A Key-revocable Attribute-based Encryption for Mobile Cloud Environments

Tsukasa Ishiguro, Shinsaku Kiyomoto and Yutaka Miyake
KDDI R&D Laboratories Inc. 2-1-15 Ohara, Fujimino, 356-8502 Saitama, Japan

Keywords: Attribute-Based Encryption (ABE), Access Control, Cloud Computing, Key Revocation, Mobile Cloud.

Abstract: In this paper, we propose a new Attribute-Based Encryption (ABE) scheme applicable to mobile cloud environments. A key issue in mobile cloud environments is how to reduce the computational cost on mobile devices and delegate the remaining computation to cloud environments. We also consider two additional issues: an efficient key revocation mechanism for ABE based on a concept of token-controlled public key encryption, and attribute hiding encryption from a cloud server. To reduce the computational cost on the client side, we propose an efficient ABE scheme jointly with secure computing on the server side. We analyze the security of our ABE scheme and evaluate the transaction time of primitive functions implemented on an Android mobile device and a PC. The transaction time of our encryption algorithm is within 150 msec for 89-bit security and about 600 msec for 128-bit security on the mobile device. Similarly, the transaction time of the decryption algorithm is within 50 msec for 89-bit security and 200 msec for 128-bit security.

1 INTRODUCTION

1.1 Background

Commercial social network services (SNS) and document management systems have been launched in cloud environments (Google, 2012; Amazon, 2012), and outsourcing the services of an enterprise system has come to be considered a potential application for cloud computing. NIST defines cloud computing as (NIST, 2009): "*Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* " Cloud computing is becoming more common and widespread, and it provides several benefits for both cloud service providers and their users.

Here, we look at a data storage service, which is one of the most common services for cloud environments, and consider a situation where a company stores their data by utilizing the storage service. It can be assumed that company personnel need to access the data using their mobile devices. The system consists of an outsourcing database in a cloud environment and mobile devices. This situation is called a "mobile cloud environment".

Security risks associated with cloud environments are of increasing concern(Cloud Security Alliance, 2009; European Network and Information Security Agency, 2010). If the cloud service is vulnerable or the cloud provider has a malicious/curious administrator, the private information of users and corporate confidential information is at risk of being leaked. Furthermore, a fine-grained access control mechanism is needed whenever a database on a cloud service is shared by users and an access policy is defined for each role of users. Accordingly, if we consider this situation, which is one of the most common situations for a cloud environment, it can be assumed that data encryption and access control are mandatory functions.

Attribute-based encryption (ABE) schemes are an efficient solution to realize both functionalities: encryption and fine-grained access control. Sensitive user data are encrypted according to an access policy in ABE schemes, and a user cannot decrypt the data if the user does not have an access right satisfying the access policy for the data. Generally, the access policy can be described by user' attributes, e.g. affiliations and roles. The computational cost of ABE schemes is still huge in practical terms because it increases markedly as the number of attributes described in the access policy increases. We focus on the mobile cloud environment and consider accesses from mobile devices. In the mobile cloud environment, the computational cost on mobile devices becomes problematic in practice. Another issue confronting ABE is key revocation. The system needs to manage the revocation of users since mobile devices are easily lost or stolen.

1.2 Related Works

Attribute-based encryption (ABE) has been extensively researched as a cryptographic protocol (Sahai and Waters, 2005; Bethencourt et al., 2007; Hinek et al., 2008; Lewko et al., 2010). In Ciphertext-Policy ABE (CP-ABE) systems (Bethencourt et al., 2007), a user encrypts data with descriptions of an access policy. The access policy defines authorized users, and their statements consisting of attributes and logical relationships such as **AND**, **OR**, or **M of N** (threshold gates); for example, users who have the attributes "*Project manager*" and "*Administration department*" can access the data, of which the access policy is defined as "*Project manager* $\wedge$ *Administration department*". The mechanism can prevent a cloud service provider or an adversary from accessing the secret information. Another type of ABE is a Key-Policy ABE (KP-ABE) (Goyal et al., 2006). In KP-ABE, a user's personal key corresponds to a combination of attributes "*Project manager* $\wedge$ *Administration department*".

Generally, ABE schemes require huge computations such as many pairing computations. Several papers have been published that deal with the implementation of pairing computations on different devices (Beuchat et al., 2010; Scott, 2011; Aranha et al., 2010; Naehrig et al., 2010). As shown in these papers, one pairing computation can be completed in less than a few milliseconds on a current PC. However, more computation time is required on other devices that have less computational power, such as smartphones. Furthermore, the computational power of the ABE mechanism increases appropriately as the number of attributes increases.

Some papers have proposed schemes to remove heavy computations from a mobile device (Zhou and Huang, 2011; Green et al., 2011) and delegate them to cloud servers at a remote site. In (Zhou and Huang, 2011), Zhou *et al.* divided the encryption step into two parts, a mobile device part and a cloud server part. For the first part, the mobile device encrypts data in a constant time; the transaction time of the first encryption function does not depend on the number of attributes. For the second part, the cloud server can operate the second encryption function without plaintext information; no information is revealed to the cloud server. In the same way, they also divide the decryption step into two parts. The Zhou *et al.* scheme has no key revocation mechanism; thus, this is an open issue for ABE in mobile cloud environments. Furthermore, they did not present sufficient security analyses for their ABE scheme.

Another solution for mobile cloud environments is outsourcing decryption processes (Green et al., 2011; Goyal et al., 2006; Waters, 2011) using a proxy server. A user has a translation key TK along with a secret key. The user sends the TK to a proxy server, for the decryption process on the proxy server. An ElGamal-style ciphertext for the decryption process is used to avoid revealing any part of a message. The decryption step is similar to ElGamal decryption, and the computational cost of decryption imposed on a mobile device can be reduced by delegating some of the processes of decryption to the proxy server, even though the cost for encryption is still heavy.

From the viewpoint of key management, a revocation mechanism is also required. However, no key revocation mechanisms can be found in the literature for the Zhou *et al.* scheme(Zhou and Huang, 2011). Some papers have proposed schemes to revoke an unauthorized user (Baek et al., 2005; Galindo and Herranz, 2006; Sadeghi et al., 2010). Token-controlled public key encryption (TCPKE) is a public key encryption scheme in which an entity cannot decrypt an encrypted message until the entity obtains additional information called a "token". Beak *et al.* introduced this concept and a concrete algorithm in (Baek et al., 2005). Galindo *et al.* formalized a natural security goal for TCPKEs and proposed an improved TCPKE. Some TCPKE schemes (Galindo and Herranz, 2006; Sadeghi et al., 2010) exist, but no schemes directly applicable to mobile cloud environments have been fully discussed.

1.3 Our Contributions

In this paper, we propose a key-revocable ABE scheme for a mobile cloud environment, which realize the following functionalities:

-
- *User Revocation*. Our scheme is based on the concept of the ABE scheme proposed by Zhou *et al.* and achieves efficient key revocation by introducing a new entity, the Token Service Provider (TSP). The TSP distributes a "token" to a valid user when the user requests decryption of a ciphertext. However, when the user is unauthorized, the TSP does not return a valid token, which revokes the user.

- *Attribute Hiding*. We realize attribute information hiding from a cloud server. The user terminal executes a few computations in order to mask attribute information.

Furthermore, we optimize the ABE scheme and show that the scheme is applicable to a current mobile cloud environment. We evaluate transaction time using an Android mobile device and a PC, and discuss the feasibility of our scheme. Security analyses based on four security requirements are presented in this paper.

2 PRELIMINARY

In this section, we define some notations used in our scheme.

Definition 1 (Bilinear Pairing e)**.** *Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let $g \in \mathbb{G}_0$ be the generator of $\mathbb{G}_0$ and e be a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The bilinear map e has the following properties:*

- *Bilinearity:* $\forall u, v \in \mathbb{G}_0, a, b \in \mathbb{Z}_p, e(u^a, v^b) = e(u, v)^{ab}D$
- *Non degeneracy:* $\forall u \in \mathbb{G}_0, e(u, u) \neq 1$

Definition 2 (Monotone Access Structure)**.** *Let $\{P_1, \cdots, P_n\}$ be a set of partiesD A collection $\mathbb{A} = 2^{\{P_1,\cdots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}D$ An access structure is a collection $\mathbb{A}$ of non-empty subsets of $\{P_1, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\cdots,P_n\}} \setminus \{\phi\}$. Sets contained in $\mathbb{A}$ are called authorized sets and the sets not in $\mathbb{A}$ are called unauthorized sets.*

Definition 3 (Access Tree)**.** *Let $\mathcal{T}$ be an access tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, which defines a required condition in the child nodes. If a node x is a non-leaf node of the tree and has num_x children and k_x is its threshold value, then $0 < k_x \leq num_x$. When $k_x = 1$, it is an **OR** gate and when $k_x = num_x$, it is an **AND** gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. This access structure satisfies the conditions for a monotone access structure.*

In this paper, we use functions to describe an access tree. Child nodes of every node are numbered from 1 to *num* in the access tree. The function $index(x)$ returns the number of the node x, and $att(x)$ denotes the attribute associated with the leaf node x in the tree. The function $f(S, W)$ returns 1 if and only if a set of attributes S satisfies an access policy W, otherwise it returns 0. We denote the number of attributes in a tree $\mathcal{T}$ by $|\mathcal{T}|$.

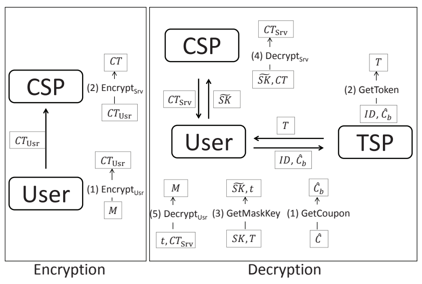

Figure 1: Outline of Encryption and Decryption Steps.

We assume that the following four problems(CDH, DDH, BDH, DBDH) are hard in our scheme:

Definition 4 (Computational Diffie-Hellman Problem(CDH)(Boneh, 1998))**.** *$\mathbb{G}$ denotes a group of a prime order p. Let choose $a, b \in \mathbb{Z}_p$ at random, and given g, g^a, g^b. The CDH is to compute g^{ab}.*

Definition 5 (Decisional Diffie-Hellman Problem(DDH)(Boneh, 1998))**.** *$\mathbb{G}$ denotes a group of a prime order p. Let choose $a, b, c \in \mathbb{Z}_p$ at random, and given g^a, g^b, g^{ab}, and g^c. The DDH is to determine whether $(g^a, g^b, g^{ab}) = (g^a, g^b, g^c)$.*

Definition 6 (Bilinear Diffie-Hellman Problem(BDH)(Boneh and Franklin, 2001; Joux, 2004))**.** *Given g, g^a, g^b, g^c for some $a, b, c \in \mathbb{Z}_p$, the BDH is to compute $e(g, g)^{abc}$.*

Definition 7 (Decisional Bilinear Diffie-Hellman Problem(DBDH)(Boneh and Franklin, 2001))**.** *Let $\mathbb{G}_0$ and $\mathbb{G}_1$ denotes a group of a prime order p. Let choose $a, b, c \in \mathbb{Z}_p$ at random and $g_1 \in \mathbb{G}_0, g_2 \in \mathbb{G}_1$, and given g_i^a, g_j^b, g_k^c, and $e(g_1, g_2)^z$, where $\{i, j, k\} \in \{(1,1,1), (1,1,2), (1,2,2), (2,2,2)\}$. The DBDH is to determine whethere$(g_1, g_2)^{abc} = e(g_1, g_2)^z$.*

3 OUR SCHEME

In this section, we explain our key-revocable ABE scheme for a mobile cloud environment.

3.1 Definition

Our scheme consists of nine algorithms **Setup**, **Key-Generation**, **Encrypt**$_{Usr}$, **Encrypt**$_{Srv}$, **Decrypt**$_{Usr}$, **Decrypt**$_{Srv}$, **GetCoupon**, **GetToken**, **GetMask Key**. There are four entities in our scheme: User, Trusted Key Generator (TKG), Cloud Service Provider (CSP), and Token Service Provider (TSP). The TSP is adopted to realize a key revocation mechanism. An overview of the encryption and decryption steps in our scheme is shown in Figure 1.

Before the encryption and decryption steps, the TKG generates a public key and users' private keys by running two algorithms, **Setup** and **KeyGeneration**, and the TKG securely distributes the keys to users. The TKG also sends *ID*s and related information such as a terminal-ID to the TSP.

In the encryption step, the user first decides on a ciphertext policy and runs **Encrypt**$_{Usr}$ to compute a temporal ciphertext, then the user sends the temporal ciphertext and the ciphertext policy to the CSP. As the second encryption step, the CSP runs **Encrypt**$_{Srv}$ to calculate a complete ABE ciphertext.

In the decryption step, the user runs **GetMaskKey**, and sends a query to the TSP to obtain a token. The TSP checks whether the user is not revoked, and returns a token to the user by running **GetToken**. Note that the **GetCoupon** is assumed to be securely computed on the TSP and the user cannot compute an invalid ciphertext coupon to use an altered terminal-ID. Thus, the validity of the terminal-ID is ensured in our scheme. Next, the user requests the CSP to run **Decrypt**$_{Srv}$ and the user receives a partial ciphertext from the CSP, provided the user has attributes that satisfy the ciphertext policy. Finally, the user runs **Decrypt**$_{Usr}$ to obtain a plaintext from the partial ciphertext.

The nine algorithms are denoted as follows;

- **Setup(**λ, n**)** $\to (PK, MK)$.
  The algorithm receives a security parameter λ and the number of possible attributes n as an input, and it outputs two keys: a system public key PK and a master key MK.
- **KeyGeneration(**MK, PK, S_u, b_u**)** $\to$ **(**ID_u, SK_u, $\widehat{D}_u$**)**.
  Let S be a group of all possible attributes in the system. A master key MK, a public key PK, the attributes $S_u \in S$ of the user u, and a terminal-ID b_u of the user are input to the algorithm. The algorithm outputs a user's ID ID_u, a secret key SK_u of the user and token data $\widehat{D}_u$.
- **Encrypt**$_{Usr}$**(**PK, M, W**)** $\to CT_{Usr}$.
  The algorithm takes a public key PK, a plaintext message M, and a decryption policy W as input, and outputs a temporal ciphertext CT_{Usr}.
- **Encrypt**$_{Srv}$**(**PK, W, CT_{Usr}**)** $\to CT$.
  From input of data, a public key PK, a decryption policy W and a user-side ciphertext CT_{Usr}, the algorithm outputs a complete ABE ciphertext CT.
- **GetCoupon(**$\widehat{C}$**)** $\to \widehat{C}_b$.
  From input a $\widehat{C}$, the algorithm returns a ciphertext coupon $\widehat{C}_b$.
- **GetToken(**$ID_u, \widehat{C}_b$**)**$\to T$ **or** $\perp$.
  A user ID ID_u and a ciphertext coupon $\widehat{C}_b$ are input to the algorithm, and the algorithm outputs a token T if the user's ID ID_u is not revoked, otherwise the algorithm outputs $\perp$.
- **GetMaskKey(**SK_u, T, b_u**)** $\to (\widetilde{SK}_u, t_u)$.
  The algorithm computes a masked secret key $\widetilde{SK}_u$ and a masked value t_u from input of a secret key SK_u and a token T.
- **Decrypt**$_{Srv}$**(**$\widetilde{SK}_u, CT$**)** $\to CT_{Srv}$.
  The algorithm receives a masked secret key $\widetilde{SK}_u$ and a ciphertext CT, and outputs a partial ciphertext CT_{Srv}.
- **Decrypt**$_{Usr}$**(**CT_{Srv}, t_u, T**)** $\to M$.
  From input of a masked value t_u and a partial ciphertext CT_{Srv} and a token T, the algorithm outputs a plaintext M.

3.2 Security Requirements

Our scheme must satisfy the following three requirements similar to existing schemes:

- *Plaintext Confidentiality*. No information about plaintext is leaked to any adversary including a malicious operator for a cloud service provider. Even if all cloud servers collude after correct key distribution, they are not able to obtain information about plaintext.
- *Unclonability*. The scheme must satisfy unclone-ability of private keys in order to prevent unauthorized use of the private keys. When a malicious user copies her private key and provides the copy to another user, the private information of the malicious user is leaked. It implies that private key cloning by a malicious user is suppressed due to the violation of the user's privacy. Thus, a malicious user would not clone the private key.
- *Privacy Preserving*. A terminal-ID b that is a unique identifier of a user should be protected, where a temporal identifier of the user ID_u is leaked in a cloud service.

Furthermore, the scheme satisfies an additional requirement as follows:

- *Attribute Hiding*. Attribute information of a user cannot be obtained from transaction data at a reasonable cost, even if other users are compromised.

In this section, we formalize the above security requirements.

3.2.1 Plaintext Confidentiality

We define the security game for *Plaintext Confidentiality* as follows;

System Setup: The challenger runs the **Setup** algorithm and gives the public parameters, *PK* to the adversary.

Challenge: The challenger initializes empty tables V and L, and an empty set D. The adversary submits two equal length messages M_0 and M_1. In addition the adversary gives a value W^*_{enc}. The challenger flips a random coin c, and encrypts M_c under W^*_{enc}. The challenger generates ID_0 and stores the entry $(0, r, r_0, S_0, \widetilde{SK}_0, t_0)$ in the table V. The resulting ciphertext CT^* and CT^*_{Srv}, ID_0 and $\widetilde{SK}_0$ are given to the adversary.

Game: The challenger sets an integer $j = 0$. The game is repeated with the restrictions that the adversary cannot

- trivially obtain a private key for the challenge ciphertext. That is, it cannot issue a corrupt query that would result in a value S_0 that satisfies $f(S_0, W^*_{enc}) = 1$ being added to D.
- issue a trivial decryption queries. That is, Decrypt$_{Usr}$ queries will be answered, except in the case where the response would be either M_0 or M_1, then the challenger responds with the special message **test** instead.

Proceeding adaptively, the adversary can repeatedly make any of the following queries:

- Create(PK, S_j): The challenger sets $j := j+1$, and $b_j = j$. It runs the **KeyGeneration** algorithm to obtain SK_j,ID_j, and $\widehat{D_j}$, and then it stores a tuple of j, ID_j, and $\widehat{D_j}$ in the table L. Next, it runs the **GetMaskKey** algorithm to obtain the pair $(SK_j, \widetilde{SK}_j, t_j)$ corresponding to attributes S_j of a user j, and stores the entry $(j, r, r_j, S_j, SK_j, \widetilde{SK}_j, t_j)$ in the table V. Finally, it returns to the adversary the value j, the masked secret key $\widetilde{SK_j}$, and the ID ID_j.
- Corrupt(i): If there exists an i-th entry in table V and $i \neq 0$, then the challenger obtains the entry $(i, r, r_i, S_i, SK_i, \widetilde{SK}_i, t_i)$. If $S_i \neq S_0$, then the challenger sets $D := D \cup \{S_i\}$ and returns the entry to the adversary including the private key SK_i. If no such entry exists or $S_i = S_0$, then it returns $\perp$.
- GetToken(i, $\widehat{C}$): Note that $\widehat{C}$ is included in CT. If i exists in the table L, then the challenger obtains the entry (i, ID_i, $\widehat{D_i}$) and returns the token T_i and ID_i to the adversary. If no such entry exists, then it returns $\perp$.
- Encrypt(i, PK, W_{enc}, M): The challenger runs the algorithms **Encrypt$_{Usr}$** and **Encrypt$_{Srv}$**. The challenger then returns CT, and CT_{Usr}.
- Decrypt$_{Srv}$($\widetilde{SK_i}, CT$): The challenger returns the output CT_{Srv} of the decryption algorithm **Decrypt$_{Srv}$** on input $(\widetilde{SK}_i, CT)$ to the adversary.
- Decrypt$_{Usr}$(i, CT_{Srv}, T_i): If there exists an i-th entry in the table V, then the challenger obtains the entry $(i, r, r_i, S_i, SK_i, \widetilde{SK}_i, t_i)$, and it returns to the adversary the output M of the decryption algorithm **Decrypt$_{Usr}$** on input (SK, T_i, t_i, CT). If no such entry exists, then it returns $\perp$.

Guess: The adversary outputs a guess c_0 of c.

We should consider a security model for ABE that supports two-stage encryption and decryption on a client terminal and a cloud server. The general notion of security against chosen-ciphertext attacks (CCA) is somewhat too strong and it cannot be directly applied to the mobile cloud environment. We thus use a relaxed security notion called *replayable* CCA security (Canetti et al., 2003), which allows modifications to the ciphertext but does not allow any changes to the underlying message in a meaningful way. The scheme achieves *Plaintext confidentiality*, where the scheme satisfies the following definition.

Definition 8 (RCCA-secure(Green et al., 2011))**.** *The ABE scheme is RCCA-secure (or secure against replayable chosen-ciphertext attacks) under the selective-ID model, if all polynomial time adversaries have at most a negligible advantage in the game defined above.*

3.2.2 Uncloneability

The scheme can be achieved *Unclonability* where the scheme satisfies the following definition.

Definition 9 (Strong Uncloneability(Hinek et al., 2008))**.** *A polynomial-time algorithm $\mathcal{F}$ exists and it executes the GetToken oracle, the Decrypt$_{Srv}$ oracle, and the following functions:*

$$\mathcal{F}^{\{GetToken, \mathrm{Decrypt}_{Srv}\}}(PK, \widehat{C}, i, \widehat{D}_i) =$$
$$\mathrm{Decrypt}_{Usr}(b_i, \mathrm{Decrypt}_{Srv}(i, CT), t_i, \mathrm{GetToken}(i, \widehat{C}))$$

If an extractor algorithm χ can compute a user identifier ID and the terminal-ID b from a given $\widetilde{SK}_i, CT, ID_i, t_i, \widehat{C}$, and T by $O(|B|)$ computation, it is said that the ABE scheme satisfies Strong Uncloneability. Note that $|B|$ is the domain of terminal-IDs of all users.

3.2.3 Privacy Preserving

The scheme can achieve *Privacy Preserving* where the scheme satisfies the following definition.

Definition 10 (Privacy Preserving(Hinek et al., 2008))**.** *Let MK, PK, T, S_u be a master key, a public key, a token, and all attributes of a user u. If*

$$\Pr[b_u|PK,MK,ID_u,S_u,T] = \Pr[b_u|ID_u,S_u],$$

then the ABE scheme satisfies Privacy Preserving.

3.2.4 Attribute Hiding

The objective of an adversary $\mathcal{A}$ is to guess attribute values embedded in a ciphertext. For the sake of simplicity, we consider the case where an access policy of the ciphertext includes an attribute, without limiting the generality of the security requirement. The scheme can achieve *attribute hiding* where the scheme satisfies the following definition. This definition is similar to a security definition of password-based key-exchange protocols (Bellare et al., 2000). That is, the adversary cannot engage in off-line searching of the attribute using transaction data.

Definition 11 (Attribute Hiding)**.** *Let q_D be the number of decrypt queries. Let the size of S_u be N. If the success probability of attribute guessing by an adversary $\mathcal{A}$ is bounded by q_D/N, then the ABE scheme satisfies attribute hiding.*

3.3 Construction

In this section, we explain our scheme in detail.

Setup(λ) $\rightarrow (PK,MK)$**.** Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be bilinear groups of prime order p (where p is a λ bit prime), $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ be a bilinear map, and g be a generator of $\mathbb{G}_0$. With randomly chosen $a \in \mathbb{Z}_p$, the system public key PK and the system master key MK are given by, $PK = (g, e(g,g)^a, e, \mathbb{G}_0, \mathbb{G}_1), MK = g^a$

KeyGeneration(MK,PK,S_u,b_u) $\rightarrow$ (ID_u, SK_u, $\widehat{D}_u$) Let $H : \{0,1\}^* \rightarrow \mathbb{G}_0$ be the collision-resistant hash function. A random number $r \in \mathbb{Z}_p$ and $r_j \in \mathbb{Z}_p$ are assigned to each attribute $j \in S_u$. The decryption key for the user-ID ID_u is given by,

$$SK_u = (\forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j})$$

The TKG computes $\widehat{D}_u = g^{\frac{(a+r)}{b_u}}$ and sends $(ID_u, \widehat{D}_u)$ to the TSP.

Encrypt$_{Usr}$($PK,M,\mathcal{T}_{\mathrm{Usr}}$) $\rightarrow CT_{Usr}$**.** A polynomial $q_R \in \mathbb{Z}_p[X], \deg(q_R) = 1$ is randomly chosen, and $s = q_R(0), s_1 = q_R(1), s_2 = q_R(2)$ are calculated. Let Y be set of leaf nodes of a $\mathcal{T}$ with a "DO" node. The user calculates CT_{Usr} as follows;

$$\begin{aligned} CT_{Usr} \quad = \quad & (s_1, \mathcal{T}_{\mathrm{U}}, \widehat{C} = g^s, \widetilde{C} = M \cdot e(g,g)^{as}; \\ & \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}) \end{aligned}$$

Encrypt$_{Srv}$($PK,\mathcal{T},CT_{Usr}$) $\rightarrow CT$**.** Polynomials $q_x \in \mathbb{Z}_p[X]$ for all nodes in $\mathcal{T}$ without a root node and a "DO" node are randomly chosen. The polynomials are chosen in a top-down manner from the start node θ which is the highest node without a root node and a "DO" node. For each node x under θ, the degree d_x of each polynomial q_x is assigned a value less than the threshold value k_x; for example, $d_x = k_x - 1$. For the start node θ, a polynomial is selected to satisfy $q_\theta(0) = s_1$. Then, for each node under θ, a polynomial is selected to satisfy $q_x(0) = q_{parent(x)}(index(x))$ using randomly chosen coefficients without constant terms.

Let Y_{U} be set of leaf nodes of a $\mathcal{T}$ without a "DO" node. CT_{Srv} is calculated as follows:

$$CT_{Srv} = (\forall y \in Y_{\mathrm{U}} : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)})$$

For Y that is the set of all leaf nodes of $\mathcal{T}$, CT is calculated as follows:

$$CT \quad = \quad (\mathcal{T}_{\mathrm{Srv}}, \widetilde{C}, \widehat{C}; \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)})$$

GetMaskKey(SK_u,T,b_u) $\rightarrow (\widetilde{SK}_u, t_u, \widehat{C}_b)$**.** $t_u \in \mathbb{Z}_p$ is randomly chosen and $\widetilde{SK}_u$ is calculated as follows;

$$\widetilde{SK}_u = (\forall j \in S_u : D_j = (g^r \cdot H(j)^{r_j})^{t_u}, D'_j = (g^{r_j})^{t_u})$$

GetCoupon($\widehat{C}$) $\rightarrow \widehat{C}_b$
For an input C, a ciphertext coupon $\widehat{C}_b$ is computed thus: $\widehat{C}_b = \widehat{C}^b$

GetToken($ID_u, \widehat{C}_b$) $\rightarrow T$ **or** $\perp$**.** The token server outputs a token $T = e(\widehat{C}_b, \widehat{D}_u) = e(g^{b_u s}, g^{\frac{a+r}{b}}) = e(g,g)^{s(a+r)}$ where the user-ID ID_u is not revoked. If not, the server outputs $\perp$. In our scheme, key revocation revokes ID_u and removes $\widehat{D}_u$ from the data stored in the TSP.

Decrypt$_{Srv}$($\widetilde{SK}_u, CT'$) $\rightarrow CT_{Srv}$**.** CT' is described as follows:

$$\begin{aligned} & CT' = (\mathcal{T}, \widetilde{C} = M \cdot e(g,g)^{as}, \\ & \forall y \in Y_{\mathrm{U}} \cup Y_{\mathrm{DO}} : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}) \end{aligned}$$

Let $\rho(\ell)$ be $(\ell, \mathrm{parent}(\ell), \mathrm{parent}(\mathrm{parent}(\ell)), \cdots, R)$ for all $\ell \in \mathcal{T}$. z_ℓ is calculated as follows:

$$\begin{aligned} z_\ell \quad & = \prod_{x \in \rho(\ell), x \neq R} \Delta_{i,S}(0), \text{where } i = \mathrm{index}(x), \\ & = \prod_{x \in \rho(\ell), x \neq R} \prod_{j \in S_u, j \neq i} \frac{-j}{i-j} \end{aligned}$$

Finally, A is calculated as:

$$\begin{aligned} A &= \prod_{\ell \in L, i=\mathrm{att}(\ell)} \left(\frac{e(D_i, C_\ell)}{e(D'_i, C'_\ell)} \right)^{z_\ell} \\ &= e(g,g)^{rst_u}, \end{aligned}$$

and $CT_{Srv} = (A, \widetilde{C})$ is sent to the user.

Decrypt$_{Usr}$**(**CT_{Srv}, t_u, T**)** $\rightarrow M$**.** M is calculated as follows:

$$\begin{aligned} \frac{\widetilde{C} A^{\frac{1}{t_u}}}{T} &= \frac{Me(g,g)^{as} \cdot e(g,g)^{rs}}{(e(g,g)^{s(a+r)})} \\ &= \frac{Me(g,g)^{(a+r)s}}{e(g,g)^{(a+r)s}} \\ &= M \end{aligned}$$

4 ANALYSIS

In this section, we analyze the security and performance of the proposed scheme.

4.1 Security Analysis

In this subsection, we present proofs of scheme security with regard to three security requirements: Plaintext Confidentiality, Strong Uncloneability, and Privacy Preserving.

4.1.1 Plaintext Confidentiality

An objective of the adversary is to output a correct value of c from a given ciphertext and related transaction data. From the adversary's viewpoint, challenge plaintext M_c is embedded in $\widetilde{C}$ of CT^*_{Srv}. It is hard to remove t_0 from A_0 of CT^*_{Srv}, if the BDH problem holds. Thus, the adversary cannot obtain M_c from the ciphertext CT^*_{Srv} to execute the Decrypt$_{Usr}$ algorithm in the game. The adversary cannot obtain s and g^a with feasible computational costs in the game, and it is difficult for the adversary to compute the value of $e(g,g)^{as}$. Thus, for the adversary, two ciphertexts $M_0 e(g,g)^{as}$ and $M_1 e(g,g)^{as}$ are indistinguishable with non-negligible success probability. First, we prove that our scheme has plaintext confidentiality where the decisional BDH assumption holds.

Theorem 1. *Our scheme has plaintext confidentiality under the decisional BDH assumption.*

Proof 1. *(Sketch) Suppose that we have an adversary $\mathcal{A}$ with a non-negligible advantage ε in the selective-ID game against our construction. We build a challenger $\mathcal{B}$ that plays the decisional BDH problem. First, the challenger takes on the BDH challenge $g, g^\alpha, g^\beta, g^\gamma, Y_w$. The challenger outputs $w = 0$ where Y_w is guessed as $Y_w = (g,g)^{\alpha\beta}$; otherwise outputs $w = 1$. As the system setup, the challenger runs the setup algorithm and gives PK to the adversary. The adversary submits two messages M_0 and M_1 as a challenge. The challenger flips a random coin c and make CT^* and CT^*_{Srv} including $\widehat{C}$, $\widetilde{C}$, and A: $\widehat{C} = g^{\beta r'}$, $\widetilde{C} = M_c \cdot e(g^{\beta r'}, g^a)$, and $A = (Y_w)^{rr'}$, where r' is a randomly chosen from $\mathbb{Z}_p$. The two ciphertexts CT^* and CT^*_{Srv}, ID_0, and $\widetilde{SK}_0$ are given to the adversary. The challenger stores the tuple (0, r', M_c, CT^*, CT^*_{Srv}) in the table E. In the game, the challenger $\mathcal{B}$ runs the adversary $\mathcal{A}$ at described in 3.2. For the KeyGeneration queries, the challenger computes $D_j = g^{\alpha r} \cdot H(j)^{r_j}$ and $\widehat{D} = (g^{\alpha r} \cdot g^a)^{1/b}$ for a randomly chosen r and r_j. If the Decrypt$_{Srv}$($\widetilde{SK}_0, CT$) is received, the challenger obtains the entry of $i = 0$ and decrypts CT to obtain M. The challenger then returns $CT_{Srv} = ((Y_w)^{rr'}, Me(g, g^{\alpha r})^a)$ and the tuple (0, r', M, CT, CT_{Srv}) to the table E. The challenger returns CT^*_{Srv} stored in the table E to the adversary, when the adversary sends the Decrypt$_{Srv}$($\widetilde{SK}_0, CT^*$) to the challenger. Note that the adversary cannot distinguish whether $A = (Y_w)^{rr'}$ or $A = e(g,g)^{rst}$ except in the case where the adversary has obtained t_0. For the Decrypt$_{Usr}$($0, CT_{Srv}, T_0$), the challenger obtains M from the table E and returns M, if the entry (0, r', M, CT, CT_{Srv}) exists in the table E; otherwise it returns $\perp$. Finally, the adversary outputs c_0 at the guess stage. The challenger outputs $w = 0$ where $c_0 = c$; otherwise outputs $w = 1$. The adversary outputs a correct bit $c_0 = c$ with the non-negligible probability ε where $Y_w = (g,g)^{\alpha\beta\gamma}$; otherwise randomly outputs the correct bit with the probability $1/2$. The challenger $\mathcal{B}$ can play the decisional BDH game with non-negligible advantage. This result contradicts the assumption. Thus, the ABE scheme has plaintext confidentiality under the decisional BDH assumption.*

Note on Collusion. The data structures of ciphertext and a private key in our scheme are the same as those of tk-CP-ABE proposed by Hinek *et al.* (Hinek et al., 2008). Our scheme can be viewed as a variant of tk-CP-ABE. In our scheme, the access policy consists of two sub-trees $\mathcal{T}_S$ and $\mathcal{T}_{DO}$. $\mathcal{T}_{DO}$ contains only a single attribute DO to reduce the computational cost. A user randomly selects a 1-degree polynomial q_x and sets $s = q_x(0), s_1 = q_x(1), s_2 = q_x(2)$. Then, the user sends $\{s_1, \mathcal{T}\}$ to CSP. CSP cannot know q_x without s and s_2. The servers get $e(g,g)^{\alpha s_1}$ easily since $e(g,g)^\alpha$ is a public parameter. A decryption server knows $e(g,g)^{r'st}$, $e(g,g)^{r's_1}$, $e(g,g)^{r's_2}$, and $e(g,g)^{r's}$ through the Decrypt$_{Srv}$ function. An encryption server has the values s_1 and $e(g,g)^{\alpha s_1}, s_1$, but the values of s_2 and s are in unknown. The decryption server obtains all blinded private keys as well as the blinded private key $\widetilde{SK}_{u'}$ of user u'. Note that $\widetilde{SK}_{u'}$ is not a valid private key since the $\widetilde{D}_u$ is embedded with a random parameter t_u. Since t_u is the exponent of the generator g, the derivation of t_u is equivalent for solving the DLP problem that is considered to be hard. Thus, the hardness of DLP on $\mathbb{G}_0$ and $\mathbb{G}_1$ are given, and cloud servers cannot derive $e(g,g)^{\alpha s_2}$ or $e(g,g)^{\alpha s}$ even if they collude.

4.1.2 Uncloneability

If a private key is easily copied from an inner user to an adversary, the adversary can access the user's data. Cloning a private key should be prevented. We prove our scheme has Strong Uncloneability (Hinek et al., 2008).

Theorem 2. *Our scheme has Strong Uncloneability.*

Proof 2. *(Sketch) First, an extractor χ computes the plaintext M using the algorithm $\mathcal{F}$. Since the extractor χ must access GetToken, it can request $T =$ GetToken$(i, \widehat{C})$, and χ computes $A = e(g,g)^{rst}$ from PK, CT, SK'. Since $M = \frac{\widetilde{C}A^{\frac{1}{t_i}}}{T}$, χ tries to check all the values of $b_i \in B$ until the above equation is satisfied. Thus, χ computes b_i within time complexity $(|B|)$ in the worst case.*

4.1.3 Privacy Preserving

Personal information should remain private even if a key generator is corrupted and colludes with a token server. We prove that our scheme preserves privacy (Hinek et al., 2008).

Theorem 3. *Our scheme is privacy preserving.*

Proof 3. *(Sketch) Since PK, MK are chosen independently of b_u,*

$$\Pr[b_u|PK, MK, ID_u, S_u, \hat{C}] = \Pr[b_u|ID_u, S_u, \hat{C}]$$

holds. Next, recall that $\hat{D}_u = g^{\frac{(a+r)}{b_u}}$ and the order of G is q. It follows that

$$\Pr[b_u|ID_u, S_u, \hat{D}_u] \quad = \quad \Pr[b_u|ID_u, S_u, \frac{(a+r)}{b_u} \bmod p].$$

Since a, r, b are independently chosen,

$$\Pr[b_u|ID_u, S_u, \frac{(a+r)}{b_u} \bmod p] = \Pr[b_u|ID_u, S_u]$$

holds. Thus, our scheme is privacy preserving.

4.1.4 Attribute Hiding

We show that our scheme satisfies *Attribute Hiding*.

Theorem 4. *Our scheme satisfies Attribute Hiding.*

Proof 4. *(Sketch) Assume for simplicity there is just one attribute j_a that is needed to decrypt a ciphertext CT. The value $C'_y = H(att(y))^{q_y(0)}$ is denoted as g^x. Thus, it is difficult to distinguish the attribute j_a using C_y and C'_y where the decisional Diffie-Hellman (DDH) problem is hard. We view an adversary $\mathcal{A}$ as trying to guess the attribute using queries defined in 3.2, and consider a set of remaining candidate attributes in the game. The size of the set of remaining attributes decreases by at most one with each oracle query. The adversary decreases the candidate attributes to use the Decrypt query as follows. First, the adversary obtains a pair of $CT_{Srv} = (A, \widetilde{C})$ for the challenge ciphertext CT to use the Decrypt$_{Srv}$ query. Next, the adversary sets a guessed attribute j'_a to $\widetilde{SK}$ and sends a Decrypt$_{Srv}$ query to an entity. If the entity outputs the correct pair of $CT_{Srv} = (A, \widetilde{C})$, then $j'_a = j_a$; otherwise the adversary removes j'_a from the candidate attributes. The success probability of the adversary is q_D/N where the number of queries is bounded by q_D.*

In addition to proofs for the security requirements, we consider the feasibility of the user revocation process.

4.1.5 User Revocation

Our proposal scheme realizes a user revocation. A system administrator can revoke malicious users by removing user's entry(i.e. $(ID_u, \widehat{D}_u)$) from the token server's database. $\widehat{D}_u$ is a power of generator g, and the exponent of $\widehat{D}_u$ is randomized. Therefore, the user can not recover the entry because of the discrete logarithm problem(DLP). Since a user needs a decryption token in order to decrypt any ciphertext (using a private key of the user), the user can not obtain any ciphertext after the revocation of the user.

4.2 Analysis of Communication Cost

In this section, we analyze communication cost of our scheme. In the Table 1, we show the parameter sizes. The parameter h represents the maximum number of attributes. The symbol $|\mathcal{T}_h|$ denotes the size of a tree using h attributes (nearly $h|\mathbb{Z}_p|$). The largest part of the communication cost is the transaction that involves sending ciphertext CT_{Usr} from the user to the server. The size of CT_{Usr} approaches asymptotically estimated as $O(h^2|\mathbb{Z}_p|)$. The communication cost is proportional to h^2 similar to that of existing schemes.

Table 2: Computational cost of our scheme.

	Hash	Mul $\mathbb{G}_0$	Exp $\mathbb{G}_0$	Mul $\mathbb{G}_1$	Inv $\mathbb{G}_1$	Exp $\mathbb{G}_1$	Pairing
Setup	-	-	1	-	-	1	1
KeyGeneration	$\lvert S_u\rvert$	$\lvert S_u\rvert$	$2\lvert S_u\rvert+2$	-	-	-	-
Encrypt$_{Usr}$	1	-	4	1	-	1	-
Encrypt$_{Srv}$	$\lvert\mathcal{T}\rvert$	-	$2\lvert\mathcal{T}\rvert$	-	-	-	-
Decrypt$_{Usr}$	-	-	-	2	1	1	-
Decrypt$_{Srv}$	-	-	-	$2\lvert\mathcal{T}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{T}\rvert$	$2\lvert\mathcal{T}\rvert+1$
GetToken	-	-	-	-	-	-	1
GetCoupon	-	-	1	-	-	-	-
GetMaskKey	-	-	$\lvert\mathcal{T}\rvert$	-	-	-	-

Table 3: Computational Cost of Zhou's scheme.

Setup	-	-	3	-	-	1	1
	Hash	Mul $\mathbb{G}_0$	Exp $\mathbb{G}_0$	Mul $\mathbb{G}_1$	Inv $\mathbb{G}_1$	Exp $\mathbb{G}_1$	Pairing
KeyGeneration	$\lvert S_u\rvert$	$\lvert S_u\rvert$	$2\lvert S_u\rvert+1$	-	-	-	-
Encrypt$_{Usr}$	1	-	3	1	-	1	-
Encrypt$_{Srv}$	$\lvert\mathcal{T}\rvert$	-	$2\lvert\mathcal{T}\rvert$	-	-	-	-
Decrypt$_{Usr}$	-	-	-	2	1	1	-
Decrypt$_{Srv}$	-	-	-	$2\lvert\mathcal{T}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{T}\rvert$	$2\lvert\mathcal{T}\rvert+1$
GetMaskKey	-	-	1	-	-	-	-

Table 1: Sizes of Data.

	Data Size	Direction
Public key PK	$\lvert\mathbb{G}_0\rvert+\lvert\mathbb{G}_1\rvert$	TA $\rightarrow$ User
Private Key SK	$\lvert\mathbb{Z}_p\rvert+2h\lvert\mathbb{G}_0\rvert$	TA $\rightarrow$ User
Ciphertext CT_{Usr}	$\lvert\mathbb{Z}_p\rvert+h\lvert\mathcal{T}_h\rvert$ $+4\lvert\mathbb{G}_0\rvert+\lvert\mathbb{G}_1\rvert$	User $\rightarrow$ Server
Ciphertext CT_{Srv}	$2\lvert\mathbb{G}_1\rvert$	Server $\rightarrow$ User
Ciphertext CT	$\lvert\mathcal{T}_{h+1}\rvert+4\lvert\mathbb{G}_0\rvert$ $+\lvert\mathbb{G}_1\rvert$	Server (local)
Token T	$\lvert\mathbb{G}_1\rvert$	TSP $\rightarrow$ User
Input of Token$\hat{C}$	$\lvert\mathbb{G}_0\rvert$	User $\rightarrow$ TSP

4.3 Performance Analysis

Our scheme reduces the computational cost on a mobile device; the computational cost of **Encrypt**$_{Usr}$, **Decrypt**$_{Usr}$ is less than that of the existing scheme (Hinek et al., 2008).

The function **GetMaskKey** is pre-computed before computing Decrypt$_{Usr}$ and it requires $O(1)$ computation. The computational cost of **Decrypt**$_{Usr}$ is also $O(1)$. Thus, the computational cost of decryption on a mobile device is $O(1)$.

The total computational cost of the whole process is $O(|\mathcal{T}|)$. The worst case is that where N attributes, and in that case, the total computational cost is estimated to be $O(N)$. Here we show the estimation of the computational cost of each operation in Table 2. The symbol S_u is the set of attributes allocated to a user u.

We show the computational cost of Zhou's scheme in Table 3. The computational costs of functions **KeyGeneration**, **Encrypt**$_{Usr}$, and **GetMaskKey** in our scheme are somewhat larger than those in Zhou's scheme. **GetMaskKey** can be pre-computed before decryption of any data. The increases of other functions are constant time; in the view of an asymptotic estimation, these increases are negligible. We conclude that the increases in computational cost from adding key revocation are negligibly small in our scheme.

4.4 Implementation of Primitives

We implemented a Reduced Modified Tate (RMT) pairing to estimate the cost of our scheme. Our target devices are a PC and an Android smartphone as shown in the Table 5.

Table 4: Parameters.

	89 bit security	**128 bit security**
m	193	509
Irreducible Polynomial	$f(x)=x^{193}-x^{64}+1$	$f(x)=x^{509}-x^{318}$ $-x^{191}+x^{127}+1$
Oder of $\mathbb{Z}_p$	$p=3^{193}-3^{97}+1$	$p=(3^{509}-3^{255}+1)/7$

Table 5: Target Devices.

	PC	**Mobile Devise(MD)**
Model	-	HTC Sensation XE
CPU	Intel Xeon W3680 3.33GHz	Qualcomm Snapdragon MSM8260 1.5GHz
Core	Hexa-core	Dual-core
Word length	64-bit	32-bit
Memory	16GB	768MB
OS	CentOS6.0	Android 2.3.4
Compiler	GCC4.4.4	Android NDK r7

Table 6: Transaction Time in 89-bit security per function (msec).

	The number of attributes (AND condition)	1	2	3	4	5
Our scheme	**Encrypt**$_{Usr}$: Android	157.29	133.88	149.25	130.10	130.94
	Encrypt$_{Srv}$: PC	7.28	15.30	22.72	29.96	32.02
	Decrypt$_{Usr}$: Android	29.18	30.40	32.82	32.16	34.28
	Decrypt$_{Srv}$: PC	17.46	24.47	26.64	39.56	46.82
CP-ABE(Bethencourt et al., 2007)	**Encrypt** : Android	95.9	140.2	184.5	228.8	273.1
	Decrypt : Android	96.9	169.7	242.5	315.3	388.1

Table 7: Transaction Time in 128-bit security per function (msec).

	The number of attributes (AND condition)	1	2	3	4	5
Our scheme	**Encrypt**$_{Usr}$: Android	557.99	556.95	586.17	558.71	557.86
	Encrypt$_{Srv}$: PC	51.48	90.14	117.89	148.13	184.87
	Decrypt$_{Usr}$: Android	185.28	189.24	193.12	184.27	193.00
	Decrypt$_{Srv}$: PC	115.54	174.10	244.51	286.07	322.59
CP-ABE(Bethencourt et al., 2007)	**Encrypt** : Android	1079	1599	2119	2639	3159
	Decrypt : Android	743	1302	1861	2420	2979

Our implementation is similar to that of Beuchat *et al.* (Beuchat et al., 2010). We adopted a design for the RMT pairing based on supersingular elliptic curves defined over a finite field of characteristic three and select the parameters for two security levels: 89-bit security and 128-bit security.

We consider the supersingular curves $E(\mathbb{F}_{3^m})$ with embedding degree $k = 6$ defined as follows;

$E(\mathbb{F}_{3^m}) = \{(x,y) \in \mathbb{F}_{3^m} \times \mathbb{F}_{3^m} \,|\, y^2 = x^3 - x + 1\} \cup \{\infty\}$. The order of $E(\mathbb{F}_{3^m})$, $\#E(\mathbb{F}_{3^m})$ is calculated as $\#E(\mathbb{F}_{3^m}) = 3^m + 1 + b'3^{(m+1)/2}$, where b' is defined as follows:

$$b' = \begin{cases} 1 \pmod{12} & \text{if } m \equiv 1, 11 \\ -1 \pmod{12} & \text{if } m \equiv 5, 7 \end{cases}$$

The RMT pairing $\hat{e}_r$ is defined using η_T pairing as follows:

$$\hat{e}_r(P,Q) = \eta_T([-\mu b' 3^{\frac{3m-1}{2}}]P,Q)^{\frac{3^{6m}-1}{N}},$$

where $P,Q \in E(\mathbb{F}_{3^m})$. The orders of groups $\mathbb{G}_0 \mathrm{C} \mathbb{G}_1$ are $\#E(\mathbb{F}_{3^m})$, $\#\mathbb{F}^*_{3^{6m}}$ respectively.

The η_T pairing consists of a main loop and a final exponentiation. We used a loop unrolling technique (Beuchat et al., 2010) for the main loop, and the final exponentiation calculation was carried out according to (Shirase et al., 2008). The computational cost of a multiplication over $\mathbb{F}_{3^m}$ is dominant in the η_T pairing. We implemented a window method (Brauer, 1939) that was the fastest multiplication algorithm for a characteristic three field, and selected a window size of $w = 4$. We selected the parameters for two security levels as shown in the Table 4.

4.5 Evaluation Result

We implemented our ABE system on the target devices in Table 5. Two functions, **Encrypt**$_{Usr}$ and **Decrypt**$_{Usr}$, were implemented on the smartphone and two other functions, **Encrypt**$_{Srv}$ and **Decrypt**$_{Srv}$, were implemented on the PC. Evaluation results of the transaction time of the four functions for 89-bit and 128-bit security settings are shown in Table 6 and Table 7, respectively. An access policy that defines an access condition **AND** of all w attributes ($w = 1,2,3,4,5$) was used for the evaluation. The access policy is the worst case for the transaction time where we use w attributes in the system. Constant transaction time is required to compute two functions on the smartphone, **Encrypt**$_{Usr}$ and **Decrypt**$_{Usr}$, and transaction time of **Encrypt**$_{Srv}$ and **Decrypt**$_{Srv}$ are proportional to the number of attributes; but are negligibly small for the PC. We compare our results with a CP-ABE scheme proposal by Bethencourd *et al.* (Bethencourt et al., 2007), which is a conventional CP-ABE and requires that the smartphone must execute all calculations for encryption and decryption. As evaluation results of primitive functions, the transaction times of the encryption and decryption of the CP-ABE using 5 attributes are estimated to be 271.3 msec and 388.1 msec for 89-bit security and 3159 msec and 2979 msec for 128-bit security respectively. These results show that our scheme is faster than the scheme (Bethencourt et al., 2007) where all functions are implemented on the smartphone, and are applicable to a mobile cloud environment.

5 CONCLUSIONS

In this paper, we proposed a new ABE scheme applicable to mobile cloud computing environments. The computational cost on a mobile device is $O(1)$, and cloud services can securely provide most computations required for the ABE scheme. Furthermore, the scheme includes a key revocation mechanism for the private keys and an attribute hiding mechanism. The transaction time of our encryption algorithm is within 150 msec for 89-bit security and about 600 msec for 128-bit security on the mobile device, respectively. Similarly, the transaction time of the decryption algorithm is within 50 msec for 89-bit security and 200 msec for 128-bit security. The evaluation of transaction time demonstrated that our scheme is feasible for mobile cloud services using current smartphones.

REFERENCES

Amazon (2012). Amazon Web Services. http://aws.amazon.com/.

Aranha, D., López, J., and Hankerson, D. (2010). High-speed parallel software implementation of the η_T pairing. In *Topics in Cryptology - CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 89–105. Springer.

Baek, J., Safavi-Naini, R., and Susilo, W. (2005). Token-controlled public key encryption. In *Information Security Practice and Experience*, volume 3439 of *Lecture Notes in Computer Science*, pages 386–397. Springer.

Bellare, M., Pointcheval, D., and Rogaway, P. (2000). Authenticated key exchange secure against dictionary attacks. pages 139–155. Springer.

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

Beuchat, J., Gonzalez-Diaz, J., Mitsunari, S., Okamoto, E., Rodriguez-Henriquez, F., and Teruya, T. (2010). High-speed software implementation of the optimal Ate pairing over Barreto-Naehrig curves. In *Pairing-Based Cryptography - Pairing 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 21–39. Springer.

Boneh, D. (1998). The decision diffie-hellman problem. pages 48–63. Springer.

Boneh, D. and Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer.

Brauer, A. (1939). On addition chains. In *Bull. Amer. Math. Soc.*, volume 45, pages 736–739.

Canetti, R., Krawczyk, H., and Nielsen, J. (2003). Relaxing chosen-ciphertext security. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer.

Cloud Security Alliance (2009). Security guidance for critical areas of focus in cloud computing. http://tinyurl.com//ycrchqj.

European Network and Information Security Agency (2010). Cloud computing risk assessment. http://www.enisa. europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport.

Galindo, D. and Herranz, J. (2006). A generic construction for token-controlled public key encryption. In Di Crescenzo, G. and Rubin, A., editors, *Financial Cryptography and Data Security*, volume 4107 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Google (2012). Google App for Buiziness. http://www.google.com/apps/intl/en/business/index.html.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 89–98. Algorithms and Computation in Mathematics.

Green, M., Hohenberger, S., and Waters, B. (2011). Outsourcing the decryption of ABE ciphertexts. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 34–34. USENIX Association.

Hinek, M. J., Jiang, S., Safavi-Naini, R., and Shahandashti, S. F. (2008). Attribute-based encryption with key cloning protection. Cryptology ePrint Archive, Report 2008/478.

Joux, A. (2004). A one round protocol for tripartite diffie?hellman. *Journal of Cryptology*, 17:263–276.

Lewko, A., Okamoto, T., Sahai, A., Takashima, K., and Waters, B. (2010). Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer.

Naehrig, M., Niederhagen, R., and Schwabe, P. (2010). New software speed records for cryptographic pairings. In *Progress in Cryptology - LATINCRYPT 2010*, volume 6212 of *Lecture Notes in Computer Science*, pages 109–123. Springer.

NIST (2009). The nist definition of cloud computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

Sadeghi, A., Schneider, T., and Winandy, M. (2010). Token-based cloud computing. In *Trust and Trustworthy Computing*, volume 6101 of *Lecture Notes in Computer Science*, pages 417–429. Springer.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 557–557. Springer.

Scott, M. (2011). On the efficient implementation of pairing-based protocols. Cryptology ePrint Archive, Report 2011/334.

Shirase, M., Takagi, T., and Okamoto, E. (2008). Some efficient algorithms for the final exponentiation of η_T pairing. *IEICE Transactions*, 91-A(1):221–228.

Waters, B. (2011). Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography - PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer.

Zhou, Z. and Huang, D. (2011). Efficient and secure data storage operations for mobile cloud computing. Cryptology ePrint Archive, Report 2011/185.