Time-Based Proxy Re-encryption Scheme for Secure Data Sharing in a Cloud Environment

Qin Liu[a,b], Guojun Wang[a,*], Jie Wu[b]

[a] *School of Information Science and Engineering*
Central South University
Changsha, Hunan Province, P. R. China, 410083
[b] *Department of Computer and Information Sciences*
Temple University
Philadelphia, PA 19122, USA

Abstract

A fundamental approach for secure data sharing in a cloud environment is to let the data owner encrypt data before outsouring. To simultaneously achieve *fine-grained access control on encrypted data* and *scalable user revocation*, existing work combines attribute-based encryption (ABE) and proxy re-encryption (PRE) to delegate the cloud service provider (CSP) to execute re-encryption. However, the data owner should be online in order to send the PRE keys to the CSP in a timely fashion, to prevent the revoked user from accessing the future data. The delay of issuing the PRE keys may cause potential security risks. In this paper, we propose a time-based proxy re-encryption (TimePRE) scheme to allow a user's access right to expire automatically after a predetermined period of time. In this case, the data owner can be offline in the process of user revocations. The basic idea is to incorporate the concept of time into the combination of ABE and PRE. Specifically, each data is associated with an *attribute-based access structure* and an *access time*, and each user is identified by a set of *attributes* and a set of *eligible time periods* which denote the period of validity of the user's access right. Then, the data owner and the CSP are required to share a *root secret key* in advance, with which CSP can automatically update the access time of the data with the time that it receives a data access request. Therefore,

*Corresponding Author:
Email address: csgjwang@mail.csu.edu.cn (Guojun Wang)
URL: http://trust.csu.edu.cn/faculty/~csgjwang (Guojun Wang)

given the re-encrypted ciphertext, only the users whose attributes satisfy the access structure and whose access rights are effective in the access time can recover corresponding data.

Keywords: Cloud computing; time; proxy re-encryption; attribute-based encryption

1. Introduction

Cloud computing has increasingly become a commercial trend due to its desirable properties, such as scalability, elasticity, fault-tolerance, and pay-per-use [1]. Small and medium-sized organizations, in particular, can achieve great flexibility at a low price by outsourcing their data and query services to the cloud. The cloud infrastructures are more powerful and reliable than personal computing devices, but they are still susceptible to internal threats (e.g., via virtual machines) and external threats (e.g., via system vulnerabilities) that may leak user sensitive data [7, 25]. Therefore, many organizations still hesitate to adopt cloud services [24].

To prevent unsolicited disclosure of sensitive information, *data owners* may have to encrypt their data before outsourcing [6, 15, 18]. In this way, only the authorized *users* with the decryption keys can recover the data, and other unsolicited accessors without the decryption keys, e.g., the cloud service provider (CSP), cannot execute decryption, even if they successfully obtain the ciphertexts stored in the cloud. However, new problems, such as *fine-grained access control on the encrypted data* and *scalable user revocation*, emerge for this solution[1].

To illustrate, let us consider the following application scenario, as shown in Fig. 1. Suppose that University A outsources the electronic library database to a cloud for easy access by its staff and students. For the protection of copyright, each piece of data is encrypted before outsourcing. In this application, the staff and students are users, and University A is the data owner who will specify the access structure for each data, and will distribute decryption keys to users. Once joining University A, each user will first be assigned an access right with certain validity for accessing the outsourced database. Once the

[1]There are certainly more than two problems existing in such an environment, e.g., impersonation of user identity and compromising data integrity. We do not solve all the potential security threats, but only address those directly related to our work.

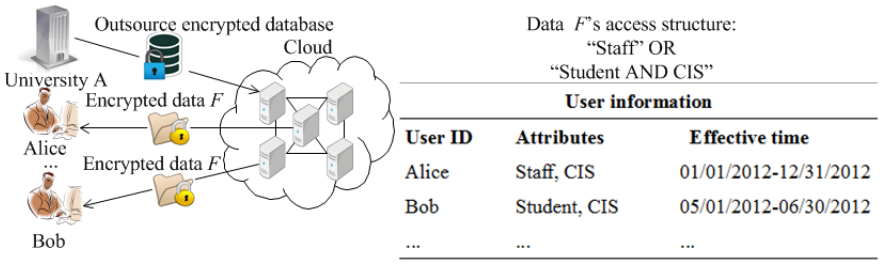

Figure 1: Company A outsources an encrypted database to the cloud.

period of validity passes, this user should request an extension for his access right from University A.

In Fig. 1, data F's access structure stipulates that only the staff or the students in computer information science (CIS) department have the right to access it. In this access structure, the data owner does not know the exact identities of the authorized users, but rather he only has a way to describe them using certain descriptive attributes, such as `Staff`, `Student`, and `CIS`. Therefore, the adopted encryption system should have the ability to efficiently implement a fine-grained access control over attributes.

Ciphertex-policy attribute-based encryption (CP-ABE) [3, 19] as a promising branch of ABE [23] has such a property. In CP-ABE, users are identified by a set of *attributes* rather than an exact identity. For each eligible attribute, the user will be issued a *user attribute secret key* (UAK). Each data is encrypted with an *attribute-based access structure*, such that only the users whose attributes satisfy the access structure can decrypt the ciphertext using their UAKs. For example, for data which is encrypted with the access structure $\{(\texttt{Student} \wedge \texttt{CIS}) \vee \texttt{Staff}\}$, either users with attributes `Student` and `CIS`, or users with attribute `Staff`, can recover data.

Furthermore, from the above application scenario, we observe that each user's access right is only effective in a predetermined period of time. For example, the effective time of Alice's access right is from 01/01/2012 to 12/31/2012 and she can access the database in year 2012, but the effective time of Bob's access right is from 05/01/2012 to 06/30/2012 and thus he cannot access the database after June.

A naïve way is to let University A expose the effective time of each user's access right to the CSP, with which the CSP can execute user revocation correctly. The main drawback of this approach is that the CSP will know the effective time of each user's access right, which may cause potential leakage of

sensitive information. For example, the CSP may guess the user whose access right has longer effective time is in a more important position than that with shorter-term access right. Due to the same reason, the ticket-based access control [21], which requires the users to expose their tickets to the CSP, may also expose the effective time of each ticket.

Another approach is to require the data owner to personally execute user revocation. A revoked user still retains the keys issued earlier, and he can recover data if he obtains corresponding ciphertexts. To prevent the revoked user from accessing further data, the data owner needs to immediately re-encrypt the ciphertexts. Furthermore, the data owner needs to distribute new keys to the remaining authorized users for them to access the database. When there are frequent user revocations, this solution will result in a heavy workload on the data owners.

A better solution should take full advantage of abundant resources in a cloud by delegating the CSP to execute computationally intensive tasks in user revocations, while *leaking the least information* to the CSP. Existing work [29, 28] proposed the idea of applying the combination of proxy re-encryption (PRE) [4, 13] and ABE to a cloud environment. This approach requires that once a user is revoked from a system, the data owner should send the *PRE keys* to the CSP, with which the CSP can be delegated to re-encrypt corresponding ciphertexts. The original schemes in [29, 28] also allow the CSP to be delegated to distribute the *update keys* to remaining authorized users for them to generate new UAKs. However, the CSP should first know the identities of these authorized users, and it will finally know the effective time of each user. Therefore, to avoid leaking additional information, the data owner should distribute the update keys on his own. Due to the security of ABE and PRE, the CSP cannot know neither underlying data nor UAKs. The main problem of this approach is that the data owner should be online in order to send the PRE keys to the CSP in a timely fashion to prevent the revoked users from accessing future data. The delay of issuing PRE keys will cause potential security risks.

In this paper, we propose a time-based proxy re-encryption (TimePRE) scheme, by incorporating the concept of time into a combination of CP-ABE and PRE. Our scheme allows a user's access right to automatically expire after a predetermined period of time. Therefore, our scheme can avoid the security risks caused by the delay of issuing PRE keys. The data owner, who can be offline in the process of user revocations, has much less workload.

Specifically, we associate each data with an *attribute-based access struc-*

ture and an *access time*. We identify each user by a set of *attributes* and a set of *eligible time periods* which denotes how long the user is eligible for these attributes, i.e., the period of validity of his access right. Then, we construct a *time tree* for the actual time, and require the data owner and the CSP to share a *root secret key* in advance, with which CSP can automatically calculate appropriate *PRE keys*, and use these PRE keys to re-encrypt the ciphertext, i.e., update the access time of the data with the time that it receives a data access request. Given the re-encrypted ciphertext, only the users whose attributes satisfy the access structure and whose access rights are effective in the access time can recover corresponding data.

The security of the TimePRE scheme can be derived from CP-ABE and PRE. CP-ABE ensures that ciphertexts are semantically secure [5], and PRE ensures that the CSP re-encrypts ciphertexts without knowing the underlying data and UAKs. Note that our security model does not assume that there is no trust relationship between the CSP and the data owner. Actually, as in existing work [29, 28], we assume that the CSP will correctly carry out our scheme to ensure system-wide security. Our scheme can be considered as a first step for secure data sharing in a cloud environment.

Our contributions are threefold:

1. **It is secure.** The TimePRE scheme enables the CSP to automatically re-encrypt data without receiving any PRE keys from the data owner. Our scheme can avoid potential security risks that are raised by the delay of issuing the PRE keys.
2. **It is practical.** We extend a CP-ABE system by applying the PRE technique to reduce the workload on the data owner. Our scheme is more applicable to the environment where there are frequent user revocations and the data owners cannot be often online.
3. **It is efficient.** We incorporate the concept of time into an efficient CP-ABE system [28, 26], so that a user can rapidly recover data by executing a constant number of bilinear maps [5].

This paper is structured as follows: First, we review related work in Section 2. Then, we provide models and assumptions in Section 3, and introduce technical preliminaries in Section 4. We outline the TimePRE scheme in Section 5 and provide a construction in Section 6. Then, we analyze the correctness, performance, and security of the TimePRE scheme in Section 7, and provide an additional discussion in Section 8. Finally, we conclude this paper and provide our future work in Section 9.

2. Related Work

In the cloud computing model, the data owners have to entrust sensitive data to a remote cloud, which is maintained by an external party, i.e., the cloud service provider (CSP). Rather than fully trusting the CSP, existing research [6, 15, 18] proposed to only outsource encrypted data to the cloud. Our work is on fine-grained access control on the encrypted data and scalable user revocation. We introduce the related work as follows:

2.1. Fine-Grained Access Control on Encrypted Data

To protect data security from an untrusted server, the work in reference [14] adopts traditional symmetric key cryptographic system to encrypt data. Before outsourcing, the data owner will first classify data with similar access control lists (ACLs) into a file-group, and then encrypt each file-group with a symmetric key. The symmetric key will be distributed to the users in the ACL, so that only the users in the ACL can access this group of files. The main drawback of this approach is that the key size managed by the data owner grows linearly with the number of file-groups.

Another approach is proposed by [11], which is based on the combination of traditional symmetric key and public key cryptographic systems. The data owner first specifies an ACL for a data, and then encrypts the data with a symmetric key, which is encrypted with the public keys of users in the ACL. Therefore, only the users in the ACL can use their secret keys to recover the symmetric key, and then use the symmetric key to recover the data. The main drawback of this approach is that the costs for encrypting a data will grow linearly with the number of users in the ACL.

Therefore, an ideal approach is to encrypt each data once, and distribute appropriate keys to users once, so that each user can only decrypt his authorized data. Attribute-based encryption (ABE) [23], which has developed to two branches, key-policy ABE (KP-ABE) [12] and ciphertext-policy ABE (CP-ABE) [3, 19], is a promising cryptographic technique having such a property. This flexibility makes ABE an attractive choice when selecting an encryption scheme for cloud computing.

To guarantee security and privacy of medical data stored in the cloud, the work in reference [20] constructed a secure and privacy-preserving electronic health record (EHR) system based on ABE. Facing new challenges in cloud data centers, the work in reference [16] proposed an accountable CP-ABE scheme, which allows tracing the identities of a misbehaving users who

leaked their secret keys to others. In our previous work, an efficient CP-ABE system, termed hierarchical attribute-based encryption (HABE) [26], is proposed for secure data sharing in cloud environment. By uniquely combining hierarchical identity-based encryption (HIBE) [10] and CP-ABE, HABE requires to execute only a constant number of bilinear maps to recover data. As an improvement, we proposed a conjunctive fuzzy and precise identity-based encryption (FPIBE) scheme [27] which supports both identity-based and attribute-based access structures for data stored in the cloud.

2.2. User Revocation

User revocation is a well studied, but non-trivial task. The key problem is that the revoked users still retain the keys issued earlier, and thus can still decrypt ciphertexts. Therefore, whenever a user is revoked, the re-keying and re-encryption operations need to be executed by the data owner to prevent the revoked user from accessing the future data. For example, when ABE is adopted to encrypt data, the work in reference [22] proposed to require the data owner to periodically re-encrypt the data, and re-distribute new keys to authorized users. This approach is very inefficient due to the heavy workload introduced on the data owner.

A better solution is to let the data owner delegate a third party to execute some computational intensive tasks, e.g, re-encryption, while leaking the least information. Proxy re-encryption [4, 13] is a good choice, where a semi-trusted proxy is able to convert a ciphertext that can be decrypted by Alice into another ciphertext that can be decrypted by Bob, without knowing the underlying data and user secret keys. For example, the work in reference [29] is the first to combine KP-ABE and PRE to delegate most of the computation tasks involved in user revocation to the CSP. Our previous work [28] is the first to combine PRE and a CP-ABE system (HABE) to achieve a scalable revocation mechanism in cloud computing. The work in reference [30] that supports attribute revocation may be applicable to a cloud environment. This approach requires that once a user is revoked from a system, the data owner should send PRE keys to the CSP, with which the CSP can be delegated to execute re-encryption. The main problem of this approach is that the data owner should be online in order to send the PRE keys to the CSP in a timely fashion, to prevent the revoked user from accessing the data. The delay of issuing PRE keys may cause potential security risks.

In this paper, we extend an efficient CP-ABE system (HABE [28, 26]) by incorporating the concept of time to perform automatic proxy re-encryption. The main difference from prior work is that we enable each user's access right to be effective in a pre-determined time, and enable the CSP to re-encrypt ciphertexts automatically based on its own time. Thus, the data owner can be offline in the process of user revocations.

3. Models, Assumptions, and Design Goals

In this section, we will describe the system model and security model in our scheme, and provide our design goals and related assumptions.

3.1. System Model

We consider a cloud computing environment consisting of a cloud service provider (CSP), a data owner, and many users. The CSP maintains cloud infrastructures, which pool the bandwidth, storage space, and CPU power of many cloud servers to provide 24/7 services. We assume that the cloud infrastructures are more reliable and powerful than personal computers. In our system, the CSP mainly provides two services: data storage and re-encryption. After obtaining the encrypted data from the data owner, the CSP will store the data on several cloud servers, which can be chosen by the consistent hash function [8], where the input of the consistent hash function is the key of the data, and the outputs of the consistent hash function are the IDs of the servers that store the data. On receiving a data access request from a user, the CSP will re-encrypt the ciphertext based on its own time, and return the re-encrypted ciphertext.

The data owner outsources a set of data to the cloud. Each piece of data is encrypted before outsourcing. The data owner is responsible for determining the access structure for each data, and distributing user attribute secret keys (UAKs) corresponding to user attributes to each user. When a user wishes to access data, he will first request appropriate keys from the data owner, and then request the CSP to download the ciphertext. If his access right is effective when he requests the data, he can successfully execute decryption.

Since the TimePRE scheme is based on time, a slight time difference may impact the correctness of our scheme. The network time protocol (NTP), which can be used to achieve time synchronization in a cloud environment, still incurs time drifts of several seconds. Our work focuses on the cryptographic design and construction. Therefore, in our system model, we simply

assume that there is a *global time* to ensure time consistency among all entities. Actually, a global time is hard to achieve in a cloud environment. We may use the techniques proposed in our previous work [17] to ensure the TimePRE scheme works well in a no-global-time cloud environment. We also indicate that our scheme is more suitable for the cloud applications where a coarse-grained time accuracy is satisfactory, e.g., a day or an hour. For the applications that require a fine-grained time accuracy, e.g., a second or microsecond, the cost to ensure correctness will be excessive, even if we apply the techniques in [17] to the TimePRE scheme.

3.2. Security Model

There are two kinds of adversaries in the system: honest but curious CSP, and malicious users. The honest but curious CSP will correctly execute the protocol defined previously, but may try to gain some additional information about the stored data. The malicious user wants to access the data, to which he is not eligible to access. The communication channels are assumed to be secured under existing security protocols such as SSL to protect data security during information transferring.

Note that both an honest but curious CSP, and malicious users, can exist together. We assume that the CSP will not collude with any malicious user. However, malicious users may collude to obtain additional information. This assumption is reasonable, and has also been made known in previous research, e.g., the proxy re-encryption system [4], where the semi-trusted proxy server is assumed to not collude with other entities to ensure system-wide security. Note that our security model does not assume that there is no trust relationship between the CSP and the data owner. As in existing work [29, 28], we assume that the CSP will correctly carry out our scheme to ensure data security.

3.3. Design Goals

The main design goal is to achieve fine-grained access control and scalable user revocation while protecting data security in cloud computing. Specifically, we categorize our goals into the following points:

- *Scalability.* The data owner can be offline in the process of user revocations.
- *Fine-grained access control.* The data owner can specify expressive access structure for each data.

- *Data confidentiality.* The CSP and malicious users cannot recover data without the data owner's permission.
- *Cost efficiency.* The re-encryption cost on the CSP is relatively low.

For scalability, we should enable each user's access right to expire automatically after a predetermined period of time; for fine-grained access control, we should adopt an encryption system that supports attribute-based access structure, such as CP-ABE and KP-ABE; for data confidentiality, we should allow only the users whose attributes satisfy the access structure and whose access rights are effective in the access time to recover the data; for cost efficiency, we should apply lazy re-encryption (LRE) [11] to the TimePRE scheme, so that the CSP can re-encrypt the data only when receiving data access requests from users.

4. Technique Preliminaries

In this section, we will first introduce some basic definitions, and then we will provide an overview of one proxy re-encryption (PRE) scheme and hierarchical attribute-based encryption (HABE).

4.1. Definitions

The related definitions and complexity assumptions closely follow those in Boneh et al [5].

Definition 1 (Bilinear Map): Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order q, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ is a multiplicative group. A bilinear map, $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, satisfies the following properties:

1. Computable: There is a polynomial time algorithm to compute $\hat{e}(P, Q) \in \mathbb{G}_2$, for any $P, Q \in \mathbb{G}_1$.
2. Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}_q^*$.
3. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$.

Definition 2 (BDH Parameter Generator): A randomized algorithm $\mathcal{IG}$ is called a BDH parameter generator if $\mathcal{IG}$ takes a sufficiently large security parameter $K > 0$ as input, runs in polynomial time in K, and outputs a prime number q, the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and the description of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

Definition 3 (BDH Problem): Given a random element $P \in \mathbb{G}_1$, as well as aP, bP, and cP, for some $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P,P)^{abc} \in \mathbb{G}_2$.

Definition 4 (BDH Assumption): If $\mathcal{IG}$ is a BDH parameter generator, 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 $\mathcal{B}$ outputs $\hat{e}(P,P)^{abc}$ on inputs q, $\mathbb{G}_1$, $\mathbb{G}_2$, $\hat{e}$, P, aP, bP, cP, where $< q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} >$ are the outputs of $\mathcal{IG}$ for a sufficiently large security parameter K, P is a random element $\in \mathbb{G}_1$, and a, b, c are random elements of $\mathbb{Z}_q^*$. The BDH assumption is that $Adv_{\mathcal{IG}}(\mathcal{B})$ is negligible for any efficient algorithm $\mathcal{B}$.

4.2. Proxy Re-Encryption (PRE)

Let us illustrate the motivation of the PRE scheme [4] by the following example: Alice receives emails encrypted under her public key PK_A via a semi-trusted mail server. When she leaves for vacation, she wants to delegate her email to Bob whose public key is PK_B, but does not want to share her secret key SK_A with him. The PRE scheme allows Alice to provide a PRE key $RK_{A\to B}$ to the mail server, with which the mail server can convert a ciphertext that is encrypted under Alice's public key PK_A into another ciphertext that can be decrypted by Bob's secret key SK_B, without seeing the underlying plaintext, SK_A, and SK_B.

Let $\mathbb{G}$ be a multiplicative group of prime order q, and g be a random generator of $\mathbb{G}$. The PRE scheme is consisted of the following algorithms:

Key Generation: Alice can choose a random element $a \in \mathbb{Z}_q^*$ as her secret key SK_A, and her public key PK_A is $g^a \in \mathbb{G}$. In the same way, Bob's public/secret key pair (SK_B, PK_B) are (b, g^b). The PRE key $RK_{A\to B} = b/a(\mod q)$ is used to transfer a ciphertext that is encrypted under PK_A to the ciphertext that can be decrypted with SK_B, and vise versa.

Encryption: To encrypt a message $m \in \mathbb{G}$ to Alice, the sender randomly chooses $r \in \mathbb{Z}_q^*$, and generates ciphertext $C_A = (C_{A1}, C_{A2}) = (g^r m, g^{ar})$.

Decryption: Given the ciphertext $C_A = (C_{A1}, C_{A2})$, Alice can recover message m with her secret key a by calculating $C_{A1}/(C_{A2})^{1/a}$.

Re-encryption: Given $RK_{A\to B}$, the mail server can convert C_A to C_B that can be decrypted by Bob as follows: $C_{B1} = C_{A1}$ and $C_{B2} = (C_{A2})^{RK_{A\to B}}$. Given the ciphertext (C_{B1}, C_{B2}), Bob can recover message m with his secret key b by calculating $C_{B1}/(C_{B2})^{1/b}$.

Note that although the data is encrypted twice, first encrypted with Alice's public key, and then re-encrypted with a PRE key, Bob only needs to

execute decryption once to recover data. The PRE scheme is based on ElGamal encryption [9], and thus the ciphertext is semantically secure, and given the PRE key, the mail server cannot guess the secret keys a nor b. Please refer to [4] for more details.

4.3. Hierarchical Attribute-Based Encryption (HABE)

Our TimePRE scheme is extended from an efficient CP-ABE system, HABE [28, 26], which is constructed based on the bilinear map [5]. The access structure in HABE is expressed as disjunctive normal form (DNF). The original HABE allows a delegation mechanism in the generation of keys, as that in hierarchical identity-based encryption (HIBE) [10]. Since our TimePRE scheme focuses on automatic re-encryption, we provide a modified version of HABE as follows:

Setup. This algorithm takes a security parameter K and the universal attribute $\mathbb{UA}$ as inputs, and outputs system public key PK and system master key MK as follows:

$$PK = (\{PK_a\}_{a\in\mathbb{UA}}, q, \mathbb{G}_1, \mathbb{G}_2, Q_0, \hat{e}, P_0, P_1)$$
$$MK = (\{sk_a\}_{a\in\mathbb{UA}}, mk_0, mk_1, SK_1)$$

where $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e})$ are the outputs of a BDH parameter generator $\mathcal{IG}$, P_0 is a random generator of $\mathbb{G}_1$, and P_1 is a random element in $\mathbb{G}_1$; $sk_a \in \mathbb{Z}_q^*$ is the secret key of attribute a and $PK_a = sk_aP_0 \in \mathbb{G}_1$ is the public key of attribute a; mk_0 and mk_1 are random elements in $\mathbb{Z}_q^*$, $Q_0 = mk_0P_0 \in \mathbb{G}_1$, and $SK_1 = mk_0P_1 \in \mathbb{G}_1$.

Key Generation. This algorithm takes system public key PK, system master key MK, user public key PK_u and attribute a as inputs, and generates user identity secret key SK_u and user attribute secret key $SK_{u,a}$, as follows:

$$SK_u = mk_1mk_uP_0 \in \mathbb{G}_1$$
$$SK_{u,a} = SK_1 + mk_1mk_uPK_a \in \mathbb{G}_1$$

where $mk_u = H_1(PK_u) \in \mathbb{Z}_q^*$ and H_1: $\mathbb{G}_1 \to \mathbb{Z}_q^*$ is a hash function which can be modeled as random oracle.

Encryption. This algorithm takes system public key PK, a DNF access structure $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i)$, and data $F \in \mathbb{G}_2$ as inputs to generate ciphertext

$C_{\mathbb{A}} = (\mathbb{A}, U_0, U_1, \ldots, U_N, V)$ as follows:

$$\begin{aligned}
&\mathbb{A} = \bigvee_{i=1}^{N}(CC_i) = \bigvee_{i=1}^{N}(\bigwedge_{j=1}^{n_i} a_{ij}),\\
&U_0 = rP_0,\\
&\{U_i = r\sum_{a\in CC_i} PK_a\}_{1\le i\le N},\\
&V = F\cdot\hat{e}(Q_0, rn_{\mathbb{A}}P_1)
\end{aligned}$$

where $N \in \mathbb{Z}^+$ is the number of conjunctive clauses in $\mathbb{A}$, $n_i \in \mathbb{Z}^+$ is the number of attributes in the *i-th* conjunctive clause CC_i, and a_{ij} is the *j-th* attribute in CC_i; r is a random element in $\mathbb{Z}_q^*$, and $n_{\mathbb{A}}$ is the lowest common multiple (LCM) of n_1,..., n_N.

Decryption. This algorithm takes system public key PK, user identity secret key, and user attribute secret keys on all attributes in the *i-th* conjunctive clause CC_i as inputs to recover data F as follows:

$$F = V/\left(\frac{\hat{e}(U_0, \frac{n_{\mathbb{A}}}{n_i}\sum_{a\in CC_i} SK_{u,a})}{\hat{e}(SK_u, \frac{n_{\mathbb{A}}}{n_i}U_i)}\right)$$

To encrypt a data, we need to execute one bilinear map and $O(N)$ number of point multiplication operations to output a ciphertext of $O(N)$ length, where N is the number of conjunctive clauses in the access structure; To recover a data, we only need to execute $O(1)$ bilinear map operations. We prove HABE to be semantically secure under the random oracle model and the BDH assumption. More details can be found in [28, 26].

5. Outline of the TimePRE Scheme

In this section, we will first illustrate the main idea of the TimePRE scheme, and then we will provide the formal definition of our scheme.

5.1. Main Idea

The main idea of the TimePRE scheme is to incorporate the concept of time into the combination of HABE and PRE. Intuitively, each user is identified by a set of *attributes* and a set of *effective time periods* that denotes how long the user is eligible for these attributes, i.e., the period of validity of the user's access right. The data accessed by the users is associated with an *attribute-based access structure* and *an access time*. The access structure is

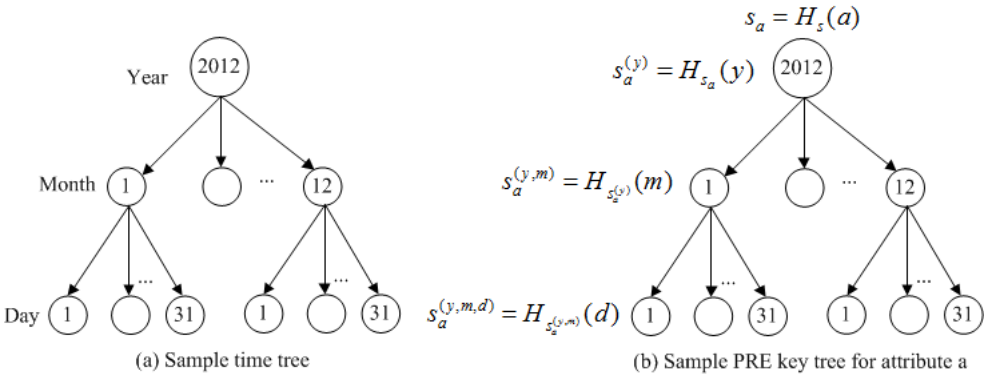

Figure 2: Three-level time tree and attribute a's PRE key tree.

specified by the data owner, but the access time is updated by the CSP with the time of receiving an access request. The data can be recovered by only the users whose attributes satisfies the access structure and whose access rights are effective in the access time.

To enable the CSP to update the access time automatically, we first express *actual time* as a *time tree*. The height of the time tree can be changed as required. For ease of presentation, in this paper we only consider a three-layer time tree as shown in Fig. 2-(a), where time is accurate to the day, and the time tree is classified into three layers in order: year, month, and day. We use (y, m, d), (y, m), and (y) to denote a particular day, month, and year, respectively. For example, $(2012, 4, 5)$ denotes April 5, 2012. The access time associated with a data corresponds to a leaf node in the time tree, and the effective time periods associated with a user correspond to a set of nodes in the time tree. If there is a node corresponding to an effective time period that is an ancestor of (or the same as) the node corresponding to the access time, then the user's access right is effective in the access time.

Then, we allow the data owner and the CSP to share a root secret key s in advance, with which the CSP can calculate required PRE keys based on its own time and re-encrypt corresponding ciphertext automatically. Specifically, at any time, each attribute a is associated with one initial public key PK_a, and three time-based public keys: day-based public key $PK_a^{(y,m,d)}$, month-based public key $PK_a^{(y,m)}$, and year-based public key $PK_a^{(y)}$, each of which denotes a's public key in a particular day (y, m, d), month (y, m), and year (y), respectively. For example, given current time $(2012, 4, 5)$, attribute

a's public keys include PK_a, $PK_a^{(2012,4,5)}$, $PK_a^{(2012,4)}$, and $PK_a^{(2012)}$. In the TimePRE scheme, the original ciphertexts are encrypted by the data owner using the initial public keys of attributes in the data access structure. On receiving a request, the CSP first uses the root secret key s to calculate PRE keys on all attributes in the access structure based on its own time, and then uses these PRE keys to re-encrypt the original ciphertext by updating the initial public keys of all attributes in the access structure to time-based public keys.

We use $s_a^{(y)}$, $s_a^{(y,m)}$, and $s_a^{(y,m,d)}$ to denote the PRE keys on attribute a in time (y), (y,m), and (y,m,d), which can be used to update attribute a's initial public key PK_a to time-based public keys $PK_a^{(y)}$, $PK_a^{(y,m)}$, and $PK_a^{(y,m,d)}$, respectively. Since the PRE key used in our scheme is derived from a root secret key s and the current access time, we use different notations as those in [4]. As shown in Fig. 2-(b), for each attribute a, the CSP can use the root secret key s and the time tree to hierarchically calculate the time-based PRE keys with Equation 1:

$$s_a^{(y)} = H_{s_a}(y) \tag{1a}$$

$$s_a^{(y,m)} = H_{s_a^{(y)}}(m) \tag{1b}$$

$$s_a^{(y,m,d)} = H_{s_a^{(y,m)}}(d) \tag{1c}$$

where $s_a = H_s(a)$, $a, y, m, d \in \{0,1\}^*$ is a string corresponding to a specific attribute, year, month, and day; and $H_s, H_{s_a}, H_{s_a^{(y)}}, H_{s_a^{(y,m)}} : \{0,1\}^* \to \mathbb{Z}_q^*$ are hash functions with indexes s, s_a, $s_a^{(y)}$, and $s_a^{(y,m)}$, respectively.

Furthermore, to incorporate the concept of time to HABE, each user is granted with a set of *time-based user attribute secret keys* (UAK). Each time-based UAK is associated with a user, an attribute, and an effective time period. If user u is eligible for attribute a in day (y,m,d), the data owner first uses the root secret key s to obtain day-based attribute public key $PK_a^{(y,m,d)}$ from initial attribute public key PK_a, and then uses $PK_a^{(y,m,d)}$ to generate a day-based UAK $SK_{u,a}^{(y,m,d)}$ for user u. The same situation holds for the case that user u is eligible for attribute a in a month (y,m) or a year (y).

Return to the application in Section 1, Alice is authorized to possess attributes `Staff` and `CIS`, and her effective time period is (2012), she will be issued time-based UAK as shown in Table 1; Bob is authorized to possess attributes `Student` and `CIS`, and his effective time periods are $(2012,5)$ and $(2012,6)$, he will be issued time-based UAK as shown in Table 2. Given an

Table 1: Alice's Time-Based User Attribute Secret Keys

Key	Description
$SK^{(2012)}_{(\texttt{Alice, Staff})}$	UAK on attribute `Staff` effective in 2012
$SK^{(2012)}_{(\texttt{Alice, CIS})}$	UAK on attribute `CIS` effective in 2012

Table 2: Bob's Time-Based User Attribute Secret Keys

Key	Description
$SK^{(2012,5)}_{(\texttt{Bob, Student})}$	UAK on attribute `Student` effective in $(2012, 5)$
$SK^{(2012,6)}_{(\texttt{Bob, Student})}$	UAK on attribute `Student` effective in $(2012, 6)$
$SK^{(2012,5)}_{(\texttt{Bob, CIS})}$	UAK on attribute `CIS` effective in $(2012, 5)$
$SK^{(2012,6)}_{(\texttt{Bob, CIS})}$	UAK on attribute `CIS` effective in $(2012, 6)$

access time $(2012, 7, 1)$ and data F with access structure $\mathbb{A} = \{(\texttt{Student} \wedge \texttt{CIS}) \vee \texttt{Staff}\}$, the CSP will use the root secret key s to calculate the PRE keys in (2012), $(2012, 7)$, and $(2012, 7, 1)$ for all attributes in $\mathbb{A}$, say $\{s^{(2012)}_{\texttt{Student}}\}$, $\{s^{(2012,7)}_{\texttt{Student}}\}$, $\{s^{(2012,7,1)}_{\texttt{Student}}\}$, $\{s^{(2012)}_{\texttt{CIS}}\}$, $\{s^{(2012,7)}_{\texttt{CIS}}\}$,$\{s^{(2012,7,1)}_{\texttt{CIS}}\}$, $\{s^{(2012)}_{\texttt{Staff}}\}$, $\{s^{(2012,7)}_{\texttt{Staff}}\}$, $\{s^{(2012,7,1)}_{\texttt{Staff}}\}$. Then, it will use these PRE keys to re-encrypt original ciphertext by updating initial public keys $\{PK_{\texttt{Student}}, PK_{\texttt{CIS}}, PK_{\texttt{Staff}}\}$ to year-based attribute public keys $\{PK^{(2012)}_{\texttt{Student}}, PK^{(2012)}_{\texttt{CIS}}, PK^{(2012)}_{\texttt{Staff}}\}$, month-based attribute public keys $\{PK^{(2012,7)}_{\texttt{Student}}, PK^{(2012,7)}_{\texttt{CIS}}, PK^{(2012,7)}_{\texttt{Staff}}\}$, and day-based attribute public keys $\{PK^{(2012,7,1)}_{\texttt{Student}}, PK^{(2012,7,1)}_{\texttt{CIS}}, PK^{(2012,7,1)}_{\texttt{Staff}}\}$. Given the re-encrypted ciphertext, only the users who possess $\{SK^{(2012)}_{u,\texttt{Student}}, SK^{(2012)}_{u,\texttt{CIS}}\}$ (or $\{SK^{(2012)}_{u,\texttt{Staff}}\}$, or $\{SK^{(2012,7)}_{u,\texttt{Student}}, SK^{(2012,7)}_{u,\texttt{CIS}}\}$, or $\{SK^{(2012,7)}_{u,\texttt{Staff}}\}$, or $\{SK^{(2012,7,1)}_{u,\texttt{Student}}, SK^{(2012,7,1)}_{u,\texttt{CIS}}\}$, or $\{SK^{(2012,7,1)}_{u,\texttt{Staff}}\}$) can recover data F. Therefore, Alice, who possesses year-based UAK $\{SK^{(2012)}_{\texttt{Alice, Staff}}\}$ can recover data F, but Bob, whose effective time periods are overdue in $(2012, 7, 1)$ cannot recover data F any more.

Remarks: (1) The CSP needs to keep the original ciphertexts for re-encryption, but only returns the re-encrypted ciphertext to the users. In Section 7, we will prove that given the original ciphertext, either the CSP or the malicious users cannot know underlying data. (2) A user's effective time period "satisfies" the access time means that the user's access right is effective in the access time. The actual time is accurate to the day. Thus, either effective time period (y, m, d), or (y, m), or (y) satisfies access time (y, m, d). For example, given a day $(2012, 4, 5)$, either year (2012), month

Table 3: Summary of Notations

Notation	Description
K	Security parameter
$\mathbb{UA}$	Universal attributes
PK	System public key
MK	System master key
s	Root secret key
PK_a	Initial public key of attribute a
sk_a	Initial secret key of attribute a
T	A specific day (y, m, d), month (y, m), or year (y)
PK_a^T	Time-based public key of attribute a [2]
$\mathbb{A}$	Data access structure
t	Data access time[3]
T_u	An effective time period with user u[4]
PK_u	User public key
SK_u	User identity secret key (UIK)
$SK_{u,a}^{T_u}$	Time-based user attribute secret key (UAK) [5]
$\subseteq$	Satisfying a condition

$(2012, 12)$, and day $(2012, 4, 5)$ satisfy the access time. (3) If in a day, a data is accessed for multiple times, the CSP only needs to re-encrypt the data file for the first time. The re-encrypted ciphertext can be used in the whole day.

5.2. Definition of the TimePRE scheme

First, we provide the summary of the most relevant notations used in our scheme, as shown in Table 3, to serve as a quick reference. Then, we define the TimePRE scheme by describing the following five algorithms:

1. $Setup(K, \mathbb{UA}) \rightarrow (PK, MK, s)$: The data owner takes a sufficiently large security parameter K as input to generate the system public key

[2] If T is a particular day (y, m, d), $PK_a^T = PK_a^{(y,m,d)}$; If T is a particular month (y, m), $PK_a^T = PK_a^{(y,m)}$; If T is a particular year (y), $PK_a^T = PK_a^{(y)}$.

[3] Data access time is a particular day (y, m, d).

[4] An effective time period associated with user u, which may be a particular day (y, m, d), month (y, m), or year (y).

[5] If T_u is a particular day (y, m, d), $SK_{u,a}^{T_u} = SK_{u,a}^{(y,m,d)}$; If T is a particular month (y, m), $SK_{u,a}^{T_u} = SK_{u,a}^{(y,m)}$; If T is a particular year (y), $SK_{u,a}^{T_u} = SK_{u,a}^{(y)}$.

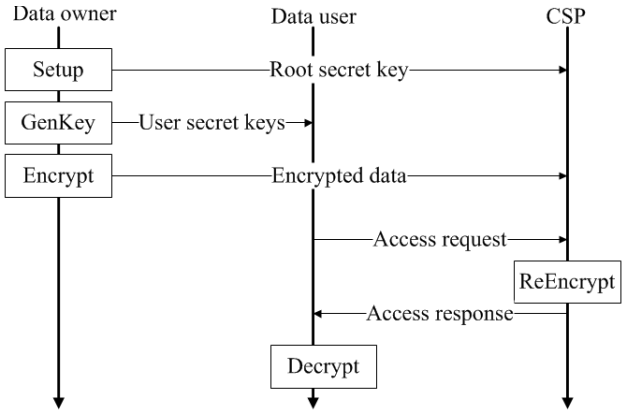

Figure 3: The working process of the TimePRE scheme.

PK, the system master key MK, and the root secret key s. The system public key will be published, the system master key will be kept secret, and the root secret key will be sent to the CSP.

2. $GenKey(PK, MK, s, PK_u, a, T_u) \rightarrow (SK_u, SK_{u,a}^{T_u})$: Suppose that user u with public key PK_u is eligible for attribute a and his access right is effective in time T_u. The data owner uses the system public key PK, the system master key MK, the root secret key s, user public key PK_u, attribute a, and effective time period T_u to generates user identity secret key (UIK) SK_u and time-based user attribute secret key (UAK) $SK_{u,a}^{T_u}$ for u.
3. $Encrypt(PK, \mathbb{A}, F) \rightarrow (C_{\mathbb{A}})$: The data owner takes a DNF access structure $\mathbb{A}$, a data F, and system public key PK, e.g., initial public keys of all attributes in the access structure $\{PK_a\}_{a \in \mathbb{A}}$ as inputs to output a ciphertext $C_{\mathbb{A}}$.
4. $ReEncrypt(C_{\mathbb{A}}, PK, s, t) \rightarrow (C_{\mathbb{A}}^t)$: Given a ciphertext $C_{\mathbb{A}}$ with structure $\mathbb{A}$, the CSP first uses the system public key PK and the root secret key s to generate PRE keys on all attributes in the access structure $\mathbb{A}$ based on the access time t, and then uses these PRE keys to re-encrypt the original ciphertext $C_{\mathbb{A}}$ to $C_{\mathbb{A}}^t$.
5. $Decrypt(PK, C_{\mathbb{A}}^t, SK_u, \{SK_{u,a}^{T_u}\}_{a \subseteq \mathbb{A}, T_u \subseteq t}) \rightarrow (F)$: User u, whose attributes satisfy the access structure $\mathbb{A}$, and whose effective time period T_u satisfy the access time t, can use SK_u and $\{SK_{u,a}^{T_u}\}_{a \subseteq \mathbb{A}}$ to recover F from $C_{\mathbb{A}}^t$.

The working process of the TimePRE scheme is shown in Fig. 3. We will provide a system-level description for the proposed scheme as follows:

1. System Setup. The data owner runs the *Setup* algorithm to generate the system public key PK, the system master key MK, and the root secret key s. It then sends PK and s along with its signatures on each component of these messages to the CSP through a trusted and authenticated channel.

2. Data Creation. Before outsourcing a data to the cloud, the data owner processes the data as follows: (1) Define a DNF attribute-based access structure for the data. (2) Select a unique ID as the keyword of the data. (3) Encode the data as that in [2], e.g, each data is divided into blocks of 1KB size and can be queried using the selected keyword. (4) Encrypt each block using the *Encrypt* algorithm. The CSP maintains a cloud user list (CUL), which records all the authorized data owners. On receiving an encrypted data, it first verifies if the sender is a valid user in CUL. If true, it duplicates and distributes the ciphertext to cloud servers as that in [8].

3. User Grant. When a new user u with public key PK_u joins the system, the data owner, first assigns a set of attributes and a set of effective time periods to the user, and runs the *GenKey* algorithm to generate a user identity secret key (UIK) and a set of user attribute secret keys (UAKs). Finally, the data owner sends the system public key PK and the generated keys along with its signatures on each component of these messages to user u over a trusted and authenticated channel. On receiving the ciphertext, user u verifies the signatures. If correct, he accepts these secret keys. Using the UIK, user u can log into the system.

5. Data Access. If user u wants to retrieve data F, he will first login the system using his UIK, and then send a request to the CSP including the keyword of data F. The CSP, on receiving the data access request, first determines current time. Then, it runs the *ReEncrypt* algorithm to generate PRE keys on all attributes in the access structure, and uses these PRE keys to re-encrypt the ciphertext, i.e., updating the access time associated with the ciphertext to the time it receives the data access request. Finally, it sends the re-encrypted ciphertext to user u. If user u's attributes satisfy the access structure, and his access right is effective in the access time, he can run the *Decrypt* algorithm to recover data.

6. Constructions

In this section, we will provide a detailed construction for the TimePRE scheme as follows:

1. $Setup(K, \mathbb{UA}) \rightarrow (PK, MK, s)$: The data owner takes a security parameter K and the universal attribute $\mathbb{UA}$ as inputs, and outputs the system public key PK, the system master key MK, and a root secret key $s \in \mathbb{Z}_q^*$ as follows:

$$PK = (\{PK_a\}_{a \in \mathbb{UA}}, q, \mathbb{G}_1, \mathbb{G}_2, Q_0, \hat{e}, P_0, P_1)$$
$$MK = (\{sk_a\}_{a \in \mathbb{UA}}, mk_0, mk_1, SK_1)$$

where $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e})$ are the outputs of a BDH parameter generator [5] $\mathcal{IG}$, P_0 is a random generator of $\mathbb{G}_1$, P_1 is a random element in $\mathbb{G}_1$; $sk_a \in \mathbb{Z}_q^*$ is the initial secret key of attribute a and $PK_a = sk_a P_0 \in \mathbb{G}_1$ is the initial public key of attribute a; mk_0 and mk_1 are random elements in $\mathbb{Z}_q^*$, $Q_0 = mk_0 P_0 \in \mathbb{G}_1$, and $SK_1 = mk_0 P_1 \in \mathbb{G}_1$. PK will be published, MK will be kept secret, and s will be sent to the CSP.

2. $GenKey(PK, MK, s, PK_u, a, T_u) \rightarrow (SK_u, SK_{u,a}^{T_u})$: After authenticating user u is eligible for attribute a and his access right is effective in time period T_u, the data owner takes the system public key PK, the system master key MK, user public key PK_u, and the root secret key s as inputs, and generates a UIK SK_u and a time-based UAK $SK_{u,a}^{T_u}$, as follows:

$$SK_u = mk_1 mk_u P_0$$
$$SK_{u,a}^{T_u} = SK_1 + mk_1 mk_u PK_a^{T_u}$$

where $mk_u = H_1(PK_u) \in \mathbb{Z}_q^*$, $H_1 \colon \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$ is a hash function which can be modeled as random oracle, and $PK_a^{T_{u,a}}$ is the time-based public key of attribute a in time T_u. Specifically, we have the following three cases: (1) T_u is a particular day (y, m, d) and $SK_{u,a}^{(y,m,d)} = SK_1 + mk_1 mk_u PK_a^{(y,m,d)}$; (2) T_u is a particular month (y, m) and $SK_{u,a}^{(y,m)} = SK_1 + mk_1 mk_u PK_a^{(y,m)}$; (3) T_u is a particular year (y) and $SK_{u,a}^{(y)} = SK_1 + mk_1 mk_u PK_a^{(y)}$.

Here, time-based attribute public keys can be calculated with Equation 2:

$$PK_a^{(y)} = PK_a + s_a^{(y)} P_0 \tag{2a}$$
$$PK_a^{(y,m)} = PK_a + s_a^{(y,m)} P_0 \tag{2b}$$
$$PK_a^{(y,m,d)} = PK_a + s_a^{(y,m,d)} P_0 \tag{2c}$$

where PRE keys $s_a^{(y)}, s_a^{(y,m)}, s_a^{(y,m,d)}$ can be calculated with Equation 1.

3. *Encrypt*$(PK, \mathbb{A}, F) \rightarrow (C_{\mathbb{A}})$: This algorithm is the same as the *Encryption* algorithm in HABE. The data owner encrypts data $F \in \mathbb{G}_2$ with access structure $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i) = \bigvee_{i=1}^{N}(\bigwedge_{j=1}^{n_i} a_{ij})$ as follows: It first picks a random element $r \in \mathbb{Z}_q^*$, and then sets $n_{\mathbb{A}}$ to be the lowest common multiple (LCM) of $n_1, \ldots, n_N$. Finally, it calculates Equation 3 to produce the ciphertext:

$$U_0 = rP_0, \tag{3a}$$

$$\{U_i = r \sum_{a \in CC_i} PK_a\}_{1 \leq i \leq N}, \tag{3b}$$

$$V = F \cdot \hat{e}(Q_0, rn_{\mathbb{A}}P_1) \tag{3c}$$

The original ciphertext is set to $C_{\mathbb{A}} = (\mathbb{A}, U_0, \{U_i\}_{1 \leq i \leq N}, V)$.

4. *ReEncrypt*$(C_{\mathbb{A}}, s, t) \rightarrow (C_{\mathbb{A}}^t)$: On receiving a user's request for data F, the CSP first determines current time, say $t = (y, m, d)$. Then, it uses the root secret key s and access time t to re-encrypt the original ciphertext $C_{\mathbb{A}}$ with Equation 4:

$$U_0^t = U_0 + r'P_0, \tag{4a}$$

$$U_{(y)i}^t = \sum_{a \in CC_i} (U_i + r'PK_a + s_a^{(y)}U_0^t), \tag{4b}$$

$$U_{(y,m)i}^t = \sum_{a \in CC_i} (U_i + r'PK_a + s_a^{(y,m)}U_0^t), \tag{4c}$$

$$U_{(y,m,d)i}^t = \sum_{a \in CC_i} (U_i + r'PK_a + s_a^{(y,m,d)}U_0^t), \tag{4d}$$

$$V^t = V \cdot \hat{e}(Q_0, r'n_{\mathbb{A}}P_1) \tag{4e}$$

where r' is randomly chosen from $\mathbb{Z}_q^*$ and the PRE keys $s_a^y, s_a^{(y,m)}, s_a^{(y,m,d)}$ can be calculated with Equation 1. The ciphertext that is re-encrypted in time t is set to $C_{\mathbb{A}}^t = (\mathbb{A}, t, U_0^t, \{U_{(y)i}^t\}_{1 \leq i \leq N}, \{U_{(y,m)i}^t\}_{1 \leq i \leq N}, \{U_{(y,m,d)i}^t\}_{1 \leq i \leq N}, V^t)$.

5. *Decrypt*$(PK, C_{\mathbb{A}}^t, SK_u, \{SK_{u,a}^{T_u}\}_{a \subseteq \mathbb{A}, T_u \subseteq t}) \rightarrow (F)$: Given ciphertext $C_{\mathbb{A}}^t$, user u, whose attributes satisfy the *i-th* conjunctive clause CC_i and whose effective time period T_u satisfies the access time t, uses his UIK SK_u and UAKs $\{SK_{u,a}^{T_u}\}_{a \subseteq \mathbb{A}, T_u \subseteq t}$ to recover data F with Equation 5:

$$F = V^t / \frac{\hat{e}(U_0^t, \frac{n_{\mathbb{A}}}{n_i} \sum_{a \in CC_i} SK_{u,a}^{T_u})}{\hat{e}(SK_u, \frac{n_{\mathbb{A}}}{n_i} U_{T_u i}^t)} \tag{5}$$

Specifically, there are three cases:
(1) T_u is a particular day (y, m, d) and Equation 5 is equivalent to:

$$F = V^t / \frac{\hat{e}(U_0^t, \frac{n_{\mathbb{A}}}{n_i} \sum_{a \in CC_i} SK_{u,a}^{(y,m,d)})}{\hat{e}(SK_u, \frac{n_{\mathbb{A}}}{n_i} U_{(y,m,d)i}^t)}$$

(2) T_u is a particular month (y, m) and Equation 5 is equivalent to:

$$F = V^t / \frac{\hat{e}(U_0^t, \frac{n_{\mathbb{A}}}{n_i} \sum_{a \in CC_i} SK_{u,a}^{(y,m)})}{\hat{e}(SK_u, \frac{n_{\mathbb{A}}}{n_i} U_{(y,m)i}^t)}$$

(3) T_u is a particular year (y) and Equation 5 is equivalent to:

$$F = V^t / \frac{\hat{e}(U_0^t, \frac{n_{\mathbb{A}}}{n_i} \sum_{a \in CC_i} SK_{u,a}^{(y)})}{\hat{e}(SK_u, \frac{n_{\mathbb{A}}}{n_i} U_{(y)i}^t)}$$

The key technique of the TimePRE scheme is that the root secret key s is simultaneously used by the data owner to generate time-based UAKs, and by the CSP to generate PRE keys. Note that each time-based UAK is generated with time-based attribute public key, which is in turn generated by s and an effective time period T_u; each data is first encrypted with initial attribute public keys, and will be updated by the CSP to day-based attribute public keys, which are in turn generated by s and the access time of receiving a request t. Therefore, even if s is only shared between the data owner and the CSP, the users still can decrypt the ciphertext when their attributes satisfy the access structure and their access rights are effective in the access time. Furthermore, the *GenKey* algorithm should take the system master key MK as inputs, which is kept secret by the data owner. Thus, given the root secret key that has nothing to do with the system master key, the CSP cannot know any information about the UAKs.

7. Analysis

In this section, we will first provide a correctness analysis for the TimePRE scheme, and then we will analyze the performance of the TimePRE scheme. Finally, we will provide an intuitive security proof for our scheme.

7.1. Correctness Analysis

To prove the correctness of the TimePRE scheme, we should prove that given a re-encrypted ciphertext, the users, whose attributes satisfy the access structure and whose access rights are effective in the access time, can successfully recover the data.

This is equivalent to proving that Equation 5 in the *Decrypt* algorithm can successfully recover F. Note that the ciphertext produced by Equation 4 in the *ReEncrypt* algorithm can further evolve into the following form:

$$
\begin{aligned}
&U_0^t = U_0 + r^{'} P_0 = (r + r^{'}) P_0, \\
&U_{(y)i}^t = \textstyle\sum_{a \in CC_i} (U_i + r^{'} PK_a + s^{(y)} U_0^t) = \sum_{a \in CC_i} ((r + r') PK_a^{(y)}), \\
&U_{(y,m)i}^t = \textstyle\sum_{a \in CC_i} (U_i + r^{'} PK_a + s^{(y,m)} U_0^t) = \sum_{a \in CC_i} ((r + r') PK_a^{(y,m)}), \\
&U_{(y,m,d)i}^t = \textstyle\sum_{a \in CC_i} (U_i + r^{'} PK_a + s^{(y,m,d)} U_0^t) = \sum_{a \in CC_i} ((r + r') PK_a^{(y,m,d)}), \\
&V^t = V \cdot \hat{e}(Q_0, r^{'} n_{\mathbb{A}} P_1) = \hat{e}(Q_0, (r + r') n_{\mathbb{A}} P_1)
\end{aligned}
$$

Suppose user u's effective time period that satisfies the access time t is a particular day. That is $T_u = (y, m, d)$. We can prove that case (1) of Equation 5 is correct as follows:

$$
\begin{aligned}
& V^t / \frac{\hat{e}(U_0^t, \frac{n_{\mathbb{A}}}{n_i} \sum\limits_{a \in CC_i} SK_{u,a}^{(y,m,d)})}{\hat{e}(SK_u, \frac{n_{\mathbb{A}}}{n_i} U_{(y,m,d)i}^t)} \\
&= V^t / \frac{\hat{e}((r + r') P_0, \frac{n_{\mathbb{A}}}{n_i} \sum\limits_{a \in CC_i} (SK_1 + mk_1 mk_u PK_a^{(y,m,d)}))}{\hat{e}(mk_1 mk_u P_0, \frac{n_{\mathbb{A}}}{n_i} (r + r') PK_a^{(y,m,d)})} \\
&= V^t / \frac{\hat{e}((r + r') P_0, n_{\mathbb{A}} SK_1) \hat{e}((r + r') P_0, \frac{n_{\mathbb{A}}}{n_i} mk_1 mk_u \sum\limits_{a \in CC_i} PK_a^{(y,m,d)})}{\hat{e}(mk_1 mk_u P_0, \frac{n_{\mathbb{A}}}{n_i} (r + r') \sum\limits_{a \in CC_i} PK_a^{(y,m,d)})} \\
&= V^t / \frac{\hat{e}((r + r') P_0, n_{\mathbb{A}} SK_1) \hat{e}(mk_1 mk_u P_0, \frac{n_{\mathbb{A}}}{n_i} (r + r') \sum\limits_{a \in CC_i} PK_a^{(y,m,d)})}{\hat{e}(mk_1 mk_u P_0, \frac{n_{\mathbb{A}}}{n_i} (r + r') \sum\limits_{a \in CC_i} PK_a^{(y,m,d)})} \\
&= V^t / \hat{e}((r + r') P_0, n_{\mathbb{A}} SK_1) \\
&= F \cdot \hat{e}(Q_0, (r + r') n_{\mathbb{A}} P_1) / \hat{e}(Q_0, (r + r') n_{\mathbb{A}} P_1) \\
&= F
\end{aligned}
$$

Table 4: Comparisons of CP-ABE Schemes

Properties	Reference [3]	Reference [19]	Our Scheme
User Key Size	$O(2n)$	$O(n)$	$O(nm)$
Ciphertext	$O(2S)$	$O(3N)$	$O(N)$
Encryption (exp)	$O(2N)$	$O(3N)$	$O(N)$
Decryption (map)	$O(2P)$	$O(1)$	$O(1)$

In the same way, when $T_u = (y, m)$, we can prove that cases (2) of Equation 5 is correct; when $T_u = (y)$, we can prove that cases (3) of Equation 5 is correct. Therefore, the TimePRE scheme is correct.

7.2. Performance Analysis

The efficiency of the *Setup* algorithms is rather straightforward. Therefore, we only analyze the costs introduced by algorithms *GenKey*, *Encrypt*, *ReEncrypt*, and *Decrypt*. If a user is identified by n attributes and his effective time periods correspond to m nodes in the time tree, the *GenKey* algorithm requires the data owner to execute $O(mn)$ point multiplications to generate secret keys of $O(mn)$ length. However, if we deliberately design the time tree, m can be limited to a relatively small value.

To encrypt data F under a DNF access control structure $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i)$, a user needs to compute one bilinear map of Q_0 and P_1, and $O(N)$ number of exponentiation operations to output a ciphertext of $O(N)$ length. Notice that the computation for the bilinear map of Q_0 and P_1 is independent of the message to be encrypted, and hence can be done once for all. To recover F, a user, whose attributes satisfy the access structure and whose access right is effective in the access time, needs to execute $O(1)$ bilinear map operations.

In Table 4, we briefly compare our scheme with other CP-ABE schemes [3, 19]. We believe that the most expensive computation is bilinear map, abbreviated as *map*; the next is the exponentiation, abbreviated as *exp*. In Table 4, n is the number of attributes associated with a user, m is the number of nodes in the time tree corresponding to a user's effective time periods, S is the number of attributes in an access structure, N is the number of conjunctive clauses in an access structure, and P is the number of attributes in an access structure that is matched by attributes in a user's secret key.

Then, we briefly compare our scheme with the work in references [29, 28], which also allow the CSP to be delegated to execute re-encryption. We first

Table 5: Comparisons of Re-encryption Cost on Data Owner

Properties	Reference [29]	Reference [28]	Our Scheme
Number of PRE keys	$O(n)$	$O(n)$	0
Number of update keys	$O(nw)$	$O(nw)$	0

Table 6: Comparisons of Re-encryption Cost on CSP

Properties	Reference [29]	Reference [28]	Our Scheme
Exp	$O(n)$	$O(n)$	$O(6N)$
Map	$O(0)$	$O(0)$	$O(1)$

consider the workload on the data owner in the user revocation, as shown in Table 5, where n is the number of attributes associated with a user and w is the number of remaining authorized users. In the TimePRE scheme, the data owner has nothing to do when a user is revoked. The *ReEncrypt* algorithm run by the CSP is without any involvement of the data owner.

Then, we compare the re-encryption costs incurred at the CSP, as shown in Table 6, where n is the number of attributes associated with a revoked user and N is the number of conjunctive clauses in an access structure. To re-encrypt a ciphertext $C_{\mathbb{A}}$ based on the access time $t = (y, m, d)$, our scheme requires the CSP to execute $O(6N)$ exponentiation operations and one bilinear map operation to re-encrypt a ciphertext. Since the CSP can batch the re-encryption operations and re-encrypt data only when receiving a data access request, the re-encryption cost is relatively low.

7.3. Security Analysis

The *Encrypt* algorithm in the TimePRE scheme is the same as the *Encryption* algorithm in HABE, which has been proven to be semantically secure in [28]. Therefore, we consider that the TimePRE scheme is secure if the following propositions hold:

- **Proposition 1.** The keys produced by the *GenKey* algorithm are secure.
- **Proposition 2.** The ciphertext produced by the *ReEncrypt* algorithm is semantically secure.
- **Proposition 3.** Given the root secret key and the original ciphertext, the CSP cannot know neither the underlying data, nor UAKs while

executing re-encryption.

For Proposition 1, we prove that the *GenKey* algorithm is as secure as the *Key Generation* algorithm in HABE. First, the way to generate UIK is the same in both algorithms. Then, given the system public key PK, the system master key MK, user public key PK_u, and attribute a, if the data owner takes the time-based attribute public key $PK_a^{(T_u)}$ as inputs of the *Key Generation* algorithm in HABE, then the produced UAK is the same as that of the *GenKey* algorithm that takes time T_u, the initial attribute public key PK_a, and the root secret key s as inputs. As proven in [28], due to the BDH assumption, the malicious users cannot obtain MK, even if all of them collude. Therefore, Proposition 1 is correct.

For Proposition 2, we prove that the *ReEncrypt* algorithm is as secure as the *Encryption* algorithm in HABE. Given system public key PK and data F with access structure $\mathbb{A}$, if the data owner takes the time-based attribute public keys $\{PK_a^{(y)}\}_{a\in\mathbb{A}}$, $\{PK_a^{(y,m)}\}_{a\in\mathbb{A}}$, $\{PK_a^{(y,m,d)}\}_{a\in\mathbb{A}}$, and a random number $r'' = r + r'$ as inputs of the *Encryption* algorithm in HABE, then the produced ciphertext is the same as that of the the *ReEncrypt* algorithm that takes time $t = (y, m, d)$, the original ciphertext $C_{\mathbb{A}}$, and root secret key s as inputs. Therefore, Proposition 2 is correct.

For completeness, we provide an intuitive security proof for the *ReEncrypt* algorithm as follows:

Recall that data F is re-encrypted to $V^t = F \cdot \hat{e}(Q_0, (r + r')n_{\mathbb{A}}P_1)$ in time $t = (y, m, d)$. Therefore, an adversary $\mathcal{A}$ needs to construct $\hat{e}(Q_0, (r + r')n_{\mathbb{A}}P_1) = \hat{e}(U_0^t, SK_1)^{n_{\mathbb{A}}}$ to recover F. From the *GenKey* algorithm, we know that the only occurrence of SK_1 is in the UAKs. In our security model, we assume that the CSP will not collude with the malicious users, who possess UAKs. Therefore, we only consider the case that malicious users work independently, or collude to compromise data security.

We consider that the TimePRE scheme is insecure if one of the following cases happens: (1) Adversary $\mathcal{A}$, whose effective time period satisfies the access time, but whose attributes do not satisfy the access control, can recover data F. (2) Adversary $\mathcal{A}$, whose attributes satisfy the access control, but whose effective time does not satisfy the access time, can recover data F.

For case (1), we have the following assumptions for ease of presentation: Adversary $\mathcal{A}$ has requested UAKs on all but one of the attributes $a_{i1}, \ldots, a_{i(k-1)}, a_{i(k+1)}, \ldots, a_{in_i}$ in CC_i for user u, and has requested a UAK on the missing attribute a_{ik} for user $u^{'}$. Both users' effective time periods

T_u and $T_{u'}$ satisfy the access time $t = (y, m, d)$. Based on Proposition 1, we know that the adversary cannot generate fake keys. The only occurrence of SK_1 is in the UAKs, so the adversary has to use UAKs requested for user u and $u^{'}$ for bilinear map, yielding for some α:

$$\begin{aligned}&\hat{e}(U_0^t, \tfrac{n_{\mathbb{A}}}{n_i} \sum_{j=1,j\neq k}^{n_i} SK_{u,a_{ij}}^{T_u} + \tfrac{n_{\mathbb{A}}}{n_i} SK_{u^{'},a_{ik}}^{T_{u'}} + \alpha)\\ &= \hat{e}(U_0^t, SK_1)^{n_{\mathbb{A}}} \hat{e}(r''P_0, \alpha) \hat{e}(SK_{u'}, r''PK_{a_{ik}}^{T_{u'}})^{\frac{n_{\mathbb{A}}}{n_i}} \hat{e}(SK_u, r'' \sum_{j=1,j\neq k}^{n_i} PK_{a_{ij}}^{T_u})^{\frac{n_{\mathbb{A}}}{n_i}}\end{aligned}$$

where $r'' = r + r'$. To obtain $\hat{e}(U_0, SK_1)^{n_{\mathbb{A}}}$, the last three elements have to be eliminated. Note that $SK_{u'}$ and SK_u are known to adversary $\mathcal{A}$, but r is randomly chosen by the data owner for the original ciphertext $C_{\mathbb{A}}$ and r' is randomly chosen by the CSP for the re-encrypted ciphertext $C_{\mathbb{A}}^t$. The adversary cannot know $r''PK_{a_{ik}}^{T_{u'}}$ or $r'' \sum_{j=1,j\neq k}^{n_i} PK_{a_{ij}}^{T_u}$, even if he knows $U_{(T_u)i}^t$ and $U_{(T_{u'})i}^t$ due to the BDH assumption. Therefore, adversary $\mathcal{A}$ cannot recover the data from V^t.

For case (2), we have the following assumptions for ease of presentation: Adversary $\mathcal{A}$ has requested UAKs on all attributes in CC_i for user u. Any effective time period T_u of this user does not satisfy the access time $t = (y, m, d)$. Based on Proposition 1, we know that the adversary cannot generate fake keys. The only occurrence of SK_1 is in the UAKs, so the adversary has to use UAKs requested for user u for bilinear map, yielding for some α:

$$\begin{aligned}&\hat{e}(U_0^t, \tfrac{n_{\mathbb{A}}}{n_i} \sum_{j=1}^{n_i} SK_{u,a_{ij}}^{T_u} + \alpha)\\ &= \hat{e}(U_0^t, SK_1)^{n_{\mathbb{A}}} \hat{e}(r''P_0, \alpha) \hat{e}(SK_u, r'' \sum_{j=1}^{n_i} PK_{a_{ij}}^{T_u})^{\frac{n_{\mathbb{A}}}{n_i}}\end{aligned}$$

where $r'' = r + r'$. To obtain $\hat{e}(U_0, SK_1)^{n_{\mathbb{A}}}$, the last two elements have to be eliminated. Note that the SK_u is known to adversary $\mathcal{A}$, but r is randomly chosen by the data owner for the original ciphertext $C_{\mathbb{A}}$ and r' is randomly chosen by the CSP for the re-encrypted ciphertext $C_{\mathbb{A}}^t$. The adversary cannot know $r'' \sum_{j=1,j\neq k}^{n_i} P_{a_{ij}}^{T_u}$, even if he knows U_i^t and $U_i^{T_u}$ due to the BDH assumption. Therefore, adversary $\mathcal{A}$ cannot recover data from V^t.

For Proposition 3, we first prove that the CSP cannot derive the system master key MK and UAKs from the root secret key s. As compared to

HABE, the TimePRE scheme discloses an additional root secret key to the CSP, which is randomly chosen by the data owner and has nothing to do with the system master key. Therefore, the CSP cannot derive the system master key from the root secret key. Based on Proposition 1, the CSP cannot obtain UAKs without the system master key.

Then, we prove that the CSP cannot compromise data security given the original ciphertext. Note that the original ciphertext is encrypted with the *Encrypt* algorithm, which is semantically secure. Therefore, the ciphertext can be decrypted by only the entity who possesses UAKs on the initial attribute public keys. In the TimePRE scheme, a users' UAKs are generated on the time-based attribute public key, rather than the initial attribute public key. Therefore, only the data owner with the initial attribute secret keys can recover the data from the original ciphertext. Neither the users, nor the CSP can decrypt the original ciphertext.

8. Discussion

8.1. Different Length of Effective Time

The design goal of this paper is to achieve scalable user revocation. Thus, in the TimePRE scheme, the effective time periods of all attributes associated with a user are the same. Some applications may require that different attributes are associated with different effective time periods. Our scheme cannot be directly applied to such an environment.

To illustrate, let us assume that data F is associated with access structure $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i) = \bigvee_{i=1}^{N}(\bigwedge_{j=1}^{n_i} a_{ij})$ and a user u possesses all attributes in CC_i, where the effective time period of $a_{i1}, \ldots, a_{i(k-1)}, a_{i(k+1)}, \ldots, a_{in_i}$ is (y, m, d) and the effective time period of a_{ik} is (y). Actually, user u has rights to access data F in time (y, m, d). But with the TimePRE scheme, he cannot recover the data, since he does not have sufficient day-based UAKs, nor year-based UAKs on attributes in CC_i.

We provide a possible improvement for the TimePRE scheme, which requires the data owner to generate additional UAKs for each user in the *GenKey* algorithm. The basic idea is that if user u is eligible for attribute a_1 in time T_{u,a_1}, and for attribute a_2 in time T_{u,a_2}, where the node in the time tree corresponding to T_{u,a_1} is an ancestor of that corresponds to T_{u,a_2}, then the data owner should generate $SK_{u,a_1}^{T_{u,a_2}}$ in addition to $SK_{u,a_1}^{T_{u,a_1}}$ and $SK_{u,a_2}^{T_{u,a_2}}$

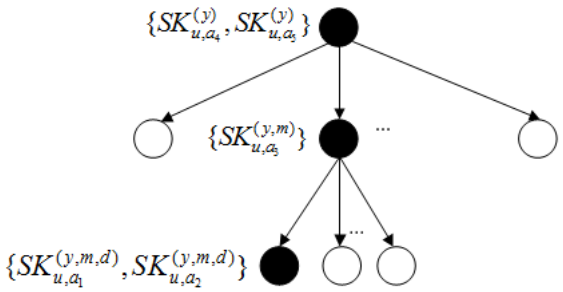

Figure 4: The solid nodes in the time tree correspond to the effective time periods associated with a user's UAKs.

for user u. Specifically, the data owner will first generate UAKs with different effective time periods to user u. Then, from bottom to top, for each non-leaf node that corresponds to an effective time period associated with an attribute a, if there are descent nodes that correspond to effective time periods associated with other attributes, then the data owner will generates UAKs on attribute a with effective time periods corresponding to the descent nodes for user u. For example, as shown in Fig. 4, the data owner first generates $SK_{u,a_1}^{(y,m,d)}$, $SK_{u,a_2}^{(y,m,d)}$, $SK_{u,a_3}^{(y,m)}$, $SK_{u,a_4}^{(y)}$, and $SK_{u,a_5}^{(y)}$ for user u. Then, for $SK_{u,a_3}^{(y,m)}$, the data owner also generates $SK_{u,a_3}^{(y,m,d)}$, since the user is eligible for attributes a_1, a_2 in the time (y, m, d), which corresponds a descent node of (y, m) in the time tree. In the same way, for $SK_{u,a_4}^{(y)}$, the data owner generates $SK_{u,a_4}^{(y,m)}$ and $SK_{u,a_4}^{(y,m,d)}$; and for $SK_{u,a_5}^{(y)}$, the data owner generates $SK_{u,a_5}^{(y,m)}$ and $SK_{u,a_5}^{(y,m,d)}$.

This solution ensures that a user, whose attributes satisfy the access structure and whose attributes are effective in the access time, has sufficient day-based, month-based, or year-based UAKs to decrypt the ciphertext. The main problem is that each user should maintain additional UAKs. Specifically, if a user initially possesses α year-based UAKs, β month-based UAKs, and γ day-based UAKs, then in the worst case, he will be issued $3\gamma + 2\beta + \alpha$ UAKs in total. However, the total number of UAKs can be limited to a relatively small value by deliberately designing the time tree.

8.2. No Global Time

In the system model, we assume that there is a global time among all entities. Actually, this assumption is hard to achieve in a cloud environment, where the cloud servers may be located all over the world. The solution

proposed in our previous work [17] may be applied to this paper.

Intuitively, we divide time $\{t_1, \ldots, t_l\}$ into time slices of equal length, denoted as $TS_i = [t_i, t_{i+1})$, where $1 \leq i \leq l$. Furthermore, we determine a maximal time difference Δ between the data owner and the cloud server, where Δ is no larger than the duration of one time slice. In other words, when the data owner is at TS_i, the cloud server's time may be TS_{i-1}, TS_i, or TS_{i+1}. On receiving a data access request, a cloud server should re-encrypt the ciphertext with the its current time slice. However, the cloud server will not respond immediately.

We require the data owner and each cloud server to agree on a maximal waiting time α. Each read or write command should be associated with a time slice. In T_i, if the cloud server receives either a read or a write command tagged with TS_i, it will hold on until $t_{i+1} + \alpha$. If the time slice associated with the command is larger than the cloud server's time slice, the cloud server will keep this command in a queue. Otherwise, it will simply discard the command. The scheme in [17] ensures that the data that is updated in T_i can be accessed by only the users whose attributes satisfy the access structure and whose access rights are effective in T_{i+1}. The main drawback of this scheme is that the number of keys issued to each user will grow linearly as the number of effective time slices associated with the user's access right.

8.3. Limitations of the TimePRE Scheme

The TimePRE scheme is more suitable to applications where the period of validity of each user's access right is predetermined and a coarse-grained time accuracy is satisfactory. If a data owner wants to revoke a user from the system at any time, then the schemes proposed in [29, 28] are better choices. For the applications that requires a fine-grained time accuracy, e.g., a second or microsecond, the cost to ensure correctness will be excessive even if we apply the techniques in [17] to the TimePRE scheme.

Finally, our security model assumes that the CSP will not collude with malicious users. The CSP is responsible for re-encryption, i.e., updating the access time associated with a data to the time of receiving a data access request. If the CSP colludes with the malicious users, it may always update the access time to a fake time, so that the revoked users can recover data using their overdue keys. Therefore, the user revocation mechanism loses effectiveness. However, the CSP cannot generate fake UAKs for the malicious users. Thus, malicious users, whose attributes do not satisfy the access structure, cannot recover the data, even if they collude with the CSP.

9. Conclusion

In this paper, we proposed the TimePRE scheme to achieve fine-grained access control and scalable user revocation in a cloud environment. Our scheme enables each user's access right to be effective in a pre-determined period of time, and enable the CSP to re-encrypt ciphertexts automatically, based on its own time. Thus, the data owner can be offline in the process of user revocations.

The main problem with our scheme is that it requires the effective time periods to be the same for all attributes associated with a user. Although we provide a possible improvement, the users will be issued more UAKs. Our future work is to allow different effective time periods for different attributes associated with a user, without increasing the number of UAKs associated with each user.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61073037 & 61272151, and Hunan Provincial Science and Technology Program under Grant Nos. 2010GK2003 & 2010GK3005.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[2] K. Bennett, C. Grothoff, T. Horozov, and I. Patrascu. Efficient sharing of encrypted data. In *Proceedings of the Australian Conference on Information Security and Privacy (ACISP)*, pages 107–120, 2002.

[3] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 321–334, 2007.

[4] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Proceedings of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 127–144, 1998.

[5] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of International Cryptology Conference (CRYPTO)*, pages 213–229, 2001.

[6] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the ACM Workshop on Cloud Computing Security (CCS)*, pages 85–90, 2009.

[7] C. ComPUtING. Cloud computing privacy concerns on our doorstep. *Communications of the ACM*, 54(1):36–38, 2011.

[8] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, pages 205–220, 2007.

[9] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of International Cryptology Conference (CRYPTO)*, pages 10–18, 1984.

[10] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 149–155, 2002.

[11] E.J. Goh, H. Shacham, N. Modadugu, and D. Boneh. Sirius: Securing remote untrusted storage. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, pages 131–145, 2003.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 89–98, 2006.

[13] M. Green and G. Ateniese. Identity-based proxy re-encryption. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, pages 288–306, 2007.

[14] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 29–42, 2003.

[15] S. Kamara and K. Lauter. Cryptographic cloud storage. In *Proceedings of the International Conference on Financial Cryptograpy and Data Security (FC)*, pages 136–149, 2010.

[16] J. Li, Q. Huang, X. Chen, S.S.M. Chow, D.S. Wong, and D. Xie. Multi-authority ciphertext-policy attribute-based encryption with accountability. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 386–390, 2011.

[17] Q. Liu, C.C. Tan, J. Wu, and G. Wang. Reliable re-encryption in unreliable clouds. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2011.

[18] Q. Liu, C.C. Tan, J. Wu, and G. Wang. Efficient information retrieval for ranked queries in cost-effective cloud environments. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[19] S. Müller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In *Proceedings of Annual International Conference on Information Security and Cryptology (ICISC)*, pages 20–36, 2009.

[20] S. Narayan, M. Gagne, and R. Safavi-Naini. Privacy preserving EHR system using attribute-based infrastructure. In *Proceedings of the ACM workshop on Cloud Computing Security (CCS)*, pages 47–52, 2010.

[21] B. Patel and J. Crowcroft. Ticket based service access for the mobile user. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 223–233, 1997.

[22] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. *Journal of Computer Security*, 18(5):799–837, 2010.

[23] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proceedings of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 557–557, 2005.

[24] B. Stone and A. Vance. Companies slowly join cloudcomputing. *New York Times*, 18:2010, 2010.

[25] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11, 2011.

[26] G. Wang, Q. Liu, and J. Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 735–737, 2010.

[27] G. Wang, Q. Liu, and J. Wu. Achieving fine-grained access control for secure data sharing on cloud servers. *Concurrency and Computation: Practice and Experience*, 23(12):1443–1464, 2011.

[28] G. Wang, Q. Liu, J. Wu, and M. Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, 30(5):320–331, 2011.

[29] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 534–542, 2010.

[30] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 261–270, 2010.