ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/319053963

Auditable σ−Time Outsourced Attribute−Based Encryption for Access Control in Cloud Computing

Article *in* IEEE Transactions on Information Forensics and Security · August 2017

DOI: 10.1109/TIFS.2017.2738601

CITATIONS
46

READS
242

6 authors, including:

Jianting Ning
National University of Singapore
30 PUBLICATIONS **310** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Searchable Encryption View project

All content following this page was uploaded by Jianting Ning on 15 October 2018.

The user has requested enhancement of the downloaded file.

Auditable σ-Time Outsourced Attribute-Based Encryption for Access Control in Cloud Computing

Jianting Ning, Zhenfu Cao, *Senior Member, IEEE,* Xiaolei Dong, Kaitai Liang, *Member, IEEE,* Hui Ma, and Lifei Wei

***Abstract*—As a sophisticated mechanism for secure fine-grained access control over encrypted data, ciphertext-policy attribute-based encryption (CP-ABE) is one of the highly promising candidates for cloud computing applications. However, there exist two main long-lasting open problems of CP-ABE that may limit its widely deployment in commercial applications. One is that decryption yields expensive pairing cost which often grows with the increase of access policy size. The other is that one is granted access privilege for unlimited times as long as his attribute set satisfies the access policy of a given ciphertext. Such powerful access rights, which is provided by CP-ABE, may be undesirable in real-world applications (e.g., pay-as-you-use). To address the above drawbacks, in this paper, we propose a new notion called *auditable σ-time outsourced CP-ABE*, which is believed to be applicable to cloud computing. In our notion, expensive pairing operation incurred by decryption is offloaded to cloud and meanwhile, the correctness of the operation can be audited efficiently. Moreover, the notion provides *σ-time fine-grained access control*. Cloud service provider may limit a particular set of users to enjoy access privilege for at most σ times within a specified period. As of independent interest, the notion also captures *key-leakage resistance*. The leakage of a user's decryption key does not help a malicious third party in decrypting the ciphertexts belonging to the user. We design a concrete construction (satisfying our notion) in the key encapsulation mechanism setting based on Rouselakis and Waters (prime order) CP-ABE, and further present security and extensive experimental analysis to highlight the scalability and efficiency of our construction.**

***Index Terms*—Outsourced attribute-based encryption, cloud computing, auditability, access control, key-leakage resistance.**

I. Introduction

As a new prevalent commercial paradigm, cloud computing has attracted attention from both academic and industrial communities. Thanks to the advanced development of cloud computing, many enterprises and individuals are allowed to outsource the considerable amount of data to cloud instead of building and maintaining local data centers. The cloud users may also enjoy various types of computing services offered by public cloud. Despite providing long-list advantages, cloud computing may yield the concerns on data security and privacy which hinder its widely use. Attribute-based encryption (ABE) [8] is designed to protect the confidentiality of sensitive data and further to provide fine-grained access control. It may become one of the promising candidates to address the security concern in cloud computing. In particular, ciphertext-policy ABE (CP-ABE) [4] provides a scalable way of encrypting data such that enterprises and individuals can specify access policies over attributes that the potential data users possess. The users are authorized to decrypt respective pieces of outsourced data if their attribute sets satisfy the specified access policies [13], [26], [6]. However, there still exist two major long-lasting problems that hinder the wide-range deployment and commercial applications of CP-ABE to date.

A Motivating Story. Consider a company provides a cloud-based service that enables its registered users to gain access to music and video online and meanwhile, the digital content producers may outsource their encrypted data, to cloud server, under some specified access policies. A service user is assigned with several attributes (e.g., {"level 0", "trial user"}), and he is authorized if his attribute set satisfies the decryption policy of the outsourced digital content. The authorized users, with portable electronic devices (e.g., laptop, mobile phone), are allowed to access the music and video from cloud anytime and anywhere. As a versatile one-to-many encryption mechanism, CP-ABE comes to help here to offer effective and secure access control over the digital content. A more fine-grained access mechanism, nevertheless, may be needed in the sense that the number of access (w.r.t. the authorized users) can be further controlled. For example, the trial users are only allowed to access at most 3 songs (or videos) per week and up to 30 songs (or videos) per week are for the regular users, while VIPs enjoy indefinite access.

A naive solution for the above problem is to allow the server to maintain white and black lists. While an authorized user reaches his access limit, the server may simply move his "id" into the black list. Nonetheless, that is not a privacy-preserving solution. The server may know the "access interest" and even the access rights of the "specified" user. Therefore, *could we design a better solution that guarantees the anonymous limit control while employing CP-ABE?* However, two concerns are immediately raised up here by using CP-ABE.

The paper is accepted by IEEE Transactions on Information Forensics and Security (https://ieeexplore.ieee.org/abstract/document/8007294) and IEEE has the copyright.

J. Ning is with the Department of Computer Science, National University of Singapore, Singapore (e-mail: jtning88@gmail.com).

Z. Cao and X. Dong are with Shanghai Key Lab for Trustworthy Computing, East China Normal University, Shanghai, China (e-mail: {zfcao,dongxiaolei}@sei.ecnu.edu.cn).

K. Liang is with the School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, U.K. (e-mail: K.Liang@mmu.ac.uk).

H. Ma is with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China (e-mail: mahui@iie.ac.cn).

L. Wei is with the College of Information Technology, Shanghai Ocean University, Shanghai, China (e-mail: Lfwei@shou.edu.cn).

Manuscript received; revised.

The expensive decryption cost of CP-ABE, which is a heavy burden for resource-constrained mobile devices, may essentially impede its wide-range deployment (especially in resource-constrained platforms). Most of the existing "pairing-based" CP-ABE systems suffer from *linear decryption cost*. The pairing cost often grows with the increase of access policy, which is a significant challenge for resource-constrained devices. A possible solution to relieve the computational burden is to offload the pairing operations to cloud. This, however, yields the following questions: (1) How to securely offload "a huge part of" decryption to cloud? (2) How to efficiently check the validity of the "partial" decryption result returned by cloud?

The current CP-ABE systems are only designed for "unlimited" access control over encrypted data. That is, if a user is authorized (i.e., his attribute set satisfies the access policy of a ciphertext), he can access the underlying data for unlimited times (from cloud). This "all or nothing" control may restrict its practical use (please see our motivating scenario), specially for the exploration in commercial applications, to a certain extent. But how could we effectively combine limited times and anonymous access control with CP-ABE?

A. Our Contribution

In this paper, we investigate the above interesting problems and further propose a new notion called *auditable σ-time outsourced CP-ABE*. Specifically, we present a basic outsourced CP-ABE system in the key encapsulation mechanism (KEM) setting based on Rouselakis and Waters CP-ABE [23] as the first step. Following the basic construction, we propose a concrete auditable σ-time outsourced CP-ABE system (in the KEM setting). To the best of our knowledge, this is the first of its type that supports the following properties: secure decryption outsourcing, auditability of decryption, limited and anonymous fine-grained access control, key-leakage resistance and light decryption cost on user side. In particular, our contributions are described as follows.

(1) **Secure decryption outsourcing:** In our system, most of the "expensive" operations in decryption, pairings, are offloaded to a cloud, so that a cloud user only needs to perform one exponentiation operation. To achieve secure and efficient decryption outsourcing, we make use of the idea of key blinding. Different from [9], [10], [15], [17], we use both the user public key and the user master secret key initialized by the algorithm $\mathbf{Setup}_U$ to serve as the blinding factors. The computational cost for key blinding merely takes one exponentiation in G, which is much more efficient than that of [9], [10], [15], [17] (note that their cost is linear with the size of attribute set associated with decryption key). After executing a decryption request, the cloud cannot learn any private information from the "partial-decrypted" ciphertext, because the ciphertext is blinded by the user master secret key. The user is then able to recover the plaintext by using his master secret key with the cost of only one exponentiation in G_T.

(2) **Auditability of decryption:** To check if the cloud honestly performs the outsourced decryption, we design an auditable mechanism that enables an auditor to check the returned result. Different from the verifiable methods introduced in [10], [15], [22], [17], our auditable approach does not need to add any additional parameters to the ciphertext of the underlying CP-ABE system (i.e. maintaining the size of ciphertext) and meanwhile, it does not bring any extra computational cost to users (including encryptor and decryptor). It takes only one pairing operation for the auditor to check the correctness of the "partial decryption" made by the cloud. This results in an efficient and practical outsourced CP-ABE system (for resource-constrained devices).

(3) **Limited and anonymous fine-grained access control:** We present a new access control mechanism called *σ-time anonymous fine-grained access control*, which supports more flexibility in access control compared to the conventional CP-ABE. The new mechanism can be seen as a more general version of that of traditional CP-ABE. It additionally provides an efficient way to detect if a user has used up his σ-time access (to encrypted files) in the system. If so, the access privilege of the user is "expired" even if his attribute set satisfies the access policy of the encrypted files. In addition, the access control mechanism is "anonymous" and "access unlinkable" in the sense that the cloud cannot identify who is "under" the current access and meanwhile, the current access cannot be linked back to the previous ones (assuming the label of current access is within $\{1, ..., \sigma\}$). Inspired by [7], [28], we make use of the verifiable random function (VRF) [7], which provides uniqueness and pseudo-randomness properties, to fulfill the design of limited and anonymous access control. Different from [28], our design only takes one exponentiation in G and one exponentiation in G_T (on the user side) to implement the σ-time anonymous control, so that the computational cost is light for the users with resource-constrained devices.

(4) **Key-leakage resistance:** The proposed system is resistant to key leakage attacks. In particular, it employs a sophisticated key generation mechanism to blind the decryption key of a user, so that any third party, compromising the key, cannot effectively recover a valid ciphertext even if the attribute set of the key satisfies the access policy of the ciphertext. Specifically, both the cloud public key and the user public key are involved in the algorithm **KeyGen**. Given a valid leaked decryption key sk, a third party cannot fully decrypt the corresponding ciphertext(s) since the output of the algorithm **Decrypt** is still "masked" by the secret key of the cloud (as well as the user master secret key).

B. Related Work

Attribute-based encryption was first introduced by Sahai and Waters [24]. Goyal *et al.* [8] later formalized two complementary forms of ABE: ciphertext-policy attribute-based encryption (CP-ABE) and key-policy attribute-based encryption (KP-ABE). In a CP-ABE system, a decryption key is associated with a set of attributes and a ciphertext is associated with an access policy defined over attributes. KP-ABE is reversed in that a ciphertext is associated with a set of attributes and a user's decryption key is associated with an access policy. There are many ABE (including CP-ABE and KP-ABE) systems that have been proposed in the literature [4], [11], [27], [12], [19], [20], aiming at achieving the balance of efficiency, expressiveness and security. However, the relatively expensive decryption cost has been known as one of the crucial drawbacks of the standard ABE systems. To address the problem, Green *et al.* [9] introduced the outsourced decryption notion, and further presented concrete ABE systems satisfying the notion. Later, Lai *et al.* [10] introduced the verifiability of ABE to verify the correctness of outsourced decryption. Lin *et al.* [15] revisited the verifiable outsourced ABE [10] and proposed a more efficient construction. In addition to outsourcing decryption, Li *et al.* [13], [14] and Ma *et al.* [17] considered to outsource key-issuing and encryption, respectively. Qin *et al.* [22] proposed a hash-function-based approach to construct ABE with verifiable outsourced decryption. Mao *et al.* [18] later proposed generic constructions of CPA-secure and RCCA-secure ABE systems with verifiable outsourced decryption from CPA-secure ABE (with outsourced decryption). Unfortunately, the aforementioned works fail to support expressive access policy, limited time access control, and the decryption auditing, simultaneously.

II. Background

A. Notation

Define $[l] = \{1, 2, ..., l\}$ for $l \in \mathbb{N}$. Let PPT be probabilistic polynomial-time. We let $(v_1, v_2, ..., v_n)$ be a row vector and $(v_1, v_2, ..., v_n)^\perp$ be a column vector. By v_i we denote the i-th element in a vector $\vec{v}$. And by $M\vec{v}$ we denote the product of matrix M with vector $\vec{v}$. Let $\mathbb{Z}_p^{l\times n}$ be the set of matrices of size $l \times n$ with elements in $\mathbb{Z}_p$. We let $GD = (p, G, G_T, e)$ be the groups and the bilinear mapping description where G and G_T are two multiplicative cyclic groups of prime order p and $e : G \times G \to G_T$ is a bilinear map.

B. Access Structure and Linear Secret-Sharing Schemes

Definition 1. *(Access Structure [2], [21]) : Let S be the attribute universe. A collection (respectively, monotone collection) $\mathbb{A} \subseteq 2^S$ of non-empty sets of attributes is an access structure (respectively, monotone access structure) on S. A collection $\mathbb{A} \subseteq 2^S$ is called monotone if $\forall B, C \in \mathbb{A} : if\ B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.*

Here we restrict our attention to monotone access structure.

Definition 2. *(Linear Secret-Sharing Schemes (LSSS) [2], [21]). Let S denote the attribute universe and p denote a prime. A secret-sharing scheme $\prod$ with domain of secrets $\mathbb{Z}_p$ realizing access structure on S is called linear (over $\mathbb{Z}_p$) if (1) The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$; (2) For each access structure $\mathbb{A}$ on S, there exists a matrix M with l rows and n columns called the share-generating matrix for $\prod$. For $i = 1, ..., l$, we define a function ρ labels row i of M with attribute $\rho(i)$ from the attribute universe S. When we consider the column vector $\vec{v} = (s, r_2, ..., r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, ..., r_n \in \mathbb{Z}_p$ are randomly chosen. Then $M\vec{v} \in \mathbb{Z}_p^{l\times 1}$ is the vector of l shares of the secret s according to $\prod$. The share $(M\vec{v})_j$ "belongs" to attribute $\rho(j)$, where $j \in [l]$.*

As shown in [2], every linear secret-sharing scheme according to the above definition enjoys the linear reconstruction property, defined as follows: suppose that $\prod$ is an LSSS for the access structure $\mathbb{A}$, $S^* \in \mathbb{A}$ is an authorized set and let $I \subset \{1, 2, ..., l\}$ be defined as $I = \{i \in [l] \wedge \rho(i) \in S^*\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that for any valid shares $\{\lambda_i = (M\vec{v})_i\}_{i\in I}$ of a secret s according to $\prod$, $\sum_{i\in I} \omega_i\lambda_i = s$.

C. Prime Order Bilinear Groups.

Let G and G_T be two multiplicative cyclic groups of prime order p. Let g be a generator of G and $e : G \times G \to G_T$ be a bilinear map. The bilinear map e has the following properties: (1) Bilinearity: $\forall u, v \in G$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$; (2) Non-degeneracy: $e(g, g) \neq 1$. We say that G is a bilinear group if the group operations in G and the bilinear map $e : G\times G \to G_T$ can both be computed efficiently. Notice that the map $e(\cdot, \cdot)$ is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$ [21].

D. Complexity Assumption.

The decisional q-bilinear Diffie-Hellman inversion (decisional q-$BDHI$) problem [7] asks, given the tuple $\vec{y} = (g, g^x, ..., g^{x^q})$ as input, if one can distinguish $e(g, g)^{1/x}$ from a random value in G_T.

Formally, an algorithm $\mathcal{A}$ has advantage ϵ in solving the above decisional q-$BDHI$ problem if $|\Pr[\mathcal{A}(\vec{y}, e(g, g)^{1/x}) = 0] - \Pr[\mathcal{A}(\vec{y}, R) = 0]| \geq \epsilon$, where the probability is over the coin tosses of $\mathcal{A}$ and the choice of $x \in \mathbb{Z}_p^*$ and $R \in G_T$. The decisional q-$BDHI$ assumption holds if all PPT algorithms have at most a negligible advantage in solving the decisional q-$BDHI$ problem.

III. Auditable σ-Time Outsourced ABE System

A. System Model

We build our system in the key/data encapsulation mechanism (KEM/DEM) setting [25] where an ABE ciphertext (i.e., the KEM ciphertext) encrypts a symmetric session key and the key is furthered used to "construct" the DEM ciphertext. Fig. 1 shows the overview of our system. There are four main entities, namely cloud server, authority (AUT), data owner and data user. Data owner encrypts data under some predefined

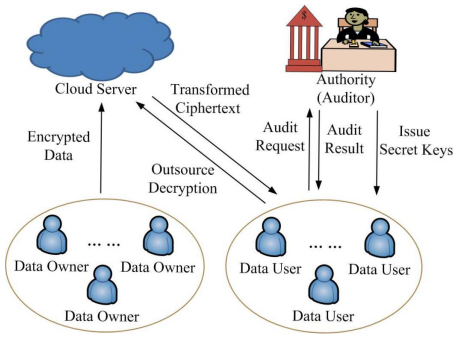


Fig. 1 Overview of system model

access police in the KEM/DEM setting and further outsources the encryption to the cloud server. The cloud server stores all (outsourced) KEM/DEM ciphertexts (generated by data owner). Providing the outsourced decryption service for data user, it returns "partial decrypted" ciphertext, i.e. transformed ciphertext. Note that hereafter we will use the term "transformed ciphertext". AUT generates system parameters and issues decryption key for data user. It is trusted by other entities to execute the audit procedure and return the audit result (to data user). The access privilege of data user is restricted to two conditions, one is that the user must be authorized, and the other is that the access is limited to σ times.

B. Definition

Based on the system model described above, an auditable σ-time outsourced ciphertext-policy attribute-based key encapsulation mechanism (ATOCP-AB-KEM) consists of the following algorithms:

- **Setup**$(\lambda, \mathcal{U}) \rightarrow (pp, msk)$: On input a security parameter λ and the attribute universe description $\mathcal{U}$, the algorithm (run by AUT) outputs the system public parameters pp and the system master secret key msk.
- **Setup**$_C(pp) \rightarrow (pp_c, sk_c)$: On input the system public parameters pp, the algorithm (run by the cloud sever) outputs the cloud public key pp_c and the cloud secret key sk_c.
- **Setup**$_U(pp) \rightarrow (pp_u, sk_u)$: On input the system public parameters pp, the algorithm (run by data user) outputs the user public key pp_u and the user master secret key sk_u.
- **KeyGen**$(pp, msk, pp_c, pp_u, S) \rightarrow sk_S$: On input the system public parameters pp, the system master secret key msk, the cloud public key pp_c, a user public key pp_u and an attribute set S (corresponding to the user), the algorithm (run by AUT) outputs the decryption key sk_S.
- **KeyGen**$_{out}(pp, sk_S, sk_u, csi) \rightarrow tk_S$: On input the system public parameters pp, a decryption key sk_S, a user master secret key sk_u and a current state information csi [1], the algorithm (run by data user) outputs the transformation key tk_S.
- **Encrypt**$(pp, \mathbb{A}) \rightarrow (ct, key)$: On input the system public parameters pp and an access structure $\mathbb{A}$, the algorithm (run by data owner) outputs the ciphertext ct and the encapsulated key key.
- **Decrypt**$_{out}(pp, ct, tk_S) \rightarrow ct'$ or $\perp$: Suppose this is the j-th request of outsourced decryption. On input the system public parameters pp, a ciphertext ct associated with an access structure $\mathbb{A}$ and a transformation key tk_S corresponding to an attribute set S, the algorithm (run by the cloud server) outputs a transformed ciphertext ct' if: (1) S satisfies $\mathbb{A}$; (2) $j \leq \sigma$, where σ (determined by $\mathbb{A}$ or by the role of the transformation key) is the maximum number of the permitted decryption request (within a period). Otherwise, it outputs $\perp$.
- **Decrypt**$_U(ct', sk_u) \rightarrow key$ or $\perp$: On input a transformed ciphertext ct' and a user master secret key sk_u, if **Audit**$(pp_u, ct, ct', msk) \rightarrow 1$, the algorithm (run by data user) outputs the encapsulated key key. Otherwise, it outputs $\perp$.
- **Audit**$(pp_u, ct, ct', msk) \rightarrow 1$ or 0. On input a user public key pp_u, a ciphertext ct, a transformed ciphertext ct' and the system master secret key msk, the algorithm (run by AUT) outputs 1 if the cloud server executes the outsourced decryption correctly. Otherwise, it outputs 0 indicating the server returns an incorrect result.

C. Security

Based on the system model, we consider the two types of adversaries:

(1) We refer to malicious data user as *Type-1 adversary*. For Type-1 adversary, any unauthorized user should not be able to decrypt the encrypted data stored in the cloud server. In particular, the collusion of a set of unauthorized malicious users still cannot obtain the decryption privilege to which has not been granted. Furthermore, any authorized user can only gain access to the outsourced decryption service, provided by the cloud server, within a permitted number of times (in a specified period).

(2) We refer to "honest but curious" cloud server as *Type-2 adversary*. Concretely, the adversary will follow the specification of the protocol, but trying to gather as much secret/private information as possible. Type-2 adversary cannot be allowed to obtain more secret information than what it has already possessed. Furthermore, it cannot identify "who has accessed the encrypted data" and "who has requested the outsourced decryption service". In addition, it cannot link a permitted outsourced decryption request with a former one.

To define the security notion of ATOCP-AB-KEM (satisfying the requirements), we design the following security games.

(1) Replayable Chosen Ciphertext Attack (RCCA) Security.

As of [9], we define RCCA security, which is a relaxation of chosen-ciphertext attacks (CCA), allowing "harmless" modifications to be made to ciphertext [5]. We define the RCCA security of our system according to different types of adversaries as follows.

[1]The current state information is a unique string which is a description of the current state. It may include current time, current IP address and some other information.

RCCA security for Type-1 adversary.

It is defined by a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The security game works as follows:

- **Setup**: $\mathcal{C}$ runs **Setup** and $\textbf{Setup}_C$, and sends (pp, pp_c, sk_c) to $\mathcal{A}$.
- **Query Phase 1**: $\mathcal{C}$ initializes an empty table T, an integer counter $j = 0$ and an empty set D. $\mathcal{A}$ adaptively issues the following queries:
  - **Create**(S): $\mathcal{C}$ sets $j = j + 1$. It runs $\textbf{Setup}_U$ to obtain a user public key pp_u and a user master secret key sk_u, **KeyGen** on S to obtain a decryption key sk_S and $\textbf{KeyGen}_{out}$ on sk_S to obtain a transformation key tk_S, respectively. It then stores the entry $(j, S, sk_S, tk_S, pp_u, sk_u)$ in T.
  - **Corrupt.SK**(i): $\mathcal{C}$ checks whether the i-th entry (i, S, sk_S, pp_u, sk_u) exists in T. If no, it returns $\perp$. Otherwise, it sets $D = D \cup \{S\}$ and returns (sk_S, pp_u, sk_u).
  - **Corrupt.TK**(i): $\mathcal{C}$ checks whether the i-th entry (i, S, tk_S) exists in T. If no, it returns $\perp$. Otherwise, it returns tk_S.
  - **Decrypt**(i, ct): $\mathcal{C}$ checks whether the i-th entry (i, S, sk_S) exists in T. If no, it returns $\perp$. Otherwise, it returns the output of decryption on ct.
- **Challenge**: $\mathcal{A}$ submits a challenge value $\mathbb{A}^*$ with the restriction that for all $S \in D$, S does not satisfy $\mathbb{A}^*$. $\mathcal{C}$ runs **Encrypt** to obtain (ct^*, key^*). It then selects a random bit $b \in \{0, 1\}$. If $b = 0$, it returns (ct^*, key^*) to $\mathcal{A}$. If $b = 1$, it selects a random key R^* in the encapsulated key space and returns (ct^*, R^*).
- **Query Phase 2**: Same as **Query Phase 1** but with the following restrictions: (1) $\mathcal{A}$ cannot trivially obtain a decryption key and the corresponding user master secret key sk_u for the challenge ciphertext. (2) $\mathcal{A}$ cannot issue a decryption query on the challenge ciphertext.
- **Guess**: $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ for b.

Note that the above security game capturers a strong adversarial capability in the sense that $\mathcal{A}$ may obtain the cloud secret key sk_c.

The advantage of $\mathcal{A}$ in this game is defined as $Adv = |\Pr[b' = b] - 1/2|$.

Definition 3. *The ATOCP-AB-KEM is RCCA secure for Type-1 adversary if all PPT Type-1 adversaries have at most a negligible advantage in the above game.*

RCCA security for Type-2 adversary.

The security game is identical to the above one except that the **Setup** is revised as follows:

- **Setup**: $\mathcal{C}$ runs **Setup** and sends pp to $\mathcal{A}$. $\mathcal{A}$ runs $\textbf{Setup}_C$ and sends the cloud public key pp_c to $\mathcal{C}$.

Note that the above security game maintains a strong adversarial capability in the sense that $\mathcal{A}$ may obtain decryption keys and the corresponding user master secret keys of (fully corrupted) data users as long as they are not for the challenge ciphertext.

Definition 4. *The ATOCP-AB-KEM is RCCA secure for Type-2 adversary if all PPT Type-2 adversaries have at most a negligible advantage in the above game.*

CPA Security. We say that an ATOCP-AB-KEM system is CPA-secure (or secure against chosen-plaintext attacks) if $\mathcal{A}$ cannot make decryption queries.

Selective Security. We say that an ATOCP-AB-KEM system is selective secure if we add an **Init** stage before **Setup** where $\mathcal{A}$ outputs the challenge $\mathbb{A}^*$.

(2) Auditability.

Auditability guarantees that AUT can check whether the transformation made by the cloud server is correct. Specifically, given a ciphertext, the malicious cloud server cannot generate two different but "valid" transformed ciphertexts. A transformed ciphertext is valid if it can pass the auditing. The model is defined by a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The security game works as follows:

- **Setup**: $\mathcal{C}$ runs **Setup** to get the system public parameters pp and the system master secret key msk, and sends pp to $\mathcal{A}$. $\mathcal{A}$ runs $\textbf{Setup}_C$ and sends pp_c to $\mathcal{C}$.
- **Query Phase 1**: $\mathcal{A}$ can adaptively issue all kinds of queries as described in the RCCA security game for Type-2 adversary.
- **Challenge**: $\mathcal{A}$ submits a challenge value $\mathbb{A}^*$ to $\mathcal{C}$. $\mathcal{C}$ runs **Encrypt** on $\mathbb{A}^*$ to obtain (ct^*, key^*), and returns ct^*.
- **Query Phase 2**: Same as **Query Phase 1**.
- **Output**: $\mathcal{A}$ outputs an attribute set S^* and a tuple $(ct_1'^*, ct_2'^*)$, where $(ct_1'^*, ct_2'^*)$ are two transformed ciphertexts of ct^*. We assume that the entry $(S^*, sk_{S^*}, tk_{S^*}, pp_u, sk_u)$ exists in T (If not, $\mathcal{C}$ can generate the entry). $\mathcal{A}$ wins the game if $\textbf{Audit}(pp_u, ct^*, ct_1'^*, msk) \rightarrow 1 \wedge \textbf{Audit}(pp_u, ct^*, ct_2'^*, msk) \rightarrow 1 \wedge \textbf{Decrypt}_U(ct_1'^*, sk_u) \neq \textbf{Decrypt}_U(ct_2'^*, sk_u)$.

Definition 5. *The ATOCP-AB-KEM system is auditable if all PPT adversaries have at most a negligible advantage in the above game.*

(3) σ-Time Limitation.

σ-time limitation guarantees that a data user cannot run $\textbf{Decrypt}_{out}$ for $\sigma + 1$ times without being detected if he is only granted to access the outsourced decryption service for at most σ times. The security is defined by a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The security game works as follows:

- **Setup**: $\mathcal{C}$ runs **Setup** and $\textbf{Setup}_C$, and sends (pp, pp_c, sk_c) to $\mathcal{A}$.
- **Query Phase**: $\mathcal{A}$ can adaptively issue all kinds of queries as described in the RCCA security game for Type-1 adversary.
- **Challenge**: Suppose this is the j-th request for outsourced decryption. $\mathcal{A}$ submits a challenge value $\mathbb{A}^*$ to $\mathcal{C}$. $\mathcal{C}$ runs **Encrypt** on $\mathbb{A}^*$ to obtain (ct^*, key^*), and returns ct^* to $\mathcal{A}$. $\mathcal{A}$ then runs $\textbf{Decrypt}_{out}(pp, ct^*, tk_S)$.
- **Output**: $\mathcal{A}$ outputs a transformed ciphertext ct'^* produced by $\textbf{Decrypt}_{out}(pp, ct^*, tk_S)$, where $\textbf{Audit}(pp_u, ct^*, ct'^*, msk) \rightarrow 1$. $\mathcal{A}$ wins if $j = \sigma + 1$.

Definition 6. *The ATOCP-AB-KEM system is σ-time limitable if all PPT adversaries have at most a negligible advantage in the above game.*

(4) Outsourcing Privacy.

Outsourcing privacy guarantees that all permitted outsourced decryption requests from data users are unlinkable and anonymous. Specifically, the cloud server cannot link a permitted outsourced decryption request with a former one. Moreover, the cloud server cannot identify who sends the outsourced decryption request even if there exist two data users who share the same attribute set. The model is defined by a security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The security game works as follows:

- **Setup**: $\mathcal{C}$ runs **Setup** to get the system public parameters pp and the system master secret key msk, and sends (pp, msk) to $\mathcal{A}$. $\mathcal{A}$ runs $\textbf{Setup}_C$ and sends the cloud public key pp_c to $\mathcal{C}$.
- **Query Phase 1**: $\mathcal{A}$ can adaptively issue all kinds of queries as described in the RCCA security game for Type-2 adversary.
- **Challenge**: $\mathcal{A}$ declares two user public keys $pp_{u,0}, pp_{u,1}$ and the corresponding attribute sets S_0, S_1 (with the restriction that $\mathcal{A}$ does not issue **Corrupt.SK**(i) and **Corrupt.TK**(i) queries for S_0 and S_1 in **Query Phase 1**), a value $\mathbb{A}^*$ (where both S_0 and S_1 satisfy $\mathbb{A}^*$) together with the maximum number of outsourced service request, σ (which is determined by $\mathbb{A}^*$ or the role of a transformation key), and sends them to $\mathcal{C}$. $\mathcal{C}$ runs **Encrypt** on $\mathbb{A}^*$ to get $ct_{\mathbb{A}^*}$. It then chooses a random bit $b \in \{0,1\}$, runs **KeyGen** on $(pp_{u,b}, S_b)$ to obtain sk_{S_b}, and $\textbf{KeyGen}_{out}$ on $(sk_{S_b}, sk_{u,b})$ to obtain tk_{S_b}, respectively. $\mathcal{C}$ sends $(tk_{S_b}, ct_{\mathbb{A}^*})$ to $\mathcal{A}$. $\mathcal{A}$ then runs the j-th $\textbf{Decrypt}_{out}$ on $(tk_{S_b}, ct_{\mathbb{A}^*})$ where $j < \sigma$.
- **Query Phase 2** : Same as **Query Phase 1** but with the restriction that $\mathcal{A}$ cannot issue **Corrupt.SK**(i) and **Corrupt.TK**(i) queries for S_0 and S_1.
- **Output**: $\mathcal{A}$ outputs a bit b'.

Note that the above security game is a strong notion in the sense that $\mathcal{A}$ may obtain the system master secret key msk.

The advantage of $\mathcal{A}$ in this game is defined as $Adv = |\Pr[b' = b] - 1/2|$.

Definition 7. *The ATOCP-AB-KEM system is outsourcing private if all PPT adversaries have at most a negligible advantage in the above game.*

IV. Outsourced Ciphertext-Policy Attribute-Based Key Encapsulation Mechanism

We here present a basic outsourced ciphertext-policy attribute-based key encapsulation mechanism (OCP-AB-KEM) based on Rouselakis and Waters CP-ABE [23], and further prove its security. To fulfill key blinding, we design an algorithm $\textbf{Setup}_U$ that generates the user public key and the user master secret key which are later served as the blinding factors. To prevent data user from decrypting their own ciphertexts "trivially", we design an algorithm $\textbf{Setup}_C$ that constructs the cloud public key (and the cloud secret key) which is later embedded into data user's secret key (regarding as "sort of" limitation on user decryption ability). As a result, the secret key of a data user is indeed blinded by the cloud secret key and the user master secret key simultaneously. The outsourced decryption service provided by the cloud server will "help" a data user to remove the "hindrance" from a given ciphertext (to yield the corresponding transformed ciphertext). The resulting ciphertext, however, is masked by the user master secret key (of the user), preventing the server from "seeing" the underlying sensitive information.

A. A Basic Construction

- $\textbf{Setup}(\lambda) \rightarrow (pp, msk)$: The algorithm calls the group generator $\mathcal{G}$ with λ as input and gets the descriptions of the groups and the bilinear mapping $D = (G, G_T, p, e)$, where p is the prime order of the groups G and G_T. The attribute universe is $\mathcal{U} = \mathbb{Z}_p$. It chooses random $g, h, u, v, w \in G$ and $\alpha \in \mathbb{Z}_p$. It sets the system public parameters $pp = (D, g, h, u, v, w, e(g,g)^\alpha)$ and the system master secret key $msk = \alpha$.
- $\textbf{Setup}_C(pp) \rightarrow (pp_c, sk_c)$: The algorithm chooses a random $y_c \in \mathbb{Z}_p$. It then publishes the cloud public key $pp_c = (Y_c = g^{y_c})$ and sets the cloud secret key $sk_c = y_c$.
- $\textbf{Setup}_U(pp) \rightarrow (pp_u, sk_u)$: The algorithm chooses a random $z_u \in \mathbb{Z}_p$. It then publishes the user public key $pp_u = (Z_u = g^{z_u})$ and sets the user master secret key $sk_u = z_u$.
- $\textbf{KeyGen}(pp, msk, pp_c, pp_u, S = \{A_1, A_2, ..., A_k\} \subseteq \mathbb{Z}_p) \rightarrow sk_S$: The algorithm chooses random $\beta, r, \{r_\tau\}_{\tau \in [k]} \in \mathbb{Z}_p$ and computes

$$K_0 = Z_u^\alpha Y_c^\beta w^r, K_1 = g^\beta, K_2 = g^r,$$

$$\{K_{\tau,3} = g^{r_\tau}, K_{\tau,4} = (u^{A_\tau} h)^{r_\tau} v^{-r}\}_{\tau \in [k]}.$$

  It outputs the secret key $sk_S = (S, K_0, K_1, K_2, \{K_{\tau,3}, K_{\tau,4}\}_{\tau \in [k]})$.
- $\textbf{KeyGen}_{out}(sk_S) \rightarrow tk_S$: The algorithm simply sets $tk_S = sk_S$ and returns the transformation key tk_S.
- $\textbf{Encrypt}(pp, (M, \rho)) \rightarrow (ct, key)$: On input the public parameters pp and an LSSS access structure (M, ρ) (where M is an $l \times n$ matrix), the algorithm first chooses a random $s \in \mathbb{Z}_p$. It then picks random $x_2, ..., x_n \in \mathbb{Z}_p$, sets the vector $\vec{x} = (s, x_2, ..., x_n)^T$ (where T denotes the transpose of the matrix) and computes a vector of shares of s as $(\lambda_1, ..., \lambda_l)^T = M\vec{x}$. It then chooses random $\{t_\tau\}_{\tau \in [l]} \in \mathbb{Z}_p$ and computes

$$C_0 = g^s, \{C_{\tau,1} = w^{\lambda_\tau} v^{t_\tau},$$

$$C_{\tau,2} = (u^{\rho(\tau)} h)^{-t_\tau}, C_{\tau,3} = g^{t_\tau}\}_{\tau \in [l]}.$$

  The encapsulated key is set as $key = e(g,g)^{\alpha s}$ and the ciphertext ct is set as

$$((M, \rho), C_0, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau \in [l]}).$$

- $\textbf{Decrypt}_{out}(pp, ct, tk_S) \rightarrow ct'$ or $\perp$: On input $ct = ((M, \rho), C_0, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau \in [l]})$ and $tk_S = (S, K_0, K_1, K_2, \{K_{\tau,3}, K_{\tau,4}\}_{\tau \in [k]})$ for attribute set S, the algorithm outputs $\perp$ if S does not satisfy (M, ρ).

Otherwise, it sets $I = \{i : \rho(i) \in S\}$ and computes the constants $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} w_i \vec{M}_i = (1, 0, ..., 0)$, where $\vec{M}_i$ is the i-th row of the matrix M. It then computes

$$P = \prod_{i\in I} (e(C_{\tau,1}, K_2)e(C_{\tau,2}, K_{\tau,3})e(C_{\tau,3}, K_{\tau,4}))^{w_i},$$

$$C' = \frac{e(C_0, K_0)}{e(C_0, (K_1)^{sk_c}) \cdot P} = e(g, Z_u)^{\alpha s}.$$

It then outputs the transformed ciphertext $ct' = ((M, \rho), C')$.

- **Decrypt**$_U(ct', sk_u) \rightarrow key$: On input a transformed ciphertext $ct' = ((M, \rho), C')$ and the user master secret key $sk_u = z_u$, the algorithm computes

$$key = (C')^{sk_u^{-1}} = (e(g, Z_u)^{\alpha s})^{z_u^{-1}} = e(g, g)^{\alpha s}.$$

B. Security Analysis

Theorem 1. *Assume the CP-ABE system in [23] is selectively CPA-secure, the OCP-AB-KEM system in Section IV-A is selectively CPA-secure with respect to Definition 3 and Definition 4.*

Proof. For simplicity, we denote by Σ_{kem}, Σ_e the OCP-AB-KEM system in Section IV-A and the CP-ABE system in [23], respectively. To prove the security of the OCP-AB-KEM system with respect to Definition 3, we suppose there exists a PPT Type-1 adversary $\mathcal{A}$ with a challenge access structure (M^*, ρ^*) (M^* is an $l \times n$ matrix) that has a non-negligible advantage in selectively breaking Σ_{kem}. We build a PPT simulator algorithm $\mathcal{B}$ that has a non-negligible advantage in selectively breaking Σ_e.

- **Init**: $\mathcal{B}$ gets a challenge access structure (M^*, ρ^*) from $\mathcal{A}$ and sends the received (M^*, ρ^*) to the Σ_e challenger.
- **Setup**: $\mathcal{B}$ receives the public parameters $pp = (D, g, h, u, v, w, e(g, g)^{\alpha})$ from the Σ_e challenger. It chooses a random $y_c \in \mathbb{Z}_p$ and computes $Y_c = g^{y_c}$. The cloud public key is set as $pp_c = Y_c$ and the cloud secret key is set as $sk_c = y_c$. It sends (pp, Y_c, y_c) to $\mathcal{A}$.
- **Query Phase 1**: $\mathcal{B}$ initializes an empty table T and an integer counter $j = 0$. $\mathcal{A}$ adaptively issues the following queries:
  - **Create**(S): When receiving the decryption key query from $\mathcal{A}$ with an attribute set S (with a restriction that S does not satisfy (M^*, ρ^*)), $\mathcal{B}$ sets $j = j + 1$ and sends it to the Σ_e challenger and obtains a secret key $sk'_S = (K'_0, K'_1, \{K'_{\tau,2}, K'_{\tau,3}\}_{\tau\in[|S|]})$. $\mathcal{B}$ then chooses a random $z'_u, \beta' \in \mathbb{Z}_p$ and computes $Z_u = g^{z'_u}$. The user public key is set as $pp_u = Z_u$ and the user master secret key is set as $sk_u = z'_u$. $\mathcal{B}$ computes $sk_S = (K_0 = (K'_0)^{z'_u} Y_c^{\beta'}, K_1 = g^{\beta'}, K_2 = (K'_1)^{z'_u}, \{K_{\tau,3} = (K'_{\tau,2})^{z'_u}, K_{\tau,4} = (K'_{\tau,3})^{z'_u}\}_{\tau\in[|S|]})$. It sets the transformation key $tk_S = sk_S$ and stores the entry $(j, S, sk_S, tk_S, pp_u, sk_u)$ in T.
  - **Corrupt.SK**(i): $\mathcal{B}$ checks whether the i-th entry $(i, S, sk_S, tk_S, pp_u, sk_u)$ exists in T. If no such entry exists, it returns $\perp$. Otherwise, it returns (sk_S, pp_u, sk_u).
  - **Corrupt.TK**(i): $\mathcal{B}$ checks whether the i-th entry (i, S, tk_S) exists in T. If no such entry exists, it returns $\perp$. Otherwise, it returns tk_S.
- **Challenge**: $\mathcal{A}$ declares two equal length messages (m_0, m_1) and sends them to $\mathcal{B}$. $\mathcal{B}$ submits (m_0, m_1) to the Σ_e challenger and obtains a challenge ciphertext $ct^* = ((M^*, \rho^*), C^*, C^*_0, \{C^*_{\tau,1}, C^*_{\tau,2}, C^*_{\tau,3}\}_{\tau\in[l]})$. $\mathcal{B}$ selects a random bit $b_{\mathcal{B}} \in \{0, 1\}$, computes $key_{b_{\mathcal{B}}} = C^*/m_{b_{\mathcal{B}}}$ and returns the new challenge ciphertext $ct^*_{new} = ((M^*, \rho^*), key_{b_{\mathcal{B}}}, C_0 = C^*_0, \{C_{\tau,1} = C^*_{\tau,1}, C_{\tau,2} = C^*_{\tau,2}, C_{\tau,3} = C^*_{\tau,3}\}_{\tau\in[l]})$ to $\mathcal{A}$.
- **Query Phase 2**: Same as **Query Phase 1**.
- **Guess**: $\mathcal{A}$ outputs a guess $b_{\mathcal{A}} \in \{0, 1\}$. If $b_{\mathcal{A}} = 1$, it indicates that $\mathcal{A}$ guesses that $key_{b_{\mathcal{B}}}$ is a random key, $\mathcal{B}$ outputs $1 - b_{\mathcal{B}}$. If $b_{\mathcal{A}} = 0$, it indicates that $\mathcal{A}$ guesses that $key_{b_{\mathcal{B}}}$ is the key encapsulated by ct^*_{new}, $\mathcal{B}$ outputs $b_{\mathcal{B}}$.

Since the distributions of the public parameters, decryption keys and challenge ciphertext in the above game are the same as that of the real system, if $\mathcal{A}$ can selectively break Σ_{kem} with a non-negligible advantage, $\mathcal{B}$ can selectively break Σ_e with the same advantage.

The security proof of the OCP-AB-KEM system with respect to Definition 4 (i.e., for Type-2 adversary) is the same as the above proof except that the adversary runs **Setup**$_C$ instead of $\mathcal{C}$ in the **Setup** phase. □

V. Auditable σ-Time Outsourced Ciphertext-Policy Attribute-Based Key Encapsulation Mechanism

Based on the OCP-AB-KEM system, in this section, we present our auditable σ-time outsourced ciphertext-policy attribute-based key encapsulation mechanism (ATOCP-AB-KEM). We realize the mechanism of limited access control by modifying the algorithms **KeyGen**$_{out}$ and **Decrypt**$_{out}$. Specifically, the transformation key generated by the modified **KeyGen**$_{out}$ consists of the decryption key sk_S as well as some additional elements K_c, K_p, csi for limitation checking, where csi is a current state information, K_c is the VRF [7] output of csi and K_p is the corresponding proof of correctness. During the decryption outsourcing phase, given a transformation key $tk_S = (sk_S, K_c, K_p, csi)$, the algorithm **Decrypt**$_{out}$ will first make some checking on K_c, K_p, csi and the current outsourced decryption service counter ctr. If tk_S passes the check, the algorithm will further execute the outsourced decryption procedure and increment the counter ctr by one. This mechanism ensures the access control to be limited within a specified number.

A. Construction

The algorithms **Setup**$_U$, **KeyGen**, **Encrypt** are the same as those of the OCP-AB-KEM system, the remaining algorithms are modified as follows.

- **Setup**$(\lambda) \rightarrow (pp, msk)$: It operates exactly as in the OCP-AB-KEM system besides that it picks a collision

resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, computes $E = e(g,g)$, and the public parameters pp contain two additional elements: (H, E).

- **Setup**$_C(pp) \rightarrow (pp_c, sk_c)$: It operates exactly as in the OCP-AB-KEM system besides that it initializes an outsourced decryption service counter $ctr = 0$ and an empty set ST for each (potential) transformation key [2]. The cloud server also utilizes a list L to maintain the ctr and ST for each (potential) transformation key.
- **KeyGen**$_{out}(pp, sk_S, sk_u, csi) \rightarrow tk_S$: On input the system public parameters pp, a secret key sk_S, a user master secret key $sk_u = z_u$ and a current state information csi, the algorithm computes $K_c = E^{1/(z_u+H(csi))}, K_p = g^{1/(z_u+H(csi))}$. It then returns $tk_S = (sk_S, K_c, K_p, csi)$.
- **Decrypt**$_{out}(pp, ct, tk_S) \rightarrow ct'$ or $\perp$: On input $ct = ((M,\rho), C_0, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau\in[l]})$ and $tk_S = (S, K_0, K_1, K_2, \{K_{\tau,3}, K_{\tau,4}\}_{\tau\in[k]}, K_c, K_p, csi)$ for attribute set S, the algorithm outputs $\perp$ if S does not satisfy (M,ρ). Otherwise, it gets the tuple (ctr, ST) from L corresponding to tk_S and works as follows:
  (1) It checks whether the following conditions hold:
    1) $e(g^{H(csi)} \cdot Z_u, K_p) = E$ and $K_c = e(g, K_p)$;
    2) $ctr + 1 \leq \sigma$, where σ is the maximum number of the outsourced decryption service request (which is determined by the access structure (M,ρ) on the ciphertext or the role of a transformation key);
    3) $K_c \notin ST$.

    If no, it outputs $\perp$. Otherwise, go to (2).
  (2) It updates $crt \leftarrow crt + 1$ and stores K_c in ST for future use. It then computes the transformed ciphertext $ct' = ((M,\rho), C')$ exactly as in the OCP-AB-KEM system, where $C' = e(g, Z_u)^{\alpha s}$.
- **Decrypt**$_U(ct', sk_u) \rightarrow key$ or $\perp$: On input a transformed ciphertext $ct' = ((M,\rho), C')$ and a user master secret key $sk_u = z_u$, if **Audit**$(pp_u, ct, ct', msk) \rightarrow 0$, the algorithm outputs $\perp$. Otherwise, it computes

$$key = (C')^{sk_u^{-1}} = (e(g, Z_u)^{\alpha s})^{z_u^{-1}} = e(g,g)^{\alpha s}.$$

- **Audit**$(pp_u, ct, ct', msk) \rightarrow 1$ or 0: On input a user public key $pp_u = Z_u$, a ciphertext $ct = ((M,\rho), C_0, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau\in[l]})$, a transformed ciphertext $ct' = ((M,\rho), C')$ and the system master secret key $msk = \alpha$, the algorithm checks whether the following equation holds or not: $e(Z_u^{\alpha}, C_0) = C'$. If no, it outputs 0 indicating that the cloud server executes the outsourced decryption incorrectly. Otherwise, it outputs 1 indicating that the outsourced decryption is correct.

[2]In practice, ctr will be set to 0 at the beginning of a time period (e.g., Monday of a week). Besides, the cloud server may "empty" the set ST at the beginning of the period to save some storage space. The server, however, will maintain distinct ctr and ST for each (potential) transformation key. Each pair of (ctr, ST) is identified by the first element (i.e., sk_S) of a transformation key tuple (sk_S, K_c, K_p, csi). The anonymity can be achieved by running an interactive zero-knowledge proof of knowledge of the discrete log [3] of K_1, K_2 (from sk_S) with respect to g with the cloud server. Note that a transformation key does not reveal the identity of the corresponding key holder. Therefore, the server cannot identify the one "who has accessed the outsourced decryption service". Since the server is assumed to be "honest but curious", for simplicity, we assume the server will follow the specification of the protocol and will not modify the value of ctr arbitrarily. In practice, ctr could be "visible" to user (as well as auditor) who can check if ctr is compromised by the server. This can be achieved by leveraging a commitment scheme to ensure the counter will not be changed later.

B. Security Analysis

Theorem 2. *Assume the CP-ABE system in [23] is selectively CPA-secure, the ATOCP-AB-KEM system in Section V-A is selectively CPA-secure with respect to Definition 3 and Definition 4.*

Proof. The proof technique is identical to that of Theorem 1. □

Theorem 3. *The advantage of an adversary in the auditability security game (in Section III-C) is negligible for our ATOCP-AB-KEM system.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the auditability of the ATOCP-AB-KEM system, it interacts with a simulator $\mathcal{B}$ as follows.

- **Setup**: $\mathcal{B}$ chooses random $g, h, u, v, w \in G$ and $\alpha \in \mathbb{Z}_p$, picks a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and computes $E = e(g,g)$, sets $pp = (D, g, h, u, v, w, e(g,g)^{\alpha}, H, E)$ and sends pp to $\mathcal{A}$. $\mathcal{A}$ chooses a random $y_c \in \mathbb{Z}_p$. It then publishes the cloud public key $pp_c = (Y_c = g^{y_c})$.
- **Query Phase 1**: $\mathcal{A}$ adaptively issues **Corrupt.SK**(i) and **Corrupt.TK**(i) queries. Since $\mathcal{B}$ knows the master secret key msk, it can answer $\mathcal{A}$'s queries properly.
- **Challenge**: $\mathcal{A}$ declares an access structure $\mathbb{A}^*$ and sends it to $\mathcal{B}$. $\mathcal{B}$ generates a ciphertext $ct^* = (\mathbb{A}^*, C_0 = g^s, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau\in[l]})$ for $\mathcal{A}$.
- **Query Phase 2**: Same as **Query Phase 1**.
- **Output**: $\mathcal{A}$ outputs two transformed ciphertexts $(ct_1'^*, ct_2'^*)$ of ct^*.
  $\mathcal{A}$ wins if the following conditions are all fulfilled.
  (1) **Audit**$(pp_u, ct^*, ct_1'^*, msk) \rightarrow 1$;
  (2) **Audit**$(pp_u, ct^*, ct_2'^*, msk) \rightarrow 1$;
  (3) **Decrypt**$_U(ct_1'^*, sk_u) \neq$ **Decrypt**$_U(ct_2'^*, sk_u)$.
  The condition (1) implies $e(Z_u^{\alpha}, g^s) = ct_1'^*$, while the condition (2) implies $e(Z_u^{\alpha}, g^s) = ct_2'^*$. From conditions (1) and (2), we have $e(Z_u^{\alpha}, g^s) = ct_1'^* = ct_2'^*$. However, the condition (3) implies that $ct_1'^* \neq ct_2'^*$, contradicting to the above fact $e(Z_u^{\alpha}, g^s) = ct_1'^* = ct_2'^*$ implied by conditions (1) and (2). Therefore, $\mathcal{A}$ wins the game only with negligible probability.

□

Theorem 4. *The advantage of an adversary in the σ-time limitation security game (in Section III-C) is negligible for our ATOCP-AB-KEM system, assume the decisional q-$BDHI$ assumption holds, where q is the number of* **Decrypt**$_{out}$ *query.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the σ-time limitation security of the ATOCP-AB-KEM system, it interacts with a simulator $\mathcal{B}$ as follows.

- **Setup**: $\mathcal{B}$ chooses random $g, h, u, v, w \in G$ and $\alpha \in \mathbb{Z}_p$, picks a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, computes $E = e(g,g)$, and sets $pp =$

$(D, g, h, u, v, w, e(g,g)^{\alpha}, H, E)$. It then chooses a random $y_c \in \mathbb{Z}_p$ and computes $Y_c = g^{y_c}$. The cloud public key is set as $pp_c = Y_c$ and the cloud secret key is set as $sk_c = y_c$. It sends (pp, Y_c, y_c) to $\mathcal{A}$.

- **Query Phase**: $\mathcal{A}$ adaptively issues **Corrupt.SK**(i) and **Corrupt.TK**(i) queries. Since $\mathcal{B}$ knows the master secret key msk, it can answer $\mathcal{A}$'s queries properly.
- **Challenge**: Suppose this is the j-th request for outsourced decryption. $\mathcal{A}$ declares an access structure $\mathbb{A}^*$ and sends it to $\mathcal{B}$. $\mathcal{B}$ generates a ciphertext $ct^* = (\mathbb{A}^*, C_0 = g^s, \{C_{\tau,1}, C_{\tau,2}, C_{\tau,3}\}_{\tau \in [l]})$ for $\mathcal{A}$. $\mathcal{A}$ then runs **Decrypt**$_{out}$ on ct^*.
- **Output**: $\mathcal{A}$ outputs a transformed ciphertext ct' produced by **Decrypt**$_{out}$ on ct^*.

  The adversary who is allowed to run **Decrypt**$_{out}$ for at most σ times wins the game if $j > \sigma$. Without loss of generality, we assume $j = \sigma + 1$. $\mathcal{A}$ wins with $j = \sigma + 1$ implies that the outsourced decryption service counter $ctr = \sigma + 1$ and the set ST stores $\{K_{c,i}\}_{i \in [\sigma+1]}$, where $K_{c,i}$ is used for the i-th outsourced decryption for $\forall i \in [\sigma + 1]$. Note that $K_{c,i} = E^{1/(z_u + H(csi))}$ is actually the output of the VRF [7], and the uniqueness property of VRF (i.e., $K_{c,i}$) holds if the decisional q-$BDHI$ assumption holds. The uniqueness property guarantees that $K_{c,i} \neq K_{c,i'}$ for $\forall i, i' \in [\sigma + 1], i \neq i'$. Since $\mathcal{A}$ is only allowed to run **Decrypt**$_{out}$ for at most σ times, by the pigeonhole principle, it implies that at least two values of $K_{c,i}$ are identical. This contradicts a fact that the cloud server will reject the outsourced decryption request with repeating $K_{c,i}$ value. Therefore, $\mathcal{A}$ wins the game only with negligible probability.

□

Theorem 5. *The advantage of an adversary in the outsourcing privacy security game (in Section III-C) is negligible for our ATOCP-AB-KEM system, assume the decisional q-$BDHI$ assumption holds, where q is the number of* ***Decrypt***$_{out}$ *query.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that attacks the outsourcing privacy security of the ATOCP-AB-KEM system, it interacts with a simulator $\mathcal{B}$ as follows.

- **Setup**: $\mathcal{B}$ chooses random $g, h, u, v, w \in G$ and $\alpha \in \mathbb{Z}_p$, picks a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, computes $E = e(g,g)$, sets $pp = (D, g, h, u, v, w, e(g,g)^{\alpha}, H, E)$ and sends pp to $\mathcal{A}$. $\mathcal{A}$ chooses a random $y_c \in \mathbb{Z}_p$ and publishes the cloud public key $pp_c = (Y_c = g^{y_c})$. In addition, $\mathcal{C}$ receives the public parameter Z_u^* from the VRF challenger [7].
- **Query Phase 1**: $\mathcal{A}$ adaptively issues **Corrupt.SK**(i) and **Corrupt.TK**(i) queries. Since $\mathcal{B}$ knows the master secret key msk, it can answer $\mathcal{A}$'s queries properly.
- **Challenge**: $\mathcal{A}$ chooses two user public keys $pp_{u,0}, pp_{u,1}$ and the corresponding attribute sets S_0, S_1 (with a restriction that $\mathcal{A}$ does not issue **Corrupt.SK**(i) and **Corrupt.TK**(i) queries for S_0 and S_1 in **Query Phase 1**), an access structure $\mathbb{A}^*$ (where both S_0 and S_1 satisfy $\mathbb{A}^*$) together with the maximum number of outsourced service request σ. If neither of $pp_{u,0}, pp_{u,1}$ equals to Z_u^*, $\mathcal{B}$ aborts. Otherwise, without loss of generality, assume $pp_{u,0} = Z_u^*$ and the corresponding current state information is csi^*. $\mathcal{A}$ sends them to $\mathcal{B}$. $\mathcal{B}$ submits $H(csi^*)$ to the VRF challenger [7] and gets (K_c^*, K_p^*), which may be equal to $(e(g,g)^{1/(z_u^* + H(csi^*))}, g^{1/(z_u^* + H(csi^*))})$ or a pair of random elements in G_T and G. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$, runs $sk_{S_b} \leftarrow$ **KeyGen**$(pp, pp_c, pp_{u,b}, S_b)$ and sets $tk_{S_b} = (sk_{S_b}, K_c^*, K_p^*)$. $\mathcal{B}$ runs **Encrypt** on $\mathbb{A}^*$ to obtain ct^* and sends (tk_{S_b}, ct^*) to $\mathcal{A}$. $\mathcal{A}$ then runs **Decrypt**$_{out}$ on $(tk_{I_{key,b}}, ct_{I_{enc}^*})$.
- **Query Phase 2**: Same as **Query Phase 1** with the restriction that $\mathcal{A}$ cannot issue **Corrupt.SK**(i) and **Corrupt.TK**(i) queries for S_0 and S_1.
- **Output**: $\mathcal{A}$ outputs its guess b'. If $\mathcal{A}$ wins, $\mathcal{B}$ outputs its guess that $(K_c^*, K_p^*) = (e(g,g)^{1/(z_u^* + H(csi^*))}, g^{1/(z_u^* + H(csi^*))})$. Otherwise, $\mathcal{B}$ outputs its guess that (K_c^*, K_p^*) is a pair of random elements in G_T and G.

  Note that $\mathcal{A}$ does not know z_u^*, the pair (K_c^*, K_p^*) is the output of the VRF [7], and the pseudo-randomness property of VRF holds if the decisional q-$BDHI$ assumption holds. Therefore, the pseudo-randomness property of (K_c^*, K_p^*) implies that $\mathcal{A}$ wins the game only with negligible probability.

□

VI. Performance Evaluations

A. Theoretical Analysis

We first make a comparison from the theoretical point of view. Table I gives the comparison between our work and several related works in terms of feature and performance. In particular, the computational cost of our work for key blinding is only one exponentiation in G, outperforming [9], [10], [15], [17]. The decryption cost of this work on the user side only requires one exponentiation in G_T, which is much less than that of [10], [15], [17]. The size of ciphertext (input to the decryption algorithm) of this work on the user side is shorter than that of [10], [15], [22], [17] as well. In addition, this work supports σ-time access control (over outsourced decryption service), but only costing one exponentiation in G and one in G_T. Therefore, even users with resource-constrained mobile devices may afford the cost. All these show that our work is more efficient and scalable for real-world applications.

B. Experimental Analysis

To evaluate the practical performance of the proposed systems, we implement the OCP-AB-KEM system and the ATOCP-AB-KEM system within the *Charm* framework [1]. We use 224-bit MNT elliptic curves from Pairing-Based Cryptography library [16], and further run our systems on an Intel Core i5-3320M CPU @2.6 GHz and 4 GB RAM running Ubuntu 14.04 LTS 64-bit and Python 3.4. In a CP-ABE system, the complexity of ciphertext policy impacts computational and communication cost. To illustrate this, we generate ciphertext policies in the form of (S_1 and S_2 ... and S_l) to simulate the worst case, where S_i is an attribute. We

TABLE I Comparison with other related works [1]

	CKB	CSDU	CCDU	VA	LAC
[9]	$(2+\|S\|)E_G$	$\|G_T\|$	E_{G_T}	$\times$	$\times$
[10]	$(2+\|S\|)E_G$	$\|2G_T\|+\|G\|$	$2E_G+2E_{G_T}$	$\surd$	$\times$
[15]	$(2+\|S\|)E_G$	$\|G_T\|+\|G\|+l_{KDF}$	$2E_G+E_{G_T}$	$\surd$	$\times$
[22]	E_G	$\|G_T\|+l_H$	E_{G_T}	$\surd$	$\times$
[17]	$(2+\|S\|)E_G$	$\|G_T\|+\|G\|+l_{KDF}$	$2E_G+E_{G_T}$	$\surd$	$\times$
this work	E_G	$\|G_T\|$	E_{G_T}	$\surd$	$\surd$

[1] CKB stands for the computational cost of key blinding, CSDU stands for the ciphertext size input to the decryption algorithm on the user side, CCDU stands for the computational cost of decryption on the user side, VA stands for verifiability or auditability, and LAC stands for limited (time) access control. Let E_G, E_{G_T} be the maximum amounts of time to compute an exponentiation in G and an exponentiation in G_T, respectively. Let $|S|$ be the size of the attribute set of a private key, l_H be the output size of a CRH function in [22], l_{KDF} be the output size of a KDF in [15], [17].

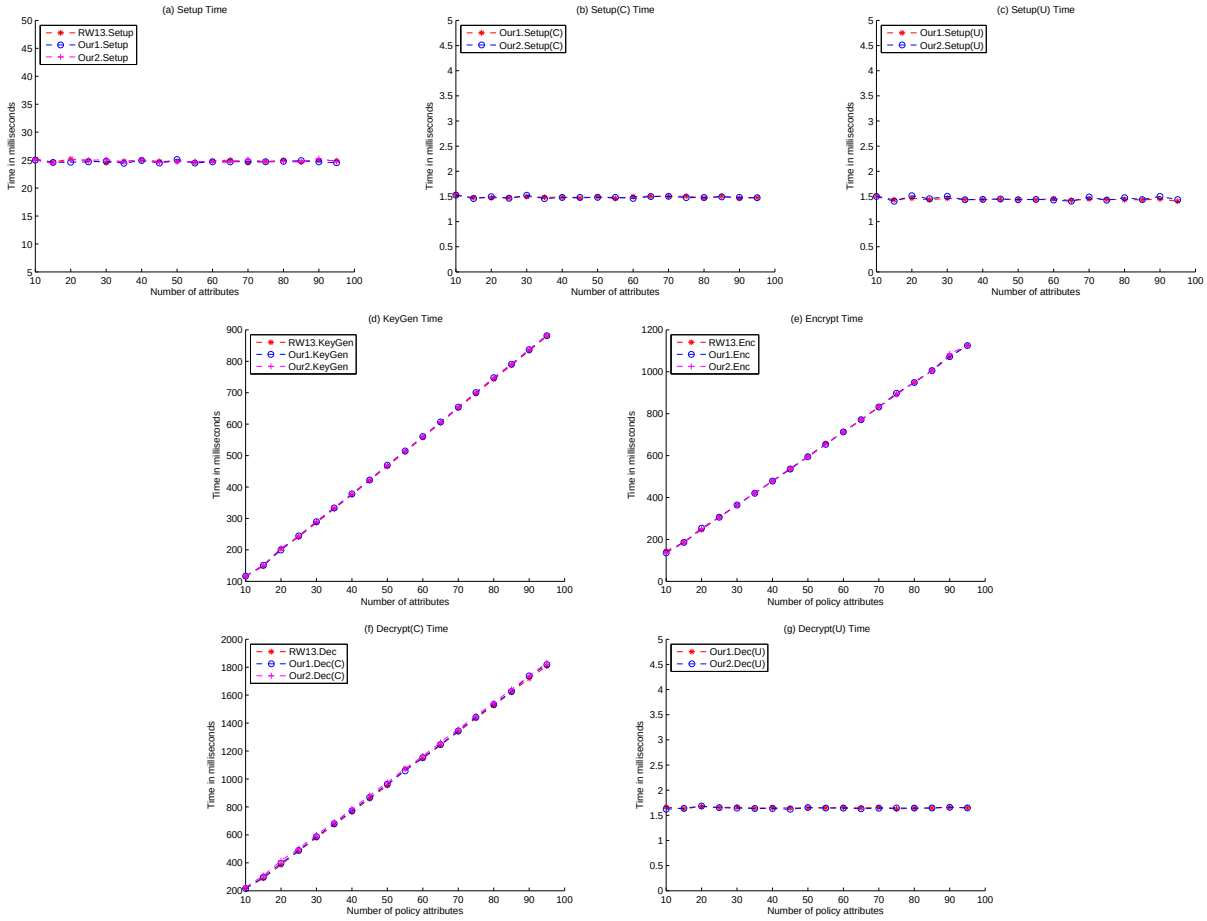


Fig. 2 Experimental results

set 20 distinct access policies in this form with the number of policy attributes increasing from 10 to 100, repeat each instance 10 times and eventually take the average. We keep all the instances completely independent to each other. The time is given in milliseconds.

We aim to evaluate the efficiency of the OCP-AB-KEM system and the ATOCP-AB-KEM system by comparing the total (running) time taken in each stage with the original CP-ABE in [23] (denoted by RW13) which does not consider the auditable outsourcing and σ-time access control functionalities. As depicted in Fig. 2, we examine the time cost of the execution of individual stage. Fig. 2(a), 2(d), 2(e) and 2(f) illustrate that the time cost for Setup, KeyGen, Encrypt and Decrypt phases of our OCP-AB-KEM system and ATOCP-AB-KEM system are almost the same with that of RW13, where Dec(C) stands for the decryption executed on the cloud server side. Fig. 2(b) and 2(c) show that the setup time for the cloud server and data user (only) require 1.49 ms on average. Fig. 2(g) shows the time cost of decryption executed on the user side, which only costs 1.63 ms on average. To sum up, our proposed systems achieve the features of auditable outsourcing and σ-time access control without incurring significant overhead compared to the original CP-ABE in [23]. In particular, the computational cost on the user side is significant lightweight, which makes our systems applicable to resource-constrained applications. We also note that the technique used to fulfill the auditable outsourcing and limited access control is "transplantable" to other CP-ABE.

VII. Conclusion and Future Work

We have addressed two interesting open problems of CP-ABE in cloud-based applications, and have presented an auditable σ-time outsourced CP-ABE system in the KEM setting. In particular, the proposed system is able to offload pairing operations (of the decryption algorithm) to cloud. Furthermore, it enables an auditor to "verify" the correctness of the outsourced decryption. The system also provides a more flexible access control (compared to the conventional CP-ABE) by limiting the access of a particular set of users to maximum σ times (within a period). As of independent interest, the proposed system is also key-leakage resistant.

Our simulation results show that our construction is efficient and practical. Note that the proposed system is *privately auditable*, that is, only the "fully trusted" auditor (e.g., the authority, with some shared secret information) can run the audit procedure. A stronger notion, named *public auditability*, may be desirable in practice, so that any system user is allowed to run the audit procedure to check the correctness of the outsourced decryption. Constructing a σ-time outsourced CP-ABE system with public auditing is an interesting open problem. In addition, in this paper, we mainly utilize the verifiable random function technique to limit the number of access. In our future work, we will also focus on designing more sophisticated solutions to the access privilege control without yielding heavy computational overhead.

VIII. Acknowledgements

We are grateful to the anonymous reviewers for their helpful and valuable comments.

References

[1] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[2] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[3] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology-CRYPTO'92*, pages 390–420. Springer, 1993.

[4] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy, SP'07*, pages 321–334. IEEE, 2007.

[5] Ran Canetti, Hugo Krawczyk, and Jesper B Nielsen. Relaxing chosen-ciphertext security. In *Advances in Cryptology-CRYPTO 2003*, pages 565–582. Springer, 2003.

[6] Hua Deng, Qianhong Wu, Bo Qin, Jian Mao, Xiao Liu, Lei Zhang, and Wenchang Shi. Who is touching my cloud. In *Computer Security - ESORICS 2014*, pages 362–379, 2014.

[7] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography-PKC 2005*, pages 416–431. Springer, 2005.

[8] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. ACM, 2006.

[9] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium*, volume 2011, 2011.

[10] Junzuo Lai, Robert H Deng, Chaowen Guan, and Jian Weng. Attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Information Forensics and Security*, 8(8):1343–1354, 2013.

[11] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology–EUROCRYPT 2010*, pages 62–91. Springer, 2010.

[12] Allison Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Advances in Cryptology–CRYPTO 2012*, pages 180–198. Springer, 2012.

[13] Jin Li, Xiaofeng Chen, Jingwei Li, Chunfu Jia, Jianfeng Ma, and Wenjing Lou. Fine-grained access control system based on outsourced attribute-based encryption. In *Computer Security–ESORICS 2013*, pages 592–609. Springer, 2013.

[14] Jin Li, Xinyi Huang, Jingwei Li, Xiaofeng Chen, and Yang Xiang. Securely outsourcing attribute-based encryption with checkability. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2201–2210, 2014.

[15] Suqing Lin, Rui Zhang, Hui Ma, and Mingsheng Wang. Revisiting attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Information Forensics and Security*, 10(10):2119–2130, 2015.

[16] Ben Lynn et al. The pairing-based cryptography library. *Internet: crypto. stanford. edu/pbc/[Mar. 27, 2013]*, 2006.

[17] Hui Ma, Rui Zhang, Zhiguo Wan, Yao Lu, and Suqing Lin. Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing. *IEEE Transactions on Dependable and Secure Computing*, to appear, 2016.

[18] Xianping Mao, Junzuo Lai, Qixiang Mei, Kefei Chen, and Jian Weng. Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption. *IEEE Transactions on Dependable and Secure Computing*, 13(5):533–546, 2016.

[19] Jianting Ning, Zhenfu Cao, Xiaolei Dong, Lifei Wei, and Xiaodong Lin. Large universe ciphertext-policy attribute-based encryption with white-box traceability. In *Computer Security-ESORICS 2014*, pages 55–72. Springer, 2014.

[20] Jianting Ning, Xiaolei Dong, Zhenfu Cao, and Lifei Wei. Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In *Computer Security–ESORICS 2015*, pages 270–289. Springer, 2015.

[21] Jianting Ning, Xiaolei Dong, Zhenfu Cao, Lifei Wei, and Xiaodong Lin. White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes. *IEEE Transactions on Information Forensics and Security*, 10(6):1274–1288, 2015.

[22] Baodong Qin, Robert H Deng, Shengli Liu, and Siqi Ma. Attribute-based encryption with efficient verifiable outsourced decryption. *IEEE Transactions on Information Forensics and Security*, 10(7):1384–1393, 2015.

[23] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, pages 463–474. ACM, 2013.

[24] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

[25] Victor Shoup. A proposal for an iso standard for public key encryption (version 2.1). *IACR Eprint Archive*, 112, 2001.

[26] Zhiguo Wan, Jun'e Liu, and Robert H Deng. Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing. *IEEE transactions on information forensics and security*, 7(2):743–754, 2012.

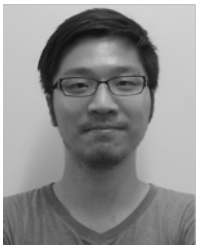
[27] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography–PKC 2011*, pages 53–70. Springer, 2011.

[28] Tsz Hon Yuen, Joseph K Liu, Man Ho Au, Xinyi Huang, Willy Susilo, and Jianying Zhou. k-times attribute-based anonymous access control for cloud computing. *IEEE Transactions on Computers*, 64(9):2595–2608, 2015.

Jianting Ning received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2016. He is currently a research fellow at Department of Computer Science, National University of Singapore. His research interests include applied cryptography and information security, in particular, Public Key Encryption, Attribute-Based Encryption, and Secure Multi-party Computation.

Zhenfu Cao (SM'10) received the B.Sc. degree in computer science and technology and the Ph.D. degree in mathematics from Harbin Institute of Technology, China, in 1983 and 1999, respectively. His research interests include Number Theory, Cryptography and Information Security. Since 1981, more than 400 academic papers have been published in Journals or conferences. He was exceptionally promoted to Associate Professor in 1987, became a Professor in 1991 and is currently a Distinguished Professor in East China Normal University. He serves as a member of the expert panel of the National Nature Science Fund of China. He has received a number of awards, including the Youth Research Fund Award of the Chinese Academy of Science in 1986, the Ying-Tung Fok Young Teacher Award in 1989, the National Outstanding Youth Fund of China in 2002, the Special Allowance by the State Council in 2005, and a corecipient of the 2007 IEEE International Conference on Communications-Computer and Communications Security Symposium Best Paper Award in 2007. He is the leaders of Asia 3 Foresight Program (61161140320) and the key project (61033014) of National Natural Science Foundation of China.

Xiaolei Dong is a Distinguished Professor in East China Normal University. After her graduation with a doctorate degree from Harbin Institute of Technology, she pursued her post-doctoral study in SJTU from September 2001 to July 2003. Then, in August 2003, she joined the Department of Computer Science and Engineering of SJTU. In 2014, she joined East China Normal University. Her primary research interests include Number Theory, Cryptography, Trusted Computing, etc.



Kaitai Liang received the PhD degree from the Department of Computer Science, City University of Hong Kong in 2014. He is an assistant professor with School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, U.K. His research interests are applied cryptography and information security in particular, encryption, network security, big data security, privacy-enhancing technology and security in cloud computing.

Hui Ma received the B.E. degree in information security from the Nanjing University of Aeronautics and Astronautics, China, in 2008. He is a Ph.D. student in information security with the Institute of Information Engineering, Chinese Academy of Sciences. He involves in the security mechanisms in cloud computing.

Lifei Wei received the Ph.D. degree in computer science from Shanghai Jiao Tong University, China in 2013. He is an assistant professor with the Department of Information and Computing Sciences, Shanghai Ocean University. His research interests include applied cryptography, cloud computing, and wireless network security.

View publication stats