IJETST- Vol.||02||Issue||01||Pages 1699-1710||January||ISSN 2348-9480 2015

Open access Journal International Journal of Emerging Trends in Science and Technology

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

Authors

Dayananda RB[1], Prof. Dr. G.Manoj Someswar[2]

[1]Associate Professor, Department of CSE, RRIT, Bangalore–90, Karnataka, India
[2]Principal & Professor, Department of CSE, Anwar-ul-uloom College of Engineering & Technology (Affiliated to JNTU, Hyderabad), Vikarabad – 501101, RR District, Telangana, India

Abstract

In this research paper, we will describe the system model and security model in our scheme and provide our design goals and related assumptions. 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, 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 cipher text based on its own time, and return the re-encrypted cipher text.

Keywords: *Attribute Based Encryption, Hierarchial Attribute Set Based Encryption,Cloud Service Provider, Network Time Protocol, Ciphertext-Policy Attribute-Based Encryption, proxy re-encryption, Collusion Resilience, Fine-grained Data Access Control , Sensor Compromise Resistance, Backward Secrecy*

INTRODUCTION

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 cipher text. If his access right is effective when he requests the data, he can successfully execute decryption.

Since the Time PRE 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 to ensure the Time PRE 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 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 to the Time PRE scheme.

Scope of the Study

Attribute Based Encryption (ABE) provides normal encryption and extra access control function. ABE is more efficient, flexible and suitable than other cryptographic techniques and may be a lightweight security solution for web services [1]. Cloud services are also delivered as web services. Thus, proposes system intends to implement ABE based security in provisioning cloud services to the users. With this approach, confidentiality of service information can be achieved even if control is lost over the service reply during transmission.

Future work lies in implementing the system in the two private-cloud systems in our university campus and proving the effectiveness of research work's proposed system.

The HASBE scheme seamlessly incorporates a hierarchical structure of system users by applying a delegation algorithm to ASBE. HASBE not only supports compound attributes due to flexible attribute set combinations, but also achieves efficient user revocation because of multiple value assignments of attributes. HASBE based on the security of CP-ABE and implemented the scheme, and conducted comprehensive performance analysis and evaluation.

Research Questions

The present attempt in this research paper is to discuss in detail about security and data integrity using attributes based encryption on cloud. The main objectives of this research study are as follows:

1. To carefully analyze Security of Cloud Service Provisioning using Attribute Based Encryption
2. To give overview of Data Sharing on Untrusted Storage with Attribute-Based Encryption
3. To carefully analyze Scalable Access Control in Cloud Computing Using Hierarchical Attribute Set Based Encryption (HASBE)
4. To deeply investigate Time-Based Proxy Re-encryption Scheme for Secure Data Sharing in a Cloud Environment
5. To discuss in detail Enhanced Secure Data Access Model for Public Clouds.
6. To explore FADE: Secure Overlay Cloud Storage with File Assured Deletion
7. To discuss in detail Designing a Secure Cloud-Based EHR System using Ciphertext-Policy Attribute-Based Encryption
8. To give an overview of CP-ABE Based Encryption for Secured Cloud Storage Access

ANALYSIS & DESIGN

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, 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 we assume that the CSP will correctly carry out our scheme to ensure data security.

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) to the Time PRE scheme, so that the CSP can re-encrypt the data only when receiving data access requests from users.

Technique Preliminaries

In this research paper, 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).

Definitions

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

Definition 1 (Bilinear Map): Let G_1 and G_2 be two cyclic groups of some large prime order *q*, where G_1 is an additive group and G_2 is a multiplicative group. A bilinear map, e: G_1 x $G_1 \rightarrow G_2$, 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 *IG* is called a BDH parameter generator if *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 G1 and G_2 of order *q*, and the description of a bilinear map e : G1 x G 1> G_2.

Definition 3 (BDH Problem): Given a random element P $\in$ G1, as well as *aP*, *bP*, and *cP*, for some a, b, c $c \in Z_q^*$, compute $e(P, P)^{abc} \in G_2$.

Definition 4 (BDH Assumption): If *IG* is a BDH parameter generator, the advantage $Adv_{IG}(\square)$ that an algorithm $\square$ has in solving the BDH problem is defined to be the probability that $\square$ outputs $e(P, P)^{abc}$ on inputs *q*, *G1*, G2, e, *P*, *aP*, *bP*, *cP*, where < *q*,*G1*,*G2*,*e* > are the outputs of *IG* for a sufficiently large security parameter *K*, *P* is a

random element ∈ G1, and *a*, *b*, *c* are random elements of Z_q^*. The BDH assumption is that $Adv_{IG}(\square)$ is negligible for any efficient algorithm $\square$.

Proxy Re-Encryption (PRE)

Let us illustrate the motivation of the PRE scheme 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 G be a multiplicative group of prime order *q*, and *g* be a random generator of G. The PRE scheme is consisted of the following algorithms:

Key Generation: Alice can choose a random element *a* $a \in Z_q^*$ as her secret key SK_A, and her public key PK_A is $g^a \in$ G. In the same way, Bob's public/secret key pair (SK_B, PK_B) are (b, g^b). The PRE key $RK_{A\to B} = b/a \pmod q$ 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 $m \in$ G to Alice, the sender randomly chooses $r \in Z_q^*$ 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 $C_{B2} = (C_{A2})$ $(C_{B2}) = (C_{A2})^{RK_{A\to B}}$.

Given the cipher text (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, and thus the cipher text is semantically secure, and given the PRE key, the mail server cannot guess the secret keys *a* nor *b*. Please refer to for more details.

Hierarchical Attribute-Based Encryption (HABE)

Our Time PRE scheme is extended from an efficient CP-ABE system, HABE which is constructed based on the bilinear map. 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). Since our Time PRE 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 UA as inputs, and outputs system public key *PK* and system master key *MK* as follows:

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

where *(q,Gi,G2,e)* are the outputs of a BDH parameter generator *IG*, P_0 is a random generator of G1, and *P1* is a random element in G1; $sk_a \in \mathbb{Z}_q^*$ is the secret key of attribute *a* and $PK_a = sk_a P_0 \in \mathbb{G}_1$ is the public key of attribute *a; mk0* and mk_1 are random elements in $\mathbb{Z}_q^*$, $Q_0 = mk_0 P_0 \in \mathbb{G}_1$,, and $SK_1 = mk_0 P_1 \in \mathbb{G}_1$..

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_{ua}, as follows:

$$SK_u = mk_1 mk_u P_0 \in \mathbb{G}_1$$
$$SK_{u,a} = SK_1 + mk_1 mk_u PK_a \in \mathbb{G}_1$$

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 cipher text = $(A,U_0,U1,...,U_N,V)$ as $C_{\mathbb{A}} = (\mathbb{A}, U_0, U_1, \ldots, U_N, V)$ as follows:

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

where $N \in Z^+$ is the number of conjunctive clauses in A, $ni \in Z^+$ is the number of attributes in the *i-th* conjunctive clause *CCi*, and *aij* is the *j-th* attribute in *CCi; r* is a random element in $\mathbb{Z}_q^*$, and NA 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 *CCi* 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 cipher text 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.

Outline of the Time PRE Scheme

In this research paper, we will first illustrate the main idea of the Time PRE scheme, and then we will provide the formal definition of our scheme.

Main Idea

The main idea of the Time PRE 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 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.

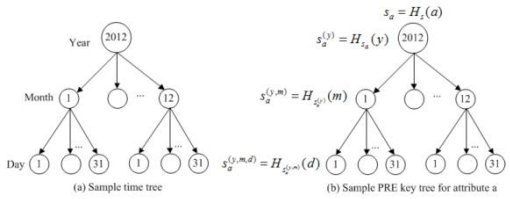

***Figure 1**: Three-level time tree and attribute a's PRE key tree.*

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 Figure 1, 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 cipher text 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, PKa($^{2012,4,5)}$, PKa($^{2012,4)}$, and $PKa^{(2012).}$ In the Time PRE scheme, the original cipher texts 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 cipher text 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 $PK_a^{(y)}$, $PKa^{(y,\ m),}$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. As shown in Figure. 2.9 (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

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

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

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

where $s_a = H_s(a)$, *a,y,m,d* G {0,1}* is a string corresponding to a specific attribute, year, month, and day; and $H_s, H_{s_a}, H_{s_a^{(y)}}, H_{s_a^{(y,m)}} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ are hash functions with indexes *s*, s_a, $sa^{(y)}$, 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 $PKa^{(y,m)}$ from initial attribute public key PK_a, and then uses $PKa^{(y,m,d)}$ 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 2.1; 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 2.2. Given an

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

Table 2.1: *A lice's Time-Based User Attribute Secret Keys*

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

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

***Table 2.2**: Bob's Time-Based User Attribute Secret Keys*

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 cipher texts for re-encryption, but only returns the re-encrypted cipher text to the users. In Section 7, we will prove that given the original cipher text, 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 (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 cipher text can be used in the whole day.

***Table 2.3**: Summary of notation*

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^T_a	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)
$SK^{T_u}_{u,a}$	Time-based user attribute secret key (UAK) [5]
$\subseteq$	Satisfying a condition

Definition of the Time PRE scheme

First, we provide the summary of the most relevant notations used in our scheme, as shown in Table 2.3, to serve as a quick reference. Then, we define the Time PRE scheme by describing the following five algorithms:

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

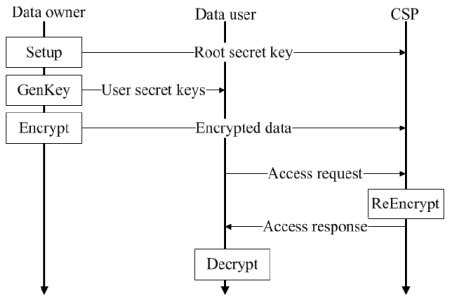

***Figure 2**: The working process of the Time PRE scheme.*

2. *GenKey (PK, MK, s, PK_u, a, T_u)* $\rightarrow (SK_u, SK^{T_u}_{u,a})$: 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) $SKu,_aTu$ for *u*.[3]

Encrypt (PK,A,F) $\rightarrow$ (CA) : The data owner takes a DNF access structure A, a data *F*, and system public key *PK*, e.g., initial public keys of all attributes in the access structure $\{PK_a\}_{a\&A}$ as inputs to output a ciphertext *CA*.

ReEncrypt (CA,PK,s,t) $\rightarrow (C^t_{\mathbb{A}})$: Given a ciphertext CA with structure 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 A based on the access time *t*, and then uses these PRE keys to re-encrypt the original ciphertext CA to $C_{\mathbb{A}}$ to $C^t_{\mathbb{A}}$.

Decrypt $(PK, C^t_{\mathbb{A}}, SK_u, \{SK^{T_u}_{u,a}\}_{a\subseteq\mathbb{A}, T_u\subseteq t}) \rightarrow (F)$: User *u*, whose at-tributes satisfy the access structure A, and whose effective time period T_u satisfy the access time *t*, can use SK_u and $\{SK^{T_u}_{u,a}\}_{a\subseteq\mathbb{A}}$ *to recover F from* $C^t_{\mathbb{A}}$.

The working process of the Time PRE scheme is shown in Figure 2. We will provide a system-level description for the proposed scheme as follows:

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.[4]

Data Creatio: 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, 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 cipher text to cloud servers as that in

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 *Gen Key* algorithm to generate a user identity secret key (UIK) and a set of user attribute secret keys (UAKs). [5] 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 cipher text, user *u* verifies the signatures. If correct, he accepts these secret keys. Using the UIK, user *u* can log into the system.

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 cipher text, i.e., updating the access time associated with the cipher text to the time it receives the data access request.[6] Finally, it sends the re-encrypted cipher text 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.

RESULTS AND EVALUATION

Scheme Evaluation

This research study evaluates our proposed scheme in terms of security and performance aspects.

Security Analysis

We evaluate the security of our work by analyzing the fulfillment of the security requirements.

Fine-grained Data Access Control: To provide fine-grained data access control, the proposed scheme should provide a strategy that is able to define and enforce complex access policies for sensor data of various types or security levels. In FDAC, the access structure embedded in each user's secret key is able to represent complicated predicates such as disjunctive normal form (DNF), conjunctive normal form (CNF), and threshold gates. The combination of these predicates are able to represent sophisticated access structures.

In fact, our scheme is able to support non-monotonic (general) access structures if we define the *NOT* of each attribute as a separate attribute, which in turn will double the number of attributes in our system.[7] To enforce our access control strategy, we encrypt the master key of the key chain in each stage under a set of attributes. Without the master key, the adversary is not able to derive the data encryption keys due to the one-wayness of the key chain, which can be guaranteed by choosing a secure one-way hash function such as SHA-1. In our basic scheme, the master key is actually encrypted under the standard key-policy attribute-based encryption (KP-ABE) scheme which is provably secure. Our advanced scheme, to achieve efficient user revocation, makes some enhancement to the standard KP-ABE when encrypting the master key. The enhanced KP-ABE is provably secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. (A formal security proof is available in our thesis). This turns out that the adversary is not able to decrypt the master key unless he owns the intended access structure. Therefore, our proposed scheme is able to control the disclosure of sensor data so that only authorized users are able to access.

Collusion Resilience: To compromise sensor data, the main task of the colluding users is to decrypt the master key of the target data if the one-wayness of the underlying one-way has function, e.g., SHA-1, is guaranteed. Since the master key is encrypted under our enhanced scheme, we have to prove that it is collusion-resistant. Intuitively, we can sketch the collusion-resistance of our enhanced scheme as follows. Recall that the master key is encrypted in the form of $Ke(g,g)^{ys}$. The user has to cancel $e(g,g)^{ys}$ to recover *K*. To compose $e(g,g)^{ys}$, the only way is to execute the following:

$e(g^{\frac{y-r}{\beta}}, g^{\beta s}) = e(g,g)^{ys}/e(g,g)^{rs}$. To extract $e(g, g)^{ys}$, the user should compute $e(g, g)^{rs}$. Actually, for each user, r is randomly and independently selected from Z_p. The secret key from one unauthorized user does not give the other user any help in terms of computing $e(g, g)^{rs}$. Actually, in our security proof the security definition implies collusion resistance. As our scheme is provably secure under this security definition, collusion resistance is also guaranteed.[8]

Sensor Compromise Resistance: To meet this security requirement, we should achieve two security goals:

(1) compromising the sensor node does not disclose the sensor data generated before the sensor is compromised, and

(2) compromising one sensor node does not give the adversary any advantage to obtain data generated by other sensor nodes. We can show the fulfillment of our scheme with respect to these two security goals as follows:

(1) In our scheme, each sensor node just keeps the current data encryption key in the memory, while erasing all the previously used keys. Because of the one-wayness of the key chain, the compromiser is not able to derive the previously keys from the current key.

(2) is easy to prove since each sensor node encrypts data independently.[9]

Backward Secrecy: As is described in the previous section, our advanced scheme is able to update the master key *y* for legitimate users while excluding those to be revoked. Since the new sensor data will be encrypted under the latest master key, the revoked users are not able to decrypt.[10] One problem in our scheme is, the user revocation instruction will not take effort until a new stage starts. Such a delay occurs because it would take one stage for each sensor node to encrypt the master key under the attributes. This delay may differ for different systems. For example, if a system defines 30 phases as a stage and each phase lasts 1 second, the delay will be at the most half a minute. Generally, if a system has a stage with less phases and each phase takes less time, e.g., each sensor node is assigned a smaller

number of attributes or has a more powerful computational capability, the delay can be shorter. In practical applications, we leave this delay as a system parameter, and the system designer can adjust this parameter by changing the number of attributes assigned to each sensor node or using a different type of sensor nodes.

In addition to the security goals listed above, there are also some other security requirements such as data integrity and authenticity, which are desired by conventional WSN applications. In fact, security requirements such as message integrity can be easily supported in our scheme with minor modifications using existing off-the-shelf techniques. A challenging orthogonal issue would be data authenticity which requires sensing nodes to provide proofs of data authenticity to users.[11] Some current work such as has provided salient solutions to this problem. As the main focuses of this work is fine-grained data access control, we do not explicitly address all those security problems.

Performance Evaluation

This research study evaluates the performance of our proposed scheme in terms of computation and communication overheads. In our scheme, sensor data are generated and encrypted by sensor nodes, and retrieved and decrypted by users. As sensor nodes are usually resource constrained, they may not be able to execute expensive cryptographic primitives efficiently and thus become the bottleneck of the scheme. For this reason, our evaluation focuses on the performance of sensor nodes. In the following section, we first discuss the numeric results in terms of computation and communication overheads for sensor nodes. Then, we present our implementation results on real sensors.

CONCLUSION

In this research paper, we addressed an important issue of secure data sharing on untrusted storage. We investigated the challenges pertained to this problem and proposed to exploit a novel PKC Attribute-Based Encryption (ABE) to provide cryptographically enforced data access control. With ABE, we are able to enjoy fine-grained access control. However, there are still several open security issues in state-of-the-art constructions of ABE. Toward providing a full-fledged cryptographic basis for secure data sharing on untrusted storage, we proposed three security-enhancing solutions for ABE: The first enhancement we made is to provide efficient user revocation in ABE. In this thesis, we particularly considered practical application scenarios in which semi-trustable proxy servers are available. With this assumption we uniquely combined the proxy re-encryption technique with ABE and enabled the authority to delegate most laborious tasks to proxy servers. Such a enhancement places minimal load on authority when revoking users. Our proposed scheme is provably secure against chosen cipher text attacks. In our second enhancement to ABE, we addressed key abuse attacks and proposed an abuse free KP-ABE (AFKP-ABE) scheme. (An abuse free CP-ABE (AFCP-ABE) scheme can also be built in the same way.) To defend against the key abuse attacks, we introduce hidden attributes in the system with which the tracing algorithm can identify any single pirate or partial colluding users. Our design enables black boxing tracing and does not require the well-formness of the user secret key. The complexity of AFKP-ABE in terms of cipher text size and user secret keys size is just $O(logN)$, where N is the total number of users. Our scheme is provably secure under DBDH assumption and D-Linear assumption. Our third enhancement is to provide better privacy preservation for ABE in terms of access policy information protection. In particular, we focused on CP-ABE and proposed two privacy-preserving CP-ABE schemes, in which data access structures are well protected from both the untrusted servers

and all the users even under powerful attacks, e.g., colluding attacks. Numerical and experimental results show that our scheme is suitable for large-scale applications since its overhead is just linear to the number of attributes rather than the number of users.

With ABE and our enhancement schemes, we presented our solutions for secure data sharing for two specific application scenarios - Cloud Computing and Wireless Sensor Networks. For Cloud Computing, we proposed a scheme that provides fine grained data access control for data owners in large-scale data centers outsourced to cloud. In doing so, we combined our enhanced ABE schemes with techniques such as dummy attribute and lazy re-encryption, and made it possible for both the data owner and users to delegate most computation-intensive operations to powerful cloud servers. Notably, our scheme can support resource-restrained cloud users such as mobile phones. Formal security proofs show that our proposed scheme is secure under standard cryptographic models. For Wireless Sensor Networks, we proposed a fine-grained data access control scheme for distributed storage in WSNs. In this thesis, each sensor node is assigned a set of attributes, and each user is assigned an access structure which designates the access capability of the user. Sensitive sensor data is encrypted with the attributes using ABE public keys and just allows users with the intended access structure to decrypt. In this way, confidentiality of sensitive data is protected even if the sensor node is compromised since the adversary cannot obtain the data decryption key via reading the sensor memory. As user access structure can be extremely expressive, fine-grained access control is supported. In our proposed scheme, we revised a current ABE scheme to minimize the computation load on sensor nodes in case of user revocation. We also showed that our proposed scheme is able to support attribute change of sensor nodes and seamlessly integrate existing PH schemes to realize concealed data aggregation. Our experiment shows that the system load is affordable to contemporary sensor nodes.

Future Enhancement: In the above user revocation scheme, $g^{\frac{\Delta y}{\beta}}$ is an update message common to all non-revoked users, which opens the door for a non-revoked user to collude with revoked users and help them decrypt the data. Boldyreva et al. proposed a user revocation scheme for IBE and KP-ABE in which user collusion attacks are well addressed. The proposed scheme is built on top of the construction of Fuzzy IBE and the binary tree data structure. More specifically, it

introduces a time attribute and use it in the encryption of each message. The root node of each user's access tree is an AND gate with one child being the time attribute and the other being the root node of ordinary access structure. When a user is to be revoked, the system administrator generates key updates on the time attribute using the binary tree, each leaf node of which is associated to one user. Since new messages will be encrypted with the updated time attribute, users didn't receive the key updates will not be able to decrypt. In this scheme, the complexity of encryption and decryption is comparable to that of current KP-ABE. The complexity of user revocation in terms of message size and computation overhead is $O(rlog(\frac{n}{r}))$ when $1 < r \leq n/2$, or $O(n-r)$ when $n/2 < r < n$, where r is the total number of revoked users and *n* is the total number of users. It should be noted that, we can also use this revocable KP-ABE in our scheme for achieving fine-grained data access control. One significant advantage of using the revocable KP-ABE is its enhanced security against user collusion. However, the complexity of user revocation is linear to the number of revoked users which could raise concerns in large scale systems when that number is approaching *n/2*. In our proposed user revocation solution, the

system designer is free to choose a broadcast scheme. For example, he/she can use which has the constant ciphertext size if communication overhead is of the most importance. However, security level is reduced in this solution. We treat the above issue as a trade-off between efficiency and security, and leave the choice to the system deployer.

REFERENCES

1. K. D. Edoh. Elliptic curve cryptography: Java implementation. In *Proceedings of the 1st Annual Conference on Information Security Curriculum Development*, InfoSecCD '04. ACM, 2004.
2. Google. Google Health, 2011. https://www.google.com/health.
3. B. W. John Bethencourt, Amit Sahai. Advanced crypto software collection, Feb 2011. http://acsc.cs.utexas.edu/cpabe/.
4. P. Junod and A. Karlov. An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In *Proceedings of the 10th Annual ACM Workshop on Digital Rights Management*, DRM '10, 2010.
5. D. R. Levinson. Audit of information technology security included in health information technology standards, May 2011. http: //oig.hhs.gov/oas/reports/ other/180930160.pdf
6. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology - EUROCRYPT 2010.* Springer, 2010.
7. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou. Fuzzy keyword search over encrypted data in cloud computing. In *Proceedings of the 29th IEEE Conference on Information Communications (INFOCOM'10),* San Diego, 2010.
8. M. Li, S. Yu, K. Ren, and W. Lou. Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings. In *Security and Privacy in Communication Networks.* Springer, 2010.
9. N. Liu, Y. Zhou, X. Niu, and Y. Yang. Querying encrypted character data in DAS model. In *Proceedings of the 2nd International Conference on Networking and Digital Society (ICNDS),* May 2010.
10. B. Lynn. The pairing-based cryptography library, Feb 2011. http: //crypto. Stanford, edu/pbc/.
11. K. Mandl, W. Simons, W. Crawford, and J. Abbett. Indivo: a personally controlled health record for health information exchange and communication. BMC Medical Informatics and Decision Making, 2007.