \notin I$, $\mathcal{B}$ computes $F_{i,j} = g^{b r'_{i,j}/\vartheta_{\mathtt{gen}^*,j}} = g^{r_{i,j}/\vartheta_{\mathtt{gen}^*,j}}$.

– $\mathcal{O}_{update}$: $\mathcal{B}$ randomly chooses $\mathbf{t}_{\mathtt{gen}+1}$ from $\mathbb{Z}_p^{*3n}$ if $\mathtt{gen} < \mathtt{gen}^* - 1$. Note that we have already had $\mathbf{T}_{\mathtt{gen}^*}$ without knowing $\mathbf{t}_{\mathtt{gen}^*}$.

– $\mathcal{O}_{cipher}$: There are three cases in this oracle.

∗ If $\mathtt{gen}_i < \mathtt{gen}^*$, $\mathcal{B}$ can re-encrypt the ciphertext as that in the real execution, since it knows all $\mathbf{t}$'s.

∗ If $\mathtt{gen}_i = \mathtt{gen}^* \neq 1$, $\mathcal{B}$ first gets $\mathtt{SPK}_i$ and $\mathtt{SDK}_i$ for generation $\mathtt{gen} = 1$, and then uses it to decrypt $\mathtt{CT}_i$ to obtain the corresponding plaintext m. After that, $\mathcal{B}$ encrypts m under A and $\mathbf{T}_{\mathtt{gen}^*}$. At last, $\mathcal{B}$ returns the resulting ciphertext to $\mathcal{A}$.

∗ If $\mathtt{gen}_i = \mathtt{gen}^* = 1$, $\mathcal{B}$ just returns $\mathtt{CT}_i$.

- **Challenge Phase:** $\mathcal{A}$ submits two messages m_0^*, m_1^* of equal length that he wants to be challenged on, $\mathcal{B}$ proceeds as follows.

$\mathcal{B}$ flips a random coin and gets $b \in \{0,1\}$ and sets $C^* = m_b^* \cdot Z$, and computes ciphertext $\mathtt{CT}^*$ as follows,

$$\begin{aligned} \mathtt{CT}^* &= (W^*, C^*, C^{*\prime}, \{C_j^*\}) \\ &= (W^*, C^*, g^c, \{g^{c\cdot\eta_{\mathtt{gen}^*,j}} | j \in I \wedge \tilde{j} = j\}, \\ &\quad \{g^{c\cdot\theta_{\mathtt{gen}^*,j}} | j \in I \wedge \tilde{j} = \bar{j}\}, \\ &\quad \{g^{c\cdot\vartheta_{\mathtt{gen}^*,j}} | j \notin I\}), \end{aligned}$$

Where $j \in N$. If $Z = e(g,g)^{abc}$, the ciphertext is easily seen to form a valid encryption of m_b^*.

- **Phase 2:** Identical to Phase 1.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $Z = e(g,g)^{abc}$. Otherwise, it outputs 0 meaning Z is $e(g,g)^z$.

When the input tuple is sampled from $e(g,g)^{abc}$, then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|\Pr[b = b'] - 1/2| \geq \epsilon$. When the input tuple is sampled from $e(g,g)^z$ then $\Pr[b = b'] = 1/2$. Therefore, with g uniform in $\mathbb{G}$, a, b, c, z uniform in $\mathbb{Z}_p^*$,we have that

$$\begin{aligned} &|\Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] \\ &- \Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^z) = 1]| \\ &\geq |(1/2 \pm \epsilon) - 1/2| = \epsilon. \end{aligned}$$

as required, which completes the proof. ■

Theorem 2: Our proposal is secure against **Type-2** attacks.

Proof: If there is an adversary $\mathcal{A}$ that can successfully launch **Type-2** attacks on our proposed solution, we can also build an algorithm $\mathcal{B}$ breaking the DBDH assumption.

- **Init:** Identical to that in the proof of Theorem 1.
- **Setup:** Identical to that in the proof of Theorem 1.
- **Phase 1:** $\mathcal{A}$ is allowed to query the following oracles adaptively.
  – $\mathcal{O}_{spk}$: Identical to the case of that S_i does not satisfy W^* of $\mathcal{O}_{spk}$ in the proof of Theorem 1.
  – $\mathcal{O}_{sdk}$: There are two cases in this oracle.
    ∗ If S_i satisfies W^*, $\mathcal{B}$ randomly chooses $r_{i,j}$ from $\mathbb{Z}_p^*$ for $j \in N$, and considers $g^{r_{i,j}}$'s as the resulting $\mathtt{SDK}_i$.
    ∗ If S_i does not satisfy W^*, it is identical to the case of that S_i does not satisfy W^* of $\mathcal{O}_{sdk}$ in the proof of Theorem 1.
  – $\mathcal{O}_{update}$: Identical to that in the proof of Theorem 1.
  – $\mathcal{O}_r$: Identical to that in the proof of Theorem 1.
- **Challenge Phase:** Identical to that in the proof of Theorem 1.
- **Phase 2:** Identical to Phase 1.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $Z = e(g,g)^{abc}$. Otherwise, it outputs 0 meaning Z is $e(g,g)^z$.

We have the similar analysis as that in the proof of Theorem 1. ■

VI. Performance Evaluation

The proposed solution in this paper can be considered as a variant of LLS$^+$15 in some sense. Hence, in this section, we would like to do numerical and experimental comparison between these two schemes. To make the comparison relatively fair, we would like to modify LLS$^+$15 from CCA-secure to CPA-secure by removing the parts related to the CCA security since our scheme is CPA-secure. Note that most of the related parts are due to the FO transformation, we can also use this method to make our scheme CCA-secure.

We denote t_e, t_{e_1} and t_p as the time for one exponentiation in $\mathbb{G}$ and $\mathbb{G}_1$, and one pairing, respectively. We also let $|\mathbb{G}|$, $|\mathbb{Z}_p|$ and $|\mathbb{G}_1|$ as the bit length of an element in $\mathbb{G}$, $\mathbb{Z}_p$ and $\mathbb{G}_1$, respectively. $|W|$ denotes the bit length of the **AND** gate, and ℓ denotes the length of security parameter.

For the numerical comparison, we give both communication and computational comparison in Table II and Table III, respectively. From Table II, it is easy to see that our scheme does not need to generate the second level ciphertext as LLS$^+$15 does. From Table III, we can see that our scheme only needs to recover data from the updated ciphertext. Furthermore, if we let $n = 1$ (as in IBE setting in LLS$^+$15) in Table II and Table III, it can be seen that our scheme is better than LLS$^+$15 in both communication and computational cost except the communication cost of the first-level ciphertext and the computational cost of the security device update.

TABLE II: Communication comparison (n is the number of all attributes)

Scheme	LLS$^+$15 [13]	**Ours**
Secret Part	$2\|\mathbb{G}\|$	$\|\mathbb{G}\|$
Security Device	$2\|\mathbb{G}\| + 2\|\mathbb{Z}_p\|$	$2n\|\mathbb{G}\|$
First-Level Ciphertext	$3\|\mathbb{G}\| + 2\ell$	$\|\mathbb{G}_1\| + (n+1)\|\mathbb{G}\| + \|W\|$
Second-Level Ciphertext	$6\|\mathbb{G}\| + 4\ell$	$\perp$
Updated Ciphertext	$3\|\mathbb{G}\| + \|\mathbb{G}_1\| + 4\ell$	$\|\mathbb{G}_1\| + (n+1)\|\mathbb{G}\| + \|W\|$

TABLE III: Computation comparison (n is the number of all attributes)

Scheme	LLS$^+$15 [13]	**Ours**
Secret Part and Security Device Gen.	$4t_e$	$(2n+1)t_e$
First-Level Ciph. Gen.	$3t_e + t_{e_1}$	$t_{e_1} + (n+1)t_e$
Second-Level Ciph. Gen.	$4t_e + t_{e_1}$	$\perp$
Security Device Update	$2t_e$	$(3n-1)t_e$
Ciphertext Update	$5t_e + 6t_p$	$(2n+1)t_e + t_{e_1}$
Data Recovery (From Original Ciph.)	$8t_e + 2t_p$	$\perp$
Data Recovery (From Updated Ciph.)	$t_{e_1} + 2t_p$	$(n+1)t_p$

For the experimental comparison, we also give communication and computational comparison in Fig. 3. Again, to make the comparison relatively fair, we set the attribute number in the system, the access structure and the private key to 1. (Note that the number of user's attributes are relatively small in real world, eg. 5 or 10. Furthermore, in order to make our scheme more efficient, we could use the attribute based encryption with outsourced decryption method [39] to outsource the large amount of computation to the cloud.)

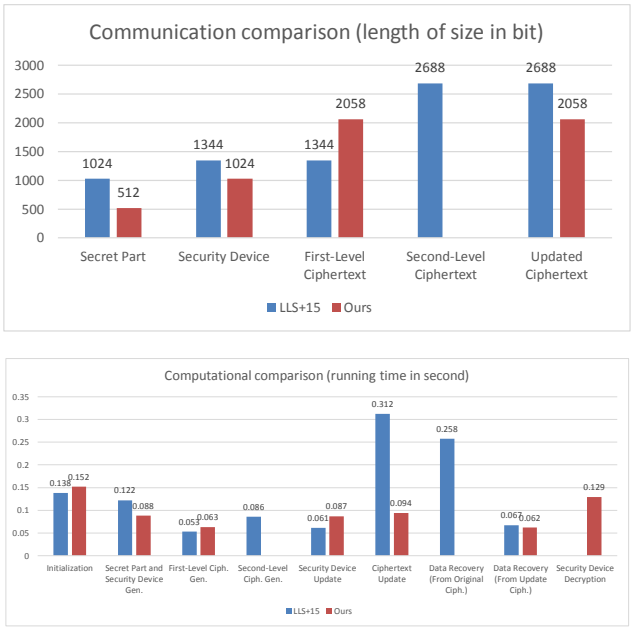


Fig. 3: Communication and computational comparison

In LLS$^+$15, the authors did not explicitly show how the partial decryption process in the security device is executed. Hence, when we did the experiments of the computational comparison, algorithm DataReveal is not separated into two parts. On the contrary, all the comparison experiments are performed in a test bed of one PC. This machine plays the roles of CA, Cloud, DO and DR. The hardware and software of this machine are as follows: Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz 2.20GHz RAM 4.00GB, Java Programming Language, and 64bit win7 operation system; the pairing is working on the type A pairings from [40], [41]. In our experiments, to achieve the corresponding security level, we set $|\mathbb{Z}_p|$ = 160 bits, $|\mathbb{G}|$ = 512 bits and $|\mathbb{G}_1|$ = 1024 bits. We further set ℓ = 160 bits and $|W|$ = 10 bits. Our experimental comparison shows the similar result as the numerical one does. From the communication comparison in Fig. 3, we can see that our scheme is better than LLS$^+$15 except the communication cost of the first-level ciphertext. From computational comparison in Fig. 3, it can be seen that our scheme does not need to generate or recover the second-level ciphertext, which can reduce much computation time. Moreover, in the ciphertext update phase, we can see that our scheme is much more efficient than LLS$^+$15.

Furthermore, to show the efficiency of our proposal, we also did the experiments on a mobile phone as a security device. The result can also be seen in Fig. 3 (the last column of the second figure). The hardware and software of this mobile phone are as follows: Samsung S7 Edge 128G, Chipset: Qualcomm MSM8996 Snapdragon 820, CPU: Quad-core (2x2.15 GHz Kryo & 2x1.6 GHz Kryo), GPU: Adreno 530, OS: Android 7.0; the paring is working on the same curve as the former one. From the result in Fig. 3, we can see that our scheme is

quite practical.

In summary, we can see that our scheme achieves fine-grained two-factor data protection with much less computational, and it is more practical in cloud storage.

VII. Conclusion

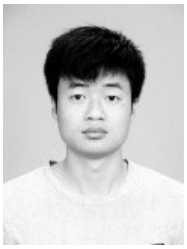
In this paper, we proposed a fine-grained two-factor data protection for cloud storage. The two-factor is realized by separating the secret key into two parts, one can be stored in a potential-insecure place, and the other is stored in a tamper resistant device. Only if one of them is kept secret, the proposal remains secure. Furthermore, with the help of CP-ABE and PRE, we obtained the fine-grained access control on encrypted data and the revocability of tamper resistant device, respectively.

Acknowledgment

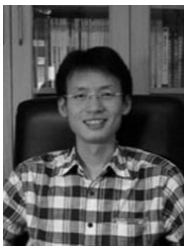
The authors thank the anonymous reviewers for the valuable comments. This work was supported by the National Natural Science Foundation of China [grant numbers 61472364, 61472365, 61379121], the Natural Science Foundation of Zhejiang Province [grant number LR13F020003], the Key Science Research Development Plan of Zhejiang Province [grant number 2017C01091].

References

[1] "Dropbox," www.dropbox.com.
[2] "Google drive," https://www.google.com/drive/.
[3] "Pcloud," www.pcloud.com/.
[4] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *Computers, IEEE Transactions on*, vol. 62, no. 2, pp. 362–375, 2013.
[5] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *Services Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 220–232, 2012.
[6] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *Services Computing, IEEE Transactions on*, vol. 6, no. 2, pp. 227–238, 2013.
[7] H. C. Chen, Y. Hu, P. P. Lee, and Y. Tang, "Nccloud: a network-coding-based storage system in a cloud-of-clouds," *Computers, IEEE Transactions on*, vol. 63, no. 1, pp. 31–44, 2014.
[8] "Encryption key virus threat," http://searchsecurity.techtarget.com/tip/Encryption-key-virus-threat.
[9] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Advances in Cryptology–EUROCRYPT 2002*. Springer, 2002, pp. 65–82.
[10] ——, "Strong key-insulated signature schemes," in *Public Key Cryptography–PKC 2003*. Springer, 2002, pp. 130–144.
[11] G. Hanaoka, Y. Hanaoka, and H. Imai, "Parallel key-insulated public key encryption," in *Public Key Cryptography-PKC 2006*. Springer, 2006, pp. 105–122.
[12] B. Libert, J.-J. Quisquater, and M. Yung, "Parallel key-insulated public key encryption without random oracles," in *Public Key Cryptography–PKC 2007*. Springer, 2007, pp. 298–314.
[13] J. Liu, K. Liang, W. Susilo, J. Liu, and Y. Xiang, "Two-factor data security protection mechanism for cloud storage system," *Computers, IEEE Transactions on*, vol. 65, no. 6, pp. 1992–2004, 2016.
[14] V. J. Winkler, "Cloud computing: Privacy, confidentiality and the cloud," https://technet.microsoft.com/en-us/magazine/dn235775.aspx.
[15] J. K. Liu and D. S. Wong, "Solutions to key exposure problem in ring signature." *IJ Network Security*, vol. 6, no. 2, pp. 170–180, 2008.
[16] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 457–473.
[17] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 99–112.
[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.
[19] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 195–203.
[20] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.
[21] L. Cheung and C. Newport, "Provably secure ciphertext policy abe," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 456–465.
[22] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography–PKC 2011*. Springer, 2011, pp. 53–70.
[23] J. Herranz, F. Laguillaumie, and C. Ràfols, "Constant size ciphertexts in threshold attribute-based encryption," in *Public Key Cryptography–PKC 2010*. Springer, 2010, pp. 19–34.
[24] N. Attrapadung, B. Libert, and E. De Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Public Key Cryptography–PKC 2011*. Springer, 2011, pp. 90–108.
[25] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 463–474.
[26] J. K. Liu, M. H. Au, X. Huang, R. Lu, and J. Li, "Fine-grained two-factor access control for web-based cloud computing services," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 3, pp. 484–497, 2016.
[27] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 417–426.
[28] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Pairing-Based Cryptography–Pairing 2009*. Springer, 2009, pp. 248–265.
[29] ——, "Attribute-based encryption supporting direct/indirect revocation modes," in *Cryptography and Coding*. Springer, 2009, pp. 278–300.
[30] J.-l. Qian and X.-l. Dong, "Fully secure revocable attribute-based encryption," *Journal of Shanghai Jiaotong University (Science)*, vol. 16, pp. 490–496, 2011.
[31] F. Zhang, Q. Li, and H. Xiong, "Efficient revocable key-policy attribute based encryption with full security," in *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*. IEEE, 2012, pp. 477–481.
[32] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in CryptologyłEUROCRYPT'98*. Springer, 1998, pp. 127–144.
[33] A.-A. Ivan and Y. Dodis, "Proxy cryptography revisited." in *NDSS*, 2003.
[34] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.
[35] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 185–194.
[36] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Public Key Cryptography–PKC 2008*. Springer, 2008, pp. 360–379.
[37] J. Shao and Z. Cao, "Cca-secure proxy re-encryption without pairings," in *Public Key Cryptography–PKC 2009*. Springer, 2009, pp. 357–376.
[38] L. Xu, X. Wu, and X. Zhang, "Cl-pre: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 87–88.
[39] C. Zuo, J. Shao, G. Wei, M. Xie, and M. Ji, "Chosen ciphertext secure attribute-based encryption with outsourced decryption," in *Australasian Conference on Information Security and Privacy*. Springer, 2016, pp. 495–508.
[40] B. Lynn, "The pairing-based cryptography library," http://crypto.stanford.edu/pbc/.
[41] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece, June 28 - July 1: IEEE, 2011, pp. 850–855, http://gas.dia.unisa.it/projects/jpbc/.



Cong Zuo received his bachelor degree from the School of Computer Engineering at Nanjing Institute of Technology, and his master degree from the School of Computer Science and Information Engineering at Zhejiang Gongshang University, China. He is currently a PhD student at Monash University under the supervision of Dr Joseph K. Liu. His main research interest is the applied cryptography.



Jun Shao received the Ph.D. degree from the Department of Computer Science and Engineering at Shanghai Jiao Tong University, Shanghai, China in 2008. He was a postdoc in the School of Information Sciences and Technology at Pennsylvania State University, USA from 2008 to 2010. He is currently a professor of the School of Computer Science and Information Engineering at Zhejiang Gongshang University, Hangzhou, China. His research interests include network security and applied cryptography.

Joseph K. Liu received the Ph.D. degree in information engineering from the Chinese University of Hong Kong in July 2004, specializing in cyber security, protocols for securing wireless networks, privacy, authentication, and provable security. He is now a senior lecturer at Monash University, Australia. His current technical focus is particularly cyber security in Cloud computing paradigm, smart city, lightweight security, and privacy enhanced technology. He has published more than 80 referred journal and conference papers and received the Best Paper Award from ESORICS 2014. He has served as the program chair of ProvSec 2007, 2014, and as the program committee of more than 35 international conferences.

Guiyi Wei is a professor of the School of Computer Science and Information Engineering at Zhejiang Gongshang University. He obtained his Ph.D. in Dec 2006 from Zhejiang University, where he was advised by Cheung Kong chair professor Yao Zheng. His research interests include wireless networks, mobile computing, cloud computing, social networks and network security.

Yun Ling Yun Ling received his B.Sc. and M.Sc. in Computer Science from Zhejiang University. He is a professor of the School of Computer Science and Information Engineering at Zhejiang Gongshang University. His main research interests include intelligent information processing, networks and distributed computing.