Outsourcing Scheme of ABE Encryption Secure against Malicious Adversary

Go Ohtake[1], Reihaneh Safavi-Naini[2] and Liang Feng Zhang[3]
[1]*Science & Technology Research Laboratories, Japan Broadcasting Corporation, Tokyo, Japan*
[2]*Department of Computer Science, University of Calgary, Calgary, Canada*
[3]*School of Information Science and Technology, ShanghaiTech University, Shanghai, China*
ohtake.g-fw@nhk.or.jp, rei@ucalgary.ca, zhanglf@shanghaitech.edu.cn

Keywords: Outsourcing Scheme, Attribute-based Encryption, Malicious Adversary, Random Oracle Model.

Abstract: Integrated broadcast-broadband services allow viewers to simultaneously receive broadcast content over the airwaves and additional information related to the content over the Internet. This integration provides opportunities for new services to be tailored and offered to individual viewers. Viewing histories provide a rich variety of data for service providers to learn the preferences of individual viewers and fine-tune their offerings. Each person's viewing history, however, is privacy-sensitive data and may reveal information that the viewer does not want revealed. In this paper, we propose a system that allows viewers to specify a policy that they would like to be applied to their viewing history, when shared with service providers, by using attribute-based encryption (ABE). A ciphertext is associated with a policy, and it can be decrypted only by service providers who conform to the policy. To reduce the computations of the user terminal, we develop a system with provable security that allows the encryption to be outsourced to a cloud server, without the need to trust the cloud server. Although our solution is described for integrated broadcast-broadband services, the architecture and results could also be used for sharing viewing histories of services such as Netflix. We implemented our scheme and showed that it significantly reduces the computation cost of a user terminal.

1 INTRODUCTION

1.1 Background

Integrated broadcast-broadband services (Baba et al., 2012; Ohmata et al., 2013; Ohmata et al., 2015; NHK, ; ETSI, ; BBC, ; KBS,) allow viewers to view content through broadcast and, simultaneously, additional content through the Internet. The additional content can be used to personalize broadcasts and provides opportunities for electronic commerce. For example, in a link navigation service (Ohmata et al., 2015), a service provider provides program-related keywords for goods, locations, and shops as the television program progresses. These keywords are linked to services that the viewers can access by clicking on the word shown on the screen on their mobile devices. To make the personalization of the broadcast and services effective, viewers must share their viewing preferences with the service provider. Viewing histories are a rich source of data for service providers to learn about viewers' interests. Their data, however, could reveal sensitive personal information about a viewer and so must be handled with care. Ideally, viewers want to share their viewing histories with service providers that pass certain criteria, including being trustworthy or having a high rating based on customer reviews. This, of course, can be achieved by encrypting a history using the public keys of the desired service providers, or by establishing a secure connection using a protocol such as TLS with those providers, and sending the data to them one-by-one. Although these solutions allow the viewer to fully control sharing of their viewing history, they have two major drawbacks. Firstly, they are not scalable: they are computationally expensive and require the viewer to encrypt the data, or establish a secure connection, for each provider independently, and this must be done at regular intervals in order to provide an up-to-date viewing history. Secondly, their use is limited to the service providers that the viewer knows, and unless the viewer does not make an effort to identify new service providers, the pool of services that will receive the data will be limited, meaning that the viewer may never learn about new services that emerge. An ideal solution for sharing viewing

Ohtake, G., Safavi-Naini, R. and Zhang, L.
Outsourcing Scheme of ABE Encryption Secure against Malicious Adversary.
DOI: 10.5220/0006129600710082
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 71-82
ISBN: 978-989-758-209-7

history would allow the viewer to produce a single encrypted copy of their viewing history, and specify an access policy that will be enforced on the service providers. It is also important to ensure that the computation of the user terminal is minimized. This is important not only because user terminals, such as television sets, cannot be expected to be equipped with powerful cryptoprocessors, but also because the computation must be performed at regular intervals as new viewing histories are generated. The terminal computations can be reduced by outsourcing them to cloud servers. This, however, raises the issue of trust. Outsourcing computations on the original data would allow a cloud server to learn the viewing history. Moreover, one must ensure that the requested computation has been performed correctly.

1.2 Our Contributions

In this paper, we propose a secure outsourcing system of *attribute-based encryption (ABE)* that (i) allows sharing of viewer's data according to a viewer-defined policy, (ii) reduces the viewer-side computation to a single encryption using ElGamal encryption, and (iii) ensures privacy of content and correctness of computation against a malicious cloud server. We use ABE to enable viewers to encrypt their own viewing history and specify the attributes of the service providers who can access the data. ABE provides an elegant means of expressing and enforcing access control policies. In a ciphertext-policy ABE, a service provider has a set of attributes that is verified by a trusted authority who issues the corresponding private key to them. A ciphertext can be generated by any viewer using the public key of the system. The ciphertext is attached to a policy that is expressed by a Boolean formula. A service provider whose attribute set satisfies this policy can use the decryption algorithm and their private key to recover the data. Encryption in ABE, however, is costly, and the computation cost grows linearly with the number of attributes. We design an outsourcing scheme that reduces the viewer-side computation to a single ElGamal encryption. This encryption protects the viewer's data against any attack from a cloud server. This ciphertext is then converted by the cloud server into an ABE ciphertext according to a policy that is provided by the viewer. We do not assume a trusted cloud server and design the outsourcing system that protects viewers against its potential misbehavior. We consider colluding attacks where the cloud server and service providers whose attributes do not satisfy the viewer's policy would like to make the viewing history readable to themselves.

We consider five different entities: a registration authority (RA), a key generation center (KGC), viewers, a cloud server, and service providers (See Figure 1). Viewers are registered with the RA and receive an ID (identification) and credentials that they can use when contacting the cloud server. This is to ensure that the data provided to the cloud server are valid and the system is not abused. Service providers are registered with the KGC and receive private keys associated with their verified attributes. The cloud server is not assumed to be honest. Each viewer's data must be protected for the sake of privacy. Also, the computation delegated to the cloud server must be verifiable. These requirements are captured in *security* and *unforgeability*, ensuring that (i) the viewers' personal information remains private in the system and only accessible to the service providers who have the required attributes, and (ii) the computation of the cloud server is verifiable by the viewers and the service providers. We use game-based definitions to model these requirements and devise an outsourcing scheme of ABE encryption that ensures these properties. We give formal proofs of security for these properties, assuming access to a random oracle by the participants. Our scheme is based on the "large universe" ABE scheme of Waters (See Appendix A in (Waters, 2008)) that supports an unlimited number of attributes and has constant-size public parameters for any policy that can be described by a Linear Secret Sharing Scheme (LSSS) matrix. We implement the encryption algorithms of our scheme and the ABE scheme of Waters (Waters, 2008) on a user terminal and measure the processing times. The experimental results show that our scheme reduces the encryption cost of a user terminal to about one third that of the ABE scheme of Waters (Waters, 2008), without assuming a trustworthy server.

Outsourcing Encryption. The encryption algorithm of an ABE scheme has a substantially higher computation cost than that of conventional public key encryption schemes such as RSA or ElGamal encryption. Therefore, a user terminal with a low-performance CPU, such as a television set or a mobile device, must bear a comparatively large computation cost when encrypting a viewing history. This problem has motivated a number of recent studies on outsourcing schemes for the computation (Hohenberger and Waters, 2014; Li et al., 2012; Zhou and Huang, 2011). In all these schemes, a large part of encryption process is outsourced to a cloud server. The schemes, however, assume that the cloud server is either *honest* and, while following the protocol, there is no collusion with other entities, or it is *honest-but-curious* and, while following the protocol, may collude with other entities to learn the private information of the users. Our work is the first to consider security against a *malicious* cloud server who does not follow the protocol and may collude with other entities. This is a realistic model for outsourcing the viewing history encryption when viewers cannot rely on a trustworthy cloud server.

Towards Attribute-based Sharing of Viewing Histories. Outsourcing the ABE ciphertext computation to a cloud server is an attractive solution in many application scenarios. Our scenario considers the resource-limited clients of integrated broadcast-broadband services, and content distribution services such as Netflix or Spotify have similar requirements. Here, the users' viewing histories are known by the service provider: in fact, they would be the basis of any recommendation system run by it. Moreover, users may be willing to share this data with third-party service providers for the purpose of receiving different services. For example, having a lot of movies for children in one's viewing history would suggest the viewer has an interest in toys, and in such case, relevant information can be provided directly by a third party or through the service provider. This sharing of information must meet the terms and conditions specified by the user that are encoded as a set of attributes that the third party must satisfy. The incentive for the service provider (e.g. Netflix) to outsource the encryption is the number of subscribers and the required update frequency of the viewing history. Once the service provider receives the user's consent and conditions, it uses the system proposed in this paper to minimize their computation.

1.3 Related Work

Sahai and Waters (Sahai and Waters, 2005) proposed the first ABE scheme as an extension of identity-based encryption (IBE). ABE schemes are attractive because they allow access control of content in realistic situations. ABE schemes can be classified into (i) key-policy ABE (KP-ABE) (e.g. (Goyal et al., 2006; Ostrovsky et al., 2007; Rouselakis and Waters, 2013; Attrapadung, 2014; Gay et al., 2015; Attrapadung et al., 2016)), and (ii) ciphertext-policy ABE (CP-ABE) (e.g. (Bethencourt et al., 2007; Cheung and Newport, 2007; Katz et al., 2008; Nishide et al., 2008; Waters, 2008; Emura et al., 2009; Lewko et al., 2010; Okamoto and Takashima, 2010; Ohtake et al., 2013; Rouselakis and Waters, 2013; Attrapadung and Yamada, 2015; Attrapadung et al., 2016)). In KP-ABE, a ciphertext is associated with a set of attributes, and a private key is associated with a policy. In CP-ABE, a private key is associated with a set of attributes, and a ciphertext is associated with a policy. A ciphertext can be decrypted by a user whose attributes satisfy the policy that is attached to the ciphertext. In this paper, we consider CP-ABE. Outsourcing of ABE was first considered in (Green et al., 2011), which proposed ABE decryption outsourcing with the goal of minimizing the users' decryption cost and assumed *honest-but-curious* cloud servers. In (Zhou and Huang, 2011), an outsourcing scheme for ABE encryption and decryption was proposed, also assuming *honest-but-curious* cloud servers. An outsourcing scheme for ABE encryption based on the scheme of (Zhou and Huang, 2011) and using a MapReduce cloud was proposed in (Li et al., 2012). In this scheme, distributed processing is performed by a number of cloud servers wherein at least one of them is assumed to be *honest*. The scheme in (Hohenberger and Waters, 2014) is an online/offline ABE scheme in which the encryption algorithm is divided into two parts: online encryption, which is performed by a user terminal, and offline encryption, which is performed by an *honest* cloud server. An outsourcing scheme for ABE key generation and decryption in which the outsourced computation results are checkable was proposed in (Li et al., 2014); this scheme assumes *honest-but-curious* cloud servers. To the best of our knowledge, ours is the first outsourcing scheme of ABE encryption for when the cloud server is potentially malicious.

2 PRELIMINARIES

Secret sharing schemes are proposed independently by (Shamir, 1979) and (Blakley, 1979). A more recent survey of the area can be found in (Beimel, 1996).

2.1 Access Structures

Definition 1 ***(Access Structure).*** *Let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection of subsets of $\mathcal{P}$, $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}}$, is a monotone access structure if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. A set A in $\mathbb{A}$ is called an authorized set, and a set A not in $\mathbb{A}$ is called an unauthorized set.*

In ABE, the role of the parties is taken by the attributes: the private key is a secret that is divided into shares, with each share assigned to an attribute. The access structure $\mathbb{A}$ includes the subset of attributes that can reconstruct the key and decrypt the ciphertext. We only consider monotone access structures.

2.2 Linear Secret Sharing Schemes

Definition 2 ***(Linear Secret Sharing Schemes (LSSS)).*** *A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if,*

1. *The shares of the parties form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix M with ℓ rows and n columns called the share-generating matrix for Π and a function ρ which maps each row of the matrix to an associated party. That is, for $i = 1,\dots,\ell$, the value $\rho(i)$ is the party associated with row i. Suppose we have a column vector $\vec{v} = (s, r_2, \dots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2,\dots,r_n \in \mathbb{Z}_p$ are randomly chosen, then $M\vec{v}$ is the vector of ℓ shares of the secret s according to Π. The share $(M\vec{v})_i$ belongs to party $\rho(i)$.*

It is shown in (Beimel, 1996) that every LSSS having the above definition enjoys the *linear reconstruction* property, defined as follows: Suppose that Π is a LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1,2,\dots,\ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i\in I} w_i\lambda_i = s$. Furthermore, it is shown in (Beimel, 1996) that these constants $\{w_i\}$ can be found in time polynomial in the size of the share-generating matrix M. Like any secret sharing scheme, it has the property that for any unauthorized set $S \notin \mathbb{A}$, the secret s should be information theoretically hidden from the parties in S.

2.3 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ be a bilinear map that has the following properties:

1. Bilinearity: $e(u^a, v^b) = e(u,v)^{ab}$ for all $u,v \in \mathbb{G}$ and $a,b \in \mathbb{Z}_p$.
2. Non-degeneracy: $e(g,g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric, since $e(g^a, g^b) = e(g,g)^{ab} = e(g^b, g^a)$.

2.4 Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption

The decisional q-parallel Bilinear Diffie-Hellman Exponent (BDHE) problem (Waters, 2008) is defined as follows. Choose a group $\mathbb{G}$ of prime order p according to the security parameter. Let $a, s, b_1, \dots, b_q \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. If an adversary is given $\vec{y} =$

$$\begin{gathered} g, g^s, g^a, \dots, g^{(a^q)}, g^{(a^{q+2})}, \dots, g^{(a^{2q})} \\ \forall_{1\le j\le q}\ \ g^{s\cdot b_j}, g^{a/b_j}, \dots, g^{(a^q/b_j)}, g^{(a^{q+2}/b_j)}, \dots, g^{(a^{2q}/b_j)} \\ \forall_{1\le j,k\le q, k\neq j}\ \ g^{a\cdot s\cdot b_k/b_j}, \dots, g^{(a^q\cdot s\cdot b_k/b_j)}, \end{gathered}$$

it must remain hard to distinguish $e(g,g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ε in solving the decisional q-parallel BDHE problem in $\mathbb{G}$ if

$$\left|\Pr\left[\mathcal{B}(\vec{y}, T = e(g,g)^{a^{q+1}s}) = 0\right] - \Pr\left[\mathcal{B}(\vec{y}, T = R) = 0\right]\right| \geq \varepsilon.$$

Definition 3. *We say that the decisional q-parallel BDHE assumption holds if no polynomial time algorithm has a non-negligible advantage in solving the decisional q-parallel BDHE problem.*

3 DEFINITIONS

3.1 Model

The model of outsourcing ABE encryption is defined as follows.

Definition 4. *The outsourcing scheme of ABE encryption is a tuple consisting of the following algorithms:*

Setup$(\lambda) \to (PK, MSK)$. *This algorithm takes as input a security parameter λ. It outputs the public parameter PK and the master key MSK.*

KeyGen$(PK, MSK, S) \to SK$. *This algorithm takes as input PK, MSK, and a set of attributes S. It outputs a private key SK associated with S.*

Encrypt$_u$(PK, $\mathbb{A}$, $\mathcal{M}$) $\to$ (CT$'$, π). *This algorithm takes as input PK, an access structure $\mathbb{A}$, and a message $\mathcal{M}$. It outputs an intermediate ciphertext CT' and a proof π.*

Encrypt$_c$(PK, $\mathbb{A}$, CT$'$) $\to$ CT. *This algorithm takes as input PK, $\mathbb{A}$, and CT'. It outputs a ciphertext CT.*

Decrypt$(CT, \pi, SK, PK) \to \mathcal{M}/\perp$. *This algorithm takes as input CT, π, SK, and PK. It outputs $\mathcal{M}$ if π is a valid proof of CT and S satisfies $\mathbb{A}$. Otherwise, it outputs the error message $\perp$.*

Figure 1 shows a model of an outsourcing scheme of ABE encryption for integrated broadcast-broadband services. There are five different entities:

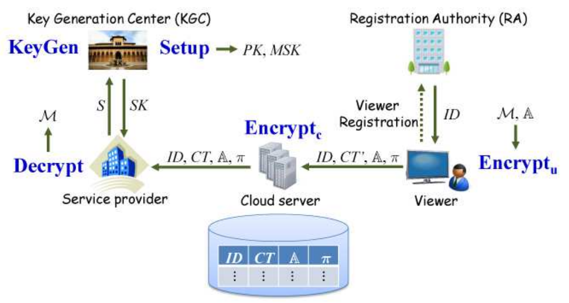

Figure 1: Model outsourcing scheme of ABE encryption for integrated broadcast-broadband service.

a registration authority (RA), a key generation center (KGC), viewers, a cloud server, and service providers. The RA registers viewers of broadcast services. A registered viewer is given identity information ID that allows a third party to verify their identity, possibly through interacting with the RA. The KGC sets up the access control system by running the **Setup** and **KeyGen** algorithms for the ABE system, and after verifying the service providers' attributes S, provides them with private keys SK associated with S. A registered viewer uses **Encrypt**$_u$ to encrypt their viewing history $\mathcal{M}$ that will be an ElGamal ciphertext sent to the cloud server. This first level of encryption ensures the privacy of the viewer information in the cloud server. The viewer provides to the cloud server a policy that they would like to enforce. The cloud server uses **Encrypt**$_c$ to convert the received ElGamal ciphertext into an ABE ciphertext that enforces the viewer's policy for the service providers, making it accessible to all service providers who possess the required attributes. Service providers who have received their private keys of the ABE system and whose attributes satisfy the viewer's policy can use **Decrypt** to access the viewing history of the viewer.

In the following, we focus on the access control part of the system and assume that the cloud server verifies the identity of the viewers who provide their viewing history, by using ID and an identity service that uses the RA registration process. This ensures that the viewing history is provided by valid viewers. Otherwise, viewing histories may be provided by fake viewers, resulting in unreliable data for service providers (resulting in, e.g., incorrect viewing statistics). Note that this does not pose a threat to the privacy of the viewer, as their viewing history is stored in encrypted form. The system works as follows: First, a viewer creates an intermediate ciphertext CT' of $\mathcal{M}$ with an access structure $\mathbb{A}$ to specify a condition for a service provider who can decrypt the viewing history. The viewer transmits ID, CT', $\mathbb{A}$, and a proof π to the cloud server. The cloud server transforms CT' into a ciphertext CT, and then it stores $(ID, CT, \mathbb{A}, \pi)$ for each viewer in a public database. A service provider downloads the tuple of $(ID, CT, \mathbb{A}, \pi)$ for a specified viewer and verifies its correctness by using π. Note that the viewer can verify the correctness of $(ID, CT, \mathbb{A}, \pi)$ stored in the public database. The service provider can decrypt CT and obtain a viewing history $\mathcal{M}$ iff the tuple is correct and the service provider has a set of attributes S satisfying $\mathbb{A}$.

3.2 Security Model

The following is our trust assumptions and security requirements for the outsourcing ABE encryption system. We assume that the viewers are *honest* and follow the protocol. (We exclude malicious viewers by using identity verification at the cloud server.) The cloud server is not trusted and may deviate from the protocol. This assumption accounts for the possibility that a cloud server may collude with service providers who do not satisfy the viewer's policy and enable them to bypass the viewer's access policy. Service providers follow the protocol, but may collude together, and possibly with the cloud server, to bypass the viewer's access policy and access his or her viewing history. Our basic security requirements are as follows:

1. A collusion of service providers, who individually do not satisfy the access policy of a viewer, and a cloud server cannot obtain any information about the viewing history of the viewer.
2. A cloud server cannot modify a ciphertext that is decrypted into a different viewing history from the original one.

Note that the cloud server in Requirement 2 would have no incentive for making an attack. In this case, the attack would interfere with a service, though it might be performed by the cloud server.

Fulfilling the above requirements would realize a secure integrated broadcast-broadband service. In Figure 1, a viewer transmits an intermediate ciphertext CT' to the cloud server for outsourcing part of the encryption process. CT' includes the viewing history of the viewer, so the cloud server might try to get it (Requirement 1). Then, a cloud server converts CT' into a ciphertext CT, but it might be "incorrect" (Requirement 2). Also, as is the case with conventional ABE schemes, the viewing history might be obtained from CT as a result of the service providers colluding (Requirement 1). In light of the above requirements, we define two kinds of security for an outsourcing scheme of ABE encryption: *security* and *unforgeability*.

Let $\Pi = ($**Setup**, **KeyGen**, **Encrypt**$_u$, **Encrypt**$_c$, **Decrypt**$)$ be an outsourcing scheme of ABE encryption. The definition of *security* includes attacks made by a cloud server that colludes with service providers. The goal of the attacker is to learn the plaintext that they are not supposed to learn. Formally, we use the following experiment to define security against an adversary $\mathcal{A}$.

Outsourcing of ABE Encryption Experiment $\mathsf{O\text{-}ABE\text{-}Exp}^{ind}_{\mathcal{A},\Pi}(\lambda)$**:**

Init. The adversary gives a challenge access structure $\mathbb{A}^*$ to the challenger.

Setup. The challenger runs the **Setup** algorithm and gives the public parameter PK to the adversary.

Phase 1. The challenger initializes an empty table T, an empty set D, and an integer counter $j = 0$. Proceeding adaptively, the adversary can repeatedly make any of the following queries:

- Create(S): The challenger sets $j := j + 1$. It runs the **KeyGen** algorithm on S to obtain the private key SK and stores in table T the entry (j, S, SK). Then it returns SK to the adversary.
- Corrupt(i): If the i^{th} entry in table T exists, the challenger obtains the entry (i, S, SK) and sets $D := D \cup \{S\}$. Then it returns SK to the adversary. If no such entry exists, it returns $\perp$.

Challenge. The adversary submits two messages $\mathcal{M}_0$ and $\mathcal{M}_1$. The challenger flips a random coin $b \in \{0,1\}$. It runs the algorithm **Encrypt**$_u$**(**PK**,** $\mathbb{A}^*$**,** $\mathcal{M}_b$**)** to obtain (CT', π^*). The challenger returns (CT', π^*) to the adversary.

Phase 2. Phase 1 is repeated with the restrictions that the adversary cannot trivially obtain a private key for the challenge ciphertext (CT^*, π^*) derived from (CT', π^*). That is, the adversary cannot make a Corrupt query that would result in a set of attributes S such that $f(S, \mathbb{A}^*) = 1$ being added to D.

Guess. The adversary outputs a guess b' of b. The output of the experiment is 1 iff $b = b'$.

Definition 5 ***(Security).*** *An outsourcing scheme of ABE encryption* Π *is secure if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function negl such that:*

$$\Pr[\mathsf{O{-}ABE{-}Exp}^{ind}_{\mathcal{A},\Pi}(\lambda) = 1] \le \frac{1}{2} + negl(\lambda).$$

The above definition includes an attack in which the cloud server receives an intermediate-ciphertext (CT', π) from a viewer and creates a ciphertext (CT^*, π^*), where $\pi^* \neq \pi$ and CT^* can be decrypted into the original message by colluding with service providers whose attributes do not satisfy the policy.

Remark 1. *The above security implies the security of conventional ABE schemes. This is because an adversary can obtain a challenge ciphertext* CT^* *from* CT' *in the Challenge phase of the security game by using the* ***Encrypt***$_c$ *algorithm.*

Next, we define the *unforgeability* of our outsourcing scheme. This security definition includes attacks made by a cloud server. The goal of the attacker is to modify a ciphertext so that it is decrypted to a different plaintext from the original one. We use the following experiment to define unforgeability with respect to an adversary $\mathcal{A}$.

Outsourcing of ABE Encryption Experiment $\mathsf{O\text{-}ABE\text{-}Exp}^{unf}_{\mathcal{A},\Pi}(\lambda)$**:**

Init. The adversary gives a challenge access structure $\mathbb{A}^*$ to the challenger.

Setup. The challenger runs the **Setup** algorithm and gives the public parameter PK to the adversary.

Query. The challenger initializes an empty table T, an empty set D, and an integer counter $j = 0$. Proceeding adaptively, the adversary can repeatedly make any of the following queries:

- Create(S): The challenger sets $j := j + 1$. It runs the **KeyGen** algorithm on S to obtain the private key SK and stores in table T the entry (j, S, SK). Then it returns SK to the adversary.
- Corrupt(i): If the i^{th} entry exists in table T, the challenger obtains the entry (i, S, SK) and sets $D := D \cup \{S\}$. It then returns SK to the adversary. If no such entry exists, it returns $\perp$.

Challenge. The adversary submits a challenge message $\mathcal{M}^*$. The challenger runs the algorithm **Encrypt**$_u$**(**PK**,** $\mathbb{A}^*$**,** $\mathcal{M}^*$**)** to obtain (CT', π^*). The challenger returns (CT', π^*) to the adversary.

Output. The adversary outputs (CT^*, π^*). The output of the experiment is 1 if **Decrypt**(CT^*, π^*, SK, PK) $\notin \{\perp, \mathcal{M}^*\}$.

Definition 6 ***(Unforgeability).*** *An outsourcing scheme of ABE encryption* Π *is unforgeable if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function negl such that:*

$$\Pr[\mathsf{O{-}ABE{-}Exp}^{unf}_{\mathcal{A},\Pi}(\lambda) = 1] \le negl(\lambda).$$

Remark 2. *In the Output phase, we assume that the adversary outputs a proof* π^* *which is the same as the proof received in the Challenge phase. This is because the adversary can easily create (*CT^**,* π^{**}*)*

satisfying $\pi^{**} \neq \pi^*$ and ***Decrypt****(CT^*, π^{**}, SK, PK) $\notin \{\perp, \mathcal{M}^*\}$ by choosing $\mathcal{M}^{**} \neq \mathcal{M}^*$ and running the* ***Encrypt$_u$*** *and* ***Encrypt$_c$*** *algorithms.*

4 PROPOSED SCHEME

We propose an outsourcing scheme for Waters' "large universe" ABE scheme (See Appendix A in (Waters, 2008)), where the number of attributes is unlimited, while the public parameter size is constant. Waters' scheme provides flexibility for the service provider to add new attributes and efficiency for storage of the public key. A policy is expressed as a monotonic Boolean formula that can be mapped into the share-generating matrix of a LSSS that is used in the ABE system.

4.1 Outsourcing Scheme of ABE Encryption

As described in Section 3.2, we assume that the cloud server is *malicious* and may not follow the protocol. Our scheme masks the secret in a column vector of a LSSS using random numbers. This prevents the cloud server from learning about the secret from the masked vector, given the intermediate ciphertext. It also uses the *hash of randomness* technique to verify the correctness of the ciphertext.

Our scheme has the following five algorithms:

Setup$()$**.** The setup algorithm chooses a bilinear group $\mathbb{G}$ of prime order p, a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, a generator g of $\mathbb{G}$, and hash functions $H : \{0,1\}^* \to \mathbb{G}$, $H' : \{0,1\}^* \times \mathbb{G}_T \times \mathbb{G}_T \times \{0,1\}^* \to \{0,1\}^*$ and $H'' : \{0,1\}^* \to \{0,1\}^*$. In addition, it chooses random numbers $\alpha, a \in \mathbb{Z}_p$. The algorithm outputs the public parameter,

$$PK = \{g, e(g,g)^\alpha, g^a, H(\cdot), H'(\cdot,\cdot,\cdot,\cdot), H''(\cdot)\},$$

and the master key $MSK = g^\alpha$.

KeyGen(MSK, S)**.** The key generation algorithm takes as input MSK and a set of attributes S. The algorithm first chooses a random value $t \in \mathbb{Z}_p$. It creates

$$K = g^\alpha g^{at}, L = g^t, K_x = H(x)^t \ (\forall x \in S).$$

The algorithm outputs a private key,

$$SK = \{K, L, K_x \ (\forall x \in S)\}.$$

Encrypt$_u$ $(PK, ID, (M, \rho), \mathcal{M})$**.** The algorithm takes as input PK, a viewer's identity ID, a LSSS access structure (M, ρ), and a message $\mathcal{M}$. We assume that each element of the access matrix M is either 0 or 1. The function ρ associates rows of M with attributes. In this construction, we limit ρ to be an injective function; i.e., an attribute is associated with at most one row of M.

Let M be an $\ell \times n$ access matrix. The algorithm randomly chooses $s, y_2, ..., y_n, \beta_1, \beta_2, ..., \beta_n \in \mathbb{Z}_p$ and sets a column vector $\vec{v} = (s+\beta_1, y_2+\beta_2, \ldots, y_n+\beta_n) \in \mathbb{Z}_p^n$. It then calculates

$$C = \mathcal{M} \cdot e(g,g)^{\alpha s}, C' = g^s,$$
$$\pi = H'(ID, C, e(g,g)^{\alpha s}, H''(\mathsf{pos}_M)).$$

Here, pos_M is a string showing all of the elements in the access matrix M that are 1. For example, if the matrix M is

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix},$$

then pos_M is as follows:

$$\mathsf{pos}_M = \{(1,2),(2,1),(2,2),(3,1)\}.$$

For $1 \leq i \leq \ell$, let J_i be a set $J_i = \{j : M_{ij} = 1 \, (1 \leq j \leq n)\}$. The algorithm calculates

$$E_i = g^{a \sum_{j \in J_i} \beta_j}$$

and outputs an intermediate ciphertext,

$$CT' = \{C, C', (E_i)_{1 \leq i \leq \ell}, \vec{v}, (M, \rho)\},$$

and a proof π.

The algorithm takes as input PK and CT'. For $1 \leq i \leq \ell$, the algorithm calculates $\lambda_i = M_i \vec{v}$, where M_i is the row vector corresponding to the ith row of M. The algorithm also chooses random numbers $r_1, \ldots, r_\ell \in \mathbb{Z}_p$, and calculates

$$C_i = \frac{g^{a\lambda_i} H(\rho(i))^{-r_i}}{E_i}, D_i = g^{r_i} \ (1 \leq i \leq \ell)$$

and outputs a ciphertext,

$$CT = \{C, C', (C_i, D_i)_{1 \leq i \leq \ell}, (M, \rho)\}.$$

Decrypt(CT, π, SK, PK, ID)**.** The decryption algorithm takes as input CT, π, SK, PK, and ID. Suppose that S satisfies the access structure, and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Moreover, let $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} w_i \lambda_i = s$. The algorithm computes

$$e(C', K) / (\prod_{i \in I} (e(C_i, L) e(D_i, K_{\rho(i)}))^{w_i}) = e(g,g)^{\alpha s}.$$

If $\pi \neq H'(ID, C, e(g,g)^{\alpha s}, H''(\mathsf{pos}_M))$, the algorithm outputs $\perp$. This will happen in two cases: (1) CT was modified; (2) SK does not satisfy the policy in CT. Hence, $\perp$ will appear whenever (1) or (2) happens. Otherwise, it outputs the message,

$$\mathcal{M} = C/e(g,g)^{\alpha s}.$$

Remark 3. *Our scheme is secure even if the cloud server is malicious because (i) the vector $\vec{v} = (s, y_2, \ldots, y_n)$ that is used to create the shares of s is masked from an adversary by using random numbers $(\beta_1, \beta_2, \ldots, \beta_n)$, and (ii) to verify the correctness of the **Encrypt**$_c$ algorithm, a proof π is added to the ciphertext. The formal security proof is shown in Section 4.2.*

Remark 4. *The proof π is like a message authentication code (MAC), since it uses a hash function H' and a shared key $e(g,g)^{\alpha s}$. A viewer can verify the correctness of a tuple of (ID, CT, $\mathbb{A}$, π) stored in the cloud server if the viewer keeps the random number s that is generated in the **Encrypt**$_u$ algorithm. Namely, even if a malicious cloud server creates another tuple of (ID, CT^*, $\mathbb{A}^*$, π^*) from scratch and stores it in a public database, the viewer can detect the attack (although this requires the viewer to check the status of the database periodically).*

Another solution to prevent the above attack is using a digital signature scheme, but it might put a heavy load on a user terminal equipped with only a low-performance CPU. That's why we use a hash function to verify the correctness of the ciphertext.

Remark 5. *The hash function H' with four input values can be implemented by a secure hash function, and using the concatenation of all inputs: $H'(ID||C||e(g,g)^{\alpha s}||H''(\mathsf{pos}_M))$.*

4.2 Security Proof

Here, we prove that our scheme has the *security* and *unforgeability* properties defined in Section 3.2. The proof of *security* are based on (Waters, 2008).

Theorem 1. *Our scheme is selectively secure under the decisional q-parallel BDHE assumption in the random oracle model.*

Proof. Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage $\varepsilon = \mathsf{Adv}_{\mathcal{A}}$ in the selective security game against our construction. Moreover, suppose it chooses a challenge matrix M^* where both dimensions are at most q. Below, we show how to build a simulator $\mathcal{B}$ that solves the decisional q-parallel BDHE problem.

Init. The simulator $\mathcal{B}$ takes as input a q-parallel BDHE challenge $(\vec{y}, T)$. $\mathcal{B}$ runs the adversary $\mathcal{A}$, which gives $\mathcal{B}$ the challenge access structure (M^*, ρ^*). Here, M^* is $\ell^* \times n^*$ matrix and $\ell^*, n^* \leq q$.

Setup. $\mathcal{B}$ chooses a random value $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting $e(g,g)^{\alpha} = e(g^a, g^{a^q})e(g,g)^{\alpha'}$. $\mathcal{B}$ sends $\langle g, e(g,g)^{\alpha}, g^a \rangle$ to $\mathcal{A}$.

Phase 1. $\mathcal{B}$ initializes empty tables T_1, T_2, T_3, T_4, an empty set D, and an integer $j = 0$. It answers the adversary's queries as follows:

- Random Oracle Hash $H(x)$: If there is an entry (x, h) in T_1, return h. Otherwise, begin by choosing a random value z_x. If there is some index i such that $\rho^*(i) = x$, calculate

  $$h = g^{z_x} g^{aM^*_{i,1}/b_i} \cdot g^{a^2 M^*_{i,2}/b_i} \cdots g^{a^{n^*} M^*_{i,n^*}/b_i}.$$

  Record (x, h) in T_1 and return h. Note that if there is no index i such that $\rho^*(i) = x$, then $H(x) = g^{z_x}$. Note that the responses from the oracle are distributed randomly because of the value of g^{z_x}.
- Random Oracle Hash $H'(ID, \bar{C}, \bar{K}, \bar{V})$: If there is an entry $(ID, \bar{C}, \bar{K}, \bar{V}, h')$ in T_2, return h'. Otherwise, choose a random value $h' \in \{0,1\}^*$, record $(ID, \bar{C}, \bar{K}, \bar{V}, h')$ in T_2 and return h'.
- Random Oracle Hash $H''(P)$: If there is an entry (P, h'') in T_3, return h''. Otherwise, choose a random value $h'' \in \{0,1\}^*$, record (P, h'') in T_3 and return h''.
- Create(S): Set $j := j + 1$. Suppose S does not satisfy (M^*, ρ^*). Choose a random value $r \in \mathbb{Z}_p$. Find a column vector $\vec{w} = (w_1, \ldots, w_{n^*}) \in \mathbb{Z}_p^{n^*}$ such that $w_1 = -1$ and $M^*_i \vec{w} = 0$ for all i where $\rho^*(i) \in S$. Set

  $$L = g^r \prod_{i=1}^{n^*} (g^{a^{q+1-i}})^{w_i} = g^t$$

  by implicitly defining $t = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^{q+1-n^*}$. Compute K as

  $$K = g^{\alpha'} g^{ar} \prod_{i=2}^{n^*} (g^{a^{q+2-i}})^{w_i}.$$

  Next, we calculate $K_x (\forall x \in S)$. If $x \in S$ for which there is no i such that $\rho^*(i) = x$, simply let $K_x = L^{z_x}$. Otherwise, create

  $$K_x = L^{z_x} \prod_{j=1}^{n^*} \left(g^{(a^j/b_i)r} \prod_{k=1, k \neq j}^{n^*} (g^{a^{q+1+j-k}/b_i})^{w_k} \right)^{M^*_{i,j}}.$$

Finally, set $SK = \langle K, L, K_x(\forall x \in S)\rangle$, store (j, S, SK) in T_4, and return SK to $\mathcal{A}$.

- Corrupt(i): $\mathcal{A}$ cannot ask to corrupt any key corresponding to the challenge structure (M^*, ρ^*). If the ith entry exists in table T_4, $\mathcal{B}$ obtains the entry (i, S, SK) and sets $D := D \cup \{S\}$. It returns SK to $\mathcal{A}$. It returns $\perp$ if no such entry exists.

Challenge. $\mathcal{A}$ submits a message pair $(\mathcal{M}_0, \mathcal{M}_1)$ to $\mathcal{B}$. $\mathcal{B}$ flips a random coin $\beta \in \{0,1\}$ and sets

$$C = \mathcal{M}_\beta T \cdot e(g^s, g^{\alpha'}), \quad C' = g^s.$$

It then chooses random values $\beta_1, \ldots, \beta_{n^*}, \beta'_1, \ldots, \beta'_{n^*} \in \mathbb{Z}_p$. $\mathcal{B}$ sets

$$E_i = g^{a(\beta_1 M_{i,1} + \cdots + \beta_{n^*} M_{i,n^*})}$$

for $i = 1, \ldots, \ell^*$ and

$$\vec{v} = (\beta'_1, \ldots, \beta'_{n^*}).$$

$\mathcal{B}$ sets $CT' = \langle C, C', (E_i)_{1 \le i \le \ell^*}, \vec{v}, (M^*, \rho^*)\rangle$. It also sets $\bar{C} = C$, $\bar{K} = C/\mathcal{M}_\beta$, and $\bar{V} = H''(\mathsf{pos}_{M^*})$ by using the random oracle hash H''. Then, if the entry $(\bar{C}, \bar{K}, \bar{V}, h')$ exists in T_2, $\mathcal{B}$ sets $\pi^* = h'$. Otherwise, $\mathcal{B}$ chooses a random value $h' \in \{0,1\}^*$ and sets $\pi^* = h'$. Finally, $\mathcal{B}$ returns CT' and π^* to $\mathcal{A}$.

Phase 2. $\mathcal{B}$ continues to answer queries as in Phase 1.

Guess. $\mathcal{A}$ outputs a guess β' of β. If $\beta' = \beta$, $\mathcal{B}$ outputs 0 to indicate that $T = e(g,g)^{a^{q+1}s}$. Otherwise, it outputs 1 to indicate that T is a random group element in $\mathbb{G}_T$.

When $T = e(g,g)^{a^{q+1}s}$, $\mathcal{B}$ gives a perfect simulation, and we have

$$\Pr\left[\mathcal{B}(\vec{y}, T = e(g,g)^{a^{q+1}s}) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

On the other hand, when T is a random group element in $\mathbb{G}_T$, the message $\mathcal{M}_\beta$ is completely hidden from $\mathcal{A}$, and we have

$$\Pr\left[\mathcal{B}(\vec{y}, T = R) = 0\right] = \frac{1}{2}.$$

Therefore, $\mathcal{B}$ can solve the decisional q-parallel BDHE problem with non-negligible advantage. □

Theorem 2. *Our scheme is selectively unforgeable if the hash function H' is collision-resistant in the random oracle model.*

Proof. Let $\mathcal{A}$ be an adversary who breaks our scheme in the selective unforgeability game and $\mathcal{B}$ be a simulator that solves the decisional q-parallel DBHE problem. As described in Section 3.2, $\mathcal{A}$ wins if their output (CT^*, π^*) satisfies **Decrypt**$(CT^*, \pi^*, SK, PK) \notin \{\perp, \mathcal{M}^*\}$.

Init. Same as **Init** in the proof of Theorem 1.

Setup. Same as **Setup** in the proof of Theorem 1.

Query. Same as **Phase 1** in the proof of Theorem 1.

Challenge. $\mathcal{A}$ gives a challenge message $\mathcal{M}^*$ to the simulator $\mathcal{B}$. $\mathcal{B}$ then calculates an intermediate ciphertext,

$$CT' = \{C, C', (E_i)_{1 \le i \le \ell^*}, \vec{v}, (M^*, \rho^*)\},$$

and a proof π^* by using the **Encrypt**$_u$ algorithm. $\mathcal{B}$ returns (CT', π^*) to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs (CT^*, π^*). $\mathcal{B}$ outputs 1 if (CT^*, π^*) satisfies **Decrypt**$(CT^*, \pi^*, SK, PK) \notin \{\perp, \mathcal{M}^*\}$. Otherwise, it outputs 0.

Let $CT = \{C,\ C',\ (C_i, D_i)_{1 \le i \le \ell^*},\ (M^*, \rho^*)\}$ be a well-formed ciphertext that is obtained from CT' and can be decrypted into $\mathcal{M}^*$. Also, let $CT^* = \{\tilde{C}, \tilde{C}', (\tilde{C}_i, \tilde{D}_i)_{1 \le i \le \ell^*}, (\tilde{M}^*, \tilde{\rho}^*)\}$ be the ciphertext forged by $\mathcal{A}$. Remember that $C = \mathcal{M}^* \cdot \bar{K}$ ($\bar{K} = e(g,g)^{\alpha s}$) and $\bar{K}$ depends on C', C_i, and D_i. Therefore, there are only two cases in which $\mathcal{A}$ wins: one case is that the hash function H' has a collision and $\tilde{C}$ is not well-formed, and the other case is that H' has a collision and either $\tilde{C}'$, $\tilde{C}_i$, or $\tilde{D}_i$ is not well-formed . That is, the probability that $\mathcal{A}$ wins is as follows:

$$\begin{aligned}
&\Pr[\mathsf{O{-}ABE{-}Exp}^{unf}_{\mathcal{A},\Pi}(\lambda) = 1] \\
&= \Pr\left[(\pi^* = H'(ID, \tilde{C}, \bar{K}, H''(\mathsf{pos}_{M^*}))) \wedge (\tilde{C} \neq C)\right] \\
&+ \Pr\left[(\pi^* = H'(ID, C, \tilde{K}, H''(\mathsf{pos}_{M^*}))) \wedge ((\tilde{C}' \neq C')\right. \\
&\left.\vee (\tilde{C}_i \neq C_i)_{1 \le \exists i \le \ell^*} \vee (\tilde{D}_i \neq D_i)_{1 \le \exists i \le \ell^*})\right].
\end{aligned}$$

If H' is collision-resistant, the above probability is negligible. Therefore, our scheme is selectively unforgeable. □

5 PERFORMANCE

5.1 Comparison with Conventional Schemes

Table 1 compares the encryption costs of our scheme and the ABE scheme of Waters (Waters, 2008). In this table, $M_{\mathbb{G}}$ and $M_{\mathbb{G}_T}$ denote the cost of one modular exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, respectively, and ℓ denotes the number of attributes in the policy and also the number of rows of the LSSS matrix. Our scheme outsources 2ℓ modular exponentiations in $\mathbb{G}$ from the

Table 1: Comparison of our scheme and conventional ABE scheme.

	[Waters08]	Our scheme
Enc. cost for user	$(3\ell+1)M_{\mathbb{G}}+M_{\mathbb{G}_T}$	$(\ell+1)M_{\mathbb{G}}+M_{\mathbb{G}_T}$
Enc. cost for cloud	-	$(3\ell)M_{\mathbb{G}}$

Table 2: Comparison of our scheme and conventional outsourcing schemes of ABE.

	[ZH11]	[LJLC12]	[HW14]	Our scheme
Enc. cost for user	$3M_{\mathbb{G}}+M_{\mathbb{G}_T}$	$3M_{\mathbb{G}}+M_{\mathbb{G}_T}$	$2S_{\mathbb{Z}_p}+P_{\mathbb{Z}_p}$	$(\ell+1)M_{\mathbb{G}}+M_{\mathbb{G}_T}$
Enc. cost for cloud	$(2m)M_{\mathbb{G}}$	$(2m)M_{\mathbb{G}}$	$(5\ell+1)M_{\mathbb{G}}+M_{\mathbb{G}_T}$	$(3\ell)M_{\mathbb{G}}$
Cloud server	honest-but-curious	honest-but-curious	honest	malicious
Security assumption	-	-	$q-1$	decisional q-parallel BDHE
Security model	generic group	generic group	standard	random oracle
Distributed processing	no	yes	no	no

original Waters' scheme to a cloud server, resulting in a smaller cost for a user terminal. This will be a particularly significant saving when the number of attributes in the policy, ℓ, is large.

Table 2 compares the encryption cost of our scheme with that of the ABE outsourcing schemes in (Zhou and Huang, 2011; Li et al., 2012; Hohenberger and Waters, 2014). In this table, $S_{\mathbb{Z}_p}$ and $P_{\mathbb{Z}_p}$ denote the cost of one subtraction and one multiplication in $\mathbb{Z}_p$, respectively, and m denotes the number of leaf nodes in a tree representation of the policy statement. Note that the schemes in (Zhou and Huang, 2011; Li et al., 2012; Hohenberger and Waters, 2014) have a lower cost for a user terminal (for $\ell > 2$) but their security is against an *honest* or *honest-but-curious* cloud server, while our scheme provides security against a *malicious* cloud server. We may consider multiple cloud servers and distributed processing in a setting such as (Li et al., 2012). The security model however needs to consider *malicious* cloud servers.

5.2 Implementation Evaluation

We implemented the encryption algorithm of our scheme and that of Waters' scheme (Waters, 2008) on a PC and a tablet computer, representing a user terminal (See Table 3 for details).

We considered a viewing history consisting of 64 records and five possible attributes for the service provider. The viewing history is encrypted using AES and a temporary key. This key is then encrypted with the ABE. We measured the processing time of the encryption algorithm run on a user terminal as the average of 100 trials. In Figure 2 and 3, the horizontal axis denotes the number of attributes in the policy and the vertical axis denotes the processing time (seconds). For simplicity, we used a ciphertext policy consisting of only **AND**-gates, or only **OR**-gates. The cost of our scheme and Waters' scheme is labeled accord-

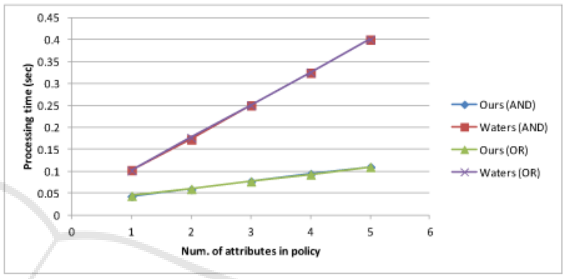

Figure 2: Experimental results (using PC).

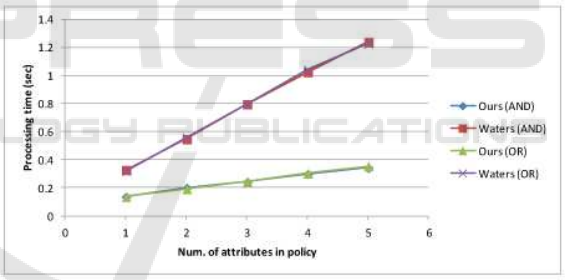

Figure 3: Experimental results (using tablet).

ingly for each case. It can be seen that our scheme costs much less than Waters' scheme. In particular, for five attributes, our scheme on a tablet computer takes 0.4 seconds, while Waters' scheme takes 1.2 seconds. Note that the processing time only depends on the number of attributes in the policy and is independent of the actual policy statement.

Remark 6. *We only measured the client-side encryption time. However, the measurements of the decryption time for a service provider, the encryption time for a cloud server, and the communication cost between a sender and a receiver do not make sense because a service provider and a cloud server would normally have a high-performance CPU and the protocol is not interactive.*

Table 3: Specifications of user terminal.

	PC	tablet
CPU	Intel Core i7-4790 (3.60GHz)	Apple A8X
Memory	8GB	-
OS	CentOS 7.2	iOS 9.2
Browser	Firefox 38.3.0	Firefox 1.4
Programming language	JavaScript	JavaScript

5.3 Discussion

Our experimental results in Section 5.2 show that our scheme significantly reduces the encryption cost of a user terminal. The CPUs in television sets are less powerful than those in PCs or tablet computers, and reducing costs is a very important consideration. We considered an attribute set size of five. A larger set of attributes would allow for a more flexible access control policy. The number of attributes in a policy can easily grow. For example, in a “5-level user rating” for an attribute of a service provider, “user rating ≥ 3” is represented as

“user rating= 3” **OR** “user rating= 4” **OR** “user rating= 5”

As noted earlier, our scheme has 2ℓ fewer modular exponentiations compared with Waters’ scheme. Hence, it can significantly reduce the encryption cost of the user terminal.

6 CONCLUSIONS

We considered outsourcing of ABE encryption to a malicious cloud server. We examined the security requirements, formalized the security model, and constructed a scheme with provable security based on Waters’ ABE scheme. We compared implementations of our scheme and Waters’ original scheme. Our security proof is in the random oracle model. Constructing an outsourcing scheme of ABE encryption that is secure without a random oracle remains an open problem. Our approach can be further refined by breaking the viewing history into smaller portions and labeling them according to the type of program. Encryption can be selectively performed on each portion, each with possibly a different policy, for relevant groups of service providers, thereby giving viewers finer control over their viewing history. For example, the user could allow his or her viewing history of food programs to be shared with service providers that have the label “food”. Constructing an outsourcing scheme for ABE encryption that allows a viewer to specify more than one policy for accessing their viewing history is another open problem.

ACKNOWLEDGEMENTS

Reihaneh Safavi-Naini’s research is supported in part by Albetra Innovates Technology Futures in the Province of Alberta, Canada. Liang Feng Zhang’s research is supported by National Natural Science Foundation of China (Grant No. 61602304) and Shanghai Pujiang Talent Program (No. 16PJ1406500).

REFERENCES

Attrapadung, N. (2014). Dual system encryption via doubly selective security: Framework, fully-secure functional encryption for regular languages, and more. In *Proc. of Eurocrypt’14*, pages 557–577.

Attrapadung, N., Hanaoka, G., Ogawa, K., Ohtake, G., Watanabe, H., and Yamada, S. (2016). Attribute-based encryption for range attributes. In *Proc. of SCN’16*, pages 42–61.

Attrapadung, N. and Yamada, S. (2015). Duality in abe: Converting attribute based encryption for dual predicate and dual policy via computational encodings. In *Proc. of CT-RSA’15*, pages 87–105.

Baba, A., Matsumura, K., Mitsuya, S., Takechi, M., Fujisawa, H., Hamada, H., Sunasaki, S., and Katoh, H. (2012). Seamless, synchronous, and supportive: Welcome to hybridcast: An advanced hybrid broadcast and broadband system. *IEEE Consumer Electronics Magazine*, 1(2):43–52.

BBC. YouView: Extraordinary TV for everyone. http://www.youview.com/.

Beimel, A. (1996). *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology.

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

Blakley, G. R. (1979). Safeguarding cryptographic keys. In *Proc. of AFIPS National Computer Conference*, volume 48, pages 313–317.

Cheung, L. and Newport, C. (2007). Provably secure ciphertext policy abe. In *Proc. of ACM Conference on Computer and Communications Security 2007*, pages 456–465.

Emura, K., Miyaji, A., Nomura, A., Omote, K., and Soshi, M. (2009). A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In *Proc. of ISPEC'09*, pages 13–23.

ETSI. TS 102 796: Hybrid Broadcast Broadband TV; V1.3.1.

Gay, R., Meaux, P., and Wee, H. (2015). Predicate encryption for multi-dimensional range queries from lattices. In *Proc. of PKC'15*, pages 752–776.

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

Green, M., Hohenberger, S., and Waters, B. (2011). Outsourcing the decryption of abe ciphertexts. In *Proc. of USENIX Security Symposium 2011*.

Hohenberger, S. and Waters, B. (2014). Online/offline attribute-based encryption. In *Proc. of PKC'14*, pages 293–310.

Katz, J., Sahai, A., and Waters, B. (2008). Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proc. of Eurocrypt'08*, pages 146–162.

KBS. icon (in Korean). http://icon.kbs.co.kr/site/main/main.php.

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 *Proc. of Eurocrypt'10*, pages 62–91.

Li, J., Huang, X., Li, J., Chen, X., and Xiang, Y. (2014). Securely outsourcing attribute-based encryption with checkability. *IEEE Trans. Parallel and Distributed Systems*, 25(8):2201–2210.

Li, J., Jia, C., Li, J., and Chen, X. (2012). Outsourcing encryption of attribute-based encryption with mapreduce. In *Proc. of ICICS'12*, pages 191–201.

NHK. Hybridcast (in Japanese). http://www.nhk.or.jp/hybridcast/online/.

Nishide, T., Yoneyama, K., and Ohta, K. (2008). Attribute-based encryption with partially hidden encryptor-specified access structures. In *Proc. of ACNS'08*, pages 111–129.

Ohmata, H., Endo, H., Baba, A., Matsumura, K., Sunasaki, S., and Kai, K. (2015). System architecture for cross-channel application services on hybridcast. In *Proc. of IEEE International Conference on Consumer Electronics 2015 (ICCE 2015)*, pages 108–109.

Ohmata, H., Takechi, M., Mitsuya, S., Otsuki, K., Baba, A., Matsumura, K., Majima, K., and Sunasaki, S. (2013). Hybridcast: A new media experience by integration of broadcasting and broadband. In *Proc. of the ITU Kaleidoscope Academic Conference 2013*.

Ohtake, G., Hironaka, Y., Kai, K., Endo, Y., Hanaoka, G., Watanabe, H., Yamada, S., Kasamatsu, K., Yamakawa, T., and Imai, H. (2013). Partially wildcarded attribute-based encryption and its efficient construction. In *Proc. of SECRYPT'13*, pages 339–346.

Okamoto, T. and Takashima, K. (2010). Fully secure functional encryption with general relations from the decisional linear assumption. In *Proc. of Crypto'10*, pages 191–208.

Ostrovsky, R., Sahai, A., and Waters, B. (2007). Attribute-based encryption with non-monotonic access structures. In *Proc. of ACM Conference on Computer and Communications Security 2007*, pages 195–203.

Rouselakis, Y. and Waters, B. (2013). Practical constructions and new proof methods for large universe attribute-based encryption. In *Proc. of ACMCCS'13*, pages 463–474.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *Proc. of Eurocrypt'05*, pages 457–473.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22:612–613.

Waters, B. (2008). Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *eprint 2008/290*.

Zhou, Z. and Huang, D. (2011). Efficient and secure data storage operations for mobile cloud computing. In *eprint 2011/185*.