Attribute-Based Encryption With Verifiable Outsourced Decryption

Junzuo Lai, Robert H. Deng, Chaowen Guan, and Jian Weng

***Abstract*—Attribute-based encryption (ABE) is a public-key-based one-to-many encryption that allows users to encrypt and decrypt data based on user attributes. A promising application of ABE is flexible access control of encrypted data stored in the cloud, using access polices and ascribed attributes associated with private keys and ciphertexts. One of the main efficiency drawbacks of the existing ABE schemes is that decryption involves expensive pairing operations and the number of such operations grows with the complexity of the access policy. Recently, Green *et al.* proposed an ABE system with outsourced decryption that largely eliminates the decryption overhead for users. In such a system, a user provides an untrusted server, say a cloud service provider, with a transformation key that allows the cloud to translate any ABE ciphertext satisfied by that user's attributes or access policy into a simple ciphertext, and it only incurs a small computational overhead for the user to recover the plaintext from the transformed ciphertext. Security of an ABE system with outsourced decryption ensures that an adversary (including a malicious cloud) will not be able to learn anything about the encrypted message; however, it does not guarantee the correctness of the transformation done by the cloud. In this paper, we consider a new requirement of ABE with outsourced decryption: verifiability. Informally, verifiability guarantees that a user can efficiently check if the transformation is done correctly. We give the formal model of ABE with verifiable outsourced decryption and propose a concrete scheme. We prove that our new scheme is both secure and verifiable, without relying on random oracles. Finally, we show an implementation of our scheme and result of performance measurements, which indicates a significant reduction on computing resources imposed on users.**

***Index Terms*—Attribute-based encryption, outsourced decryption, verifiability.**

I. Introduction

In distributed settings with untrusted servers, such as the cloud, many applications need mechanisms for complex access-control over encrypted data. Sahai and Waters [1] addressed this issue by introducing the notion of attribute-based encryption (ABE). ABE is a new public key based one-to-many encryption that enables access control over encrypted data using access policies and ascribed attributes associated with private keys and ciphertexts. There are two kinds of ABE schemes: key-policy ABE (KP-ABE) [2]–[7] and ciphertext-policy ABE (CP-ABE) [8], [9], [5], [6]. In a CP-ABE scheme, every ciphertext is associated with an access policy on attributes, and every user's private key is associated with a set of attributes. A user is able to decrypt a ciphertext only if the set of attributes associated with the user's private key satisfies the access policy associated with the ciphertext. In a KP-ABE scheme, the roles of an attribute set and an access policy are swapped from what we described for CP-ABE: attributes sets are used to annotate the ciphertexts and access polices over these attributes are associated with users' private keys. In the following, we will use the terms access policy, access structure and access formula interchangeably.

One of the main efficiency drawbacks of the most existing ABE schemes is that decryption is expensive for resource-limited devices due to pairing operations, and the number of pairing operations required to decrypt a ciphertext grows with the complexity of the access policy. At the cost of security, only proven in a weak model (i.e., selective security), there exist several expressive ABE schemes [10], [11] where the decryption algorithm only requires a constant number of pairing computations. Recently, Green *et al.* [12] proposed a remedy to this problem by introducing the notion of ABE with outsourced decryption, which largely eliminates the decryption overhead for users. Based on the existing ABE schemes, Green *et al.* [12] also presented concrete ABE schemes with outsourced decryption. In these schemes (refer to Fig. 1 below), a user provides an untrusted server, say a proxy operated by a cloud service provider, with a transformation key TK that allows the latter to translate any ABE ciphertext CT satisfied by that user's attributes or access policy into a simple ciphertext CT', and it only incurs a small overhead for the user to recover the plaintext from the

Manuscript received January 07, 2013; revised April 09, 2013 and June 10, 2013; accepted June 16, 2013. Date of publication July 03, 2013; date of current version July 10, 2013. This work was supported in part by the Office of Research, Singapore Management University. The work of J. Lai was supported in part by the Fundamental Research Funds for the Central Universities and Natural Science Foundation of China (61272534, 61272453). The work of J. Weng was supported in part by the National Science Foundation of China (61272413, 61005049, 61133014, 61070249, 61272415), in part by the Fok Ying Tung Education Foundation (131066), in part by the Program for New Century Excellent Talents in University (NCET-12-0680), in part by the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security (AGK2011003), in part by the R&D Foundation of Shenzhen Basic Research Project (JC201105170617A), in part by the Guangdong Natural Science Foundation (S2011010001206), in part by the Foundation for Distinguished Young Talents in Higher Education of Guangdong (Grant 2012LYM_0027), in part by the Fundamental Research Funds for the Central Universities, and in part by the A*STAR SERC Grant 102 101 0027 in Singapore. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Michel Abdalla. *(Corresponding author: J. Weng.)*

J. Lai and J. Weng are with the School of Information Systems, Singapore Management University, Singapore 178902, Singapore, and also with the Department of Computer Science, Jinan University, Guangzhou 510632, China (e-mail: junzuolai@smu.edu.sg; jianweng@smu.edu.sg).

R. H. Deng and C. Guan are with the School of Information Systems, Singapore Management University, Singapore 178902, Singapore (e-mail: robertdeng@smu.edu.sg; cwguan@smu.edu.sg).

Digital Object Identifier 10.1109/TIFS.2013.2271848

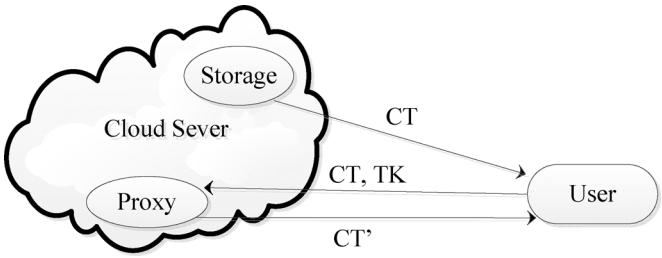

Fig. 1. ABE system with outsourced decryption.

transformed ciphertext CT'. The security property of the ABE scheme with outsourced decryption guarantees that an adversary (including the malicious cloud server) be not able to learn anything about the encrypted message; however, the scheme provides no guarantee on the correctness of the transformation done by the cloud server. In the cloud computing setting, cloud service providers may have strong financial incentives to return incorrect answers, if such answers require less work and are unlikely to be detected by users.

Consider a cloud based electronic medical record system in which patients' medical records are protected using ABE schemes with outsourced decryption (e.g., [12]) and are stored in the cloud. In order to efficiently access patients' medical records on her mobile phone, a doctor generates and delegates a transformation key to a proxy in the cloud for outsourced decryption; Given a transformed ciphertext from the proxy, the doctor can read a patient's medical record by just performing a simple step of computation. If no verification of the correctness of the transformation is guaranteed, however, the system might run into the following two problems: 1) for the purpose of saving computing cost, the proxy could return a medical record transformed previously for the same doctor; 2) due to system malfunction or malicious attack, the proxy could send the medical record of another patient or a file of the correct form but carrying wrong information. The consequence of treating the patient based on incorrect information could be very serious or even catastrophic.

The above observation motivates us to study ABE with verifiable outsourced decryption in this paper. We emphasize that an ABE scheme with secure outsourced decryption does not necessarily guarantee verifiability (i.e., correctness of the transformation done by the cloud server). For example, the secure ABE schemes with outsourced decryption proposed by Green *et al.* in [12] are not verifiable, as we will show in Section III.

A. Our Contributions

In [12], Green *et al.* also considered the verifiability of the cloud's transformation and provided a method to check the correctness of the transformation. However, the authors did not formally define verifiability. In fact, as we will demonstrate in Section III, it is not feasible to construct ABE schemes with verifiable outsourced decryption following the model defined in [12]. Moreover, the method proposed in [12] relies on random oracles (RO) [13]. Unfortunately, the RO model is heuristic, and a proof of security in the RO model does not directly imply anything about the security of an ABE scheme in the real world. It is well known that there exist cryptographic schemes which are secure in the RO model but are inherently insecure when the RO is instantiated with any real hash function [14]–[17].

In this paper, we first modify the original model of ABE with outsourced decryption in [12] to allow for verifiability of the transformations. After describing the formal definition of verifiability, we propose a new ABE model and based on this new model construct a concrete ABE scheme with verifiable outsourced decryption. Our scheme does not rely on random oracles.

During the rest of the paper, we only focus on CP-ABE with verifiable outsourced decryption. The same approach applies to KP-ABE with verifiable outsourced decryption, which we will omit here in order to keep the paper compact.

To assess the performance of our ABE scheme with verifiable outsourced decryption, we implement the CP-ABE scheme with verifiable outsourced decryption and conduct experiments on both an ARM-based mobile device and an Intel-core personal computer to model a mobile user and a proxy, respectively. Our software is based on the CP-ABE implementation in the libfenc [18] library. Through the experiments, we find that it takes almost 50 seconds for the mobile device to execute a standard decryption on ABE ciphertext with policy consisting of 100 attributes. On the other hand, the Intel processor takes less than 5 seconds to decrypt the same ABE ciphertext. With the outsourced decryption, we shift this burdensome task from the mobile device to the proxy, which results in a significant reduction on computing cost for the mobile device. As a consequence, decrypting the ciphertext took approximately 180 milliseconds on the ARM-based device.

B. Related Work

In this subsection, we review some closely related works, including noninteractive verifiable computation, pairing delegation and proxy reencryption.

Noninteractive Verifiable Computation: Noninteractive verifiable computation [19], [20] enables a computationally weak client to outsource the computation of a function to one or more workers. The workers return the result of the function evaluation as well as a noninteractive proof that the computation of the function was carried out correctly. Since these schemes [19], [20] deal with outsourcing of general computation problems and preserve the privacy of input data, they can be used to outsource decryption in ABE systems. However, the schemes proposed in [19], [20] use Gentry's fully homomorphic encryption system [21] as a building block, and thus the overhead in these schemes is currently too large to be practical [22]. Recently, Parno *et al.* [23] establish an important connection between verifiable computation and ABE. They show how to construct a verifiable computation scheme with public delegation and public verifiability from any ABE scheme and how to construct a multifunction verifiable computation scheme from the ABE scheme with outsourced decryption presented in [12]. Goldwasser *et al.* [24] propose a succinct functional encryption scheme for general functions, and show that, by replacing the ABE scheme used in [23] with their succinct functional encryption scheme, one can obtain a delegation scheme with is both publicly verifiable and secret, in the sense that the prover does not learn anything about the input or output of the function being delegated. All

these schemes [19], [20], [23], [24] focus on delegating general functions, and are not sufficiently efficient for the problem at hand.

Pairing Delegation: Pairing delegation [25], [26] enables a client to outsource the computation of parings to another entity. However, the schemes proposed in [25], [26] still require the client to compute multiple exponentiations in the target group for every pairing it outsources. Most importantly, when using paring delegation in the decryption of ABE ciphertexts, the amount of computation of the client is still proportional to the size of the access policy. Tsang *et al.* [27] consider batch pairing delegation. However, the scheme proposed in [27] can only handle batch delegation for pairings in which one of the points is a constant and it still requires the client to compute a pairing.

Proxy Reencryption: In ABE with outsourced decryption, a user provides the cloud with a transformation key that allows the cloud to translate an ABE ciphertext on message m into a simple ciphertext on the same m, without learning anything about m. This is reminiscent of the concept of proxy reencryption [28], [29]. Proxy reencryption allows a proxy, using a reencryption key, to transform an encryption of m under Alice's public key into an encryption of the same m under Bob's public key without the proxy learning anything about the encrypted message m. We emphasize that in the model of proxy reencryption, verifiability of the proxy's transformation cannot be achieved. This can be briefly explained as follows. A proxy could replace the encryption of m under Alice's public key with the encryption of another message m' under Alice's public key and then use its reencryption key to transform the latter into an encryption of m' under Bob's public key. Obviously, without interaction with Alice, Bob cannot detect this malicious behavior of the proxy.

C. Organization

The rest of the paper is organized as follows. In Section II, we review some standard notations and cryptographic definitions. In Section III, we describe our new model of CP-ABE with verifiable outsourced decryption. We present a new CP-ABE scheme with verifiable outsourced decryption and its security analysis in Section IV. Section V shows the experimental results on the performance of our proposed scheme. Finally, we state our conclusion in Section VI.

II. Preliminaries

If S is a set, then $s \xleftarrow{\$} S$ denotes the operation of picking an element s uniformly at random from S. Let $\mathbb{N}$ denote the set of natural numbers. If $\lambda \in \mathbb{N}$ then 1^λ denotes the string of λ ones. Let $z \leftarrow \mathsf{A}(x, y, \ldots)$ denote the operation of running an algorithm A with inputs $(x, y, \ldots)$ and output z. A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists a λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.

A. Bilinear Groups

Let $\mathcal{G}$ be an algorithm that takes as input a security parameter λ and outputs a tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map such that:

1) **Bilinearity:** $e(g^a, h^b) = e(g, h)^{ab}$ for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$.
2) **Nondegeneracy:** $e(g, h) \neq 1$ whenever $g, h \neq 1_{\mathbb{G}}$.
3) **Computable:** efficient computability for any input pair.

We refer to the tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$ as a *bilinear group*.

B. Complexity Assumptions

We state below the complexity assumption to be used in the paper.

a) Discrete Logarithm (DL) Assumption: Let $(p, \mathbb{G}, \mathbb{G}_T, e)$ be a prime order bilinear group system. Given $(p, \mathbb{G}, \mathbb{G}_T, e,\ g, g^x)$, where $g \in \mathbb{G}$ and $x \in \mathbb{Z}_p^*$ are chosen uniformly at random, the DL problem in $(p, \mathbb{G}, \mathbb{G}_T, e)$ is to compute x. The DL assumption in the prime order bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e)$ is that no probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ can solve the DL problem with nonnegligible advantage. The advantage of $\mathcal{A}$ is defined as

$$\Pr[\mathcal{A}(p, \mathbb{G}, \mathbb{G}_T, e,\ g, g^x) = x]$$

where the probability is over the randomly chosen g, x and the random bits consumed by $\mathcal{A}$.

C. Access Structures

Definition 1 (Access Structure [30]): Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}}$ is monotone for $\forall B$ and C, if $B \in \mathbb{A}, B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of nonempty subsets of $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.

In our context, attributes play the role of parties and we restrict our attention to monotone access structures. It is possible to (inefficiently) realize general access structures using our techniques by treating the negation of an attribute as a separate attribute.

D. Linear Secret Sharing Schemes

Our construction will employ linear secret-sharing schemes. We use the definition adapted from [30].

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 for each party form a vector over $\mathbb{Z}_p$.
2) There exists a matrix $\mathbf{A}$ with ℓ rows and n columns called the share-generating matrix for Π. For all $i = 1, \ldots, \ell$, the ith row of $\mathbf{A}$ is labeled by a party $\rho(i)$ (ρ is a function from $\{1, \ldots, \ell\}$ to $\mathcal{P}$). When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $\mathbf{A}v$ is the vector of ℓ shares of the secret s according to Π. The share $(\mathbf{A}v)_i$ belongs to party $\rho(i)$.

It is shown in [30] that every linear secret-sharing scheme according to the above definition also enjoys the linear reconstruction property, defined as follows. Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, \ldots, \ell\}$ be defined as $I = \{i \,|\, \rho(i) \in S\}$. Then there

exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i\in I}\omega_i\lambda_i = s$. Let A_i denotes the ith row of $\mathbb{A}$, we have $\sum_{i\in I}\omega_i A_i = (1, 0, \ldots, 0)$. These constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generation matrix $\mathbf{A}$ [30]. Note that, for unauthorized sets, no such constants $\{\omega_i\}$ exist.

Boolean Formulas Access structures might also be described in terms of monotonic boolean formulas. Using standard techniques one can convert any monotonic boolean formula into an LSSS representation. We can represent the boolean formula as an access tree. An access tree of ℓ nodes will result in an LSSS matrix of ℓ rows. We refer the reader to the Appendix of [31] for a discussion on how to perform this conversion.

E. CP-ABE

A CP-ABE scheme consists of the following four algorithms:

- $\mathsf{Setup}(\lambda, U)$ takes as input a security parameter λ and an attribute universe description U. It outputs the public parameters PK and a master secret key MSK.
- $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ takes as input the public parameters PK, the master secret key MSK and a set of attributes $\mathcal{S}$. It outputs a private key $\mathsf{SK}_{\mathcal{S}}$.
- $\mathsf{Encrypt}(\mathsf{PK}, M, \mathbb{A})$ takes as input the public parameters PK, a message M and an access structure $\mathbb{A}$. It outputs a ciphertext CT.
- $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$ takes as input the public parameters PK, a private key $\mathsf{SK}_{\mathcal{S}}$ for $\mathcal{S}$ and a ciphertext CT. It outputs a message M.

Let $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$, $CT \leftarrow \mathsf{Encrypt}(\mathsf{PK}, M, \mathbb{A})$. For correctness, we require the following to hold:

1) If the set $\mathcal{S}$ of attributes satisfies the access structure $\mathbb{A}$, then $M \leftarrow \mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$;
2) Otherwise, $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$ outputs the error symbol $\perp$.

We now give the definition of indistinguishability under chosen-ciphertext attack (CCA security) for CP-ABE scheme. This is described by a game between a challenger and an adversary $\mathcal{A}$. The game proceeds as follows:

- Setup The challenger runs Setup algorithm to obtain the public parameters PK and a master secret key MSK. It gives the public parameters PK to the adversary $\mathcal{A}$ and keeps MSK to itself.
- $\mathsf{Query\ phase\ 1}$ The challenger initializes an empty set D. The adversary $\mathcal{A}$ adaptively issues queries:
  1) *Private key* query, on input a set of attributes $\mathcal{S}$: The challenger runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ and sets $D = D \cup \{\mathcal{S}\}$. It then returns to the adversary the private key $\mathsf{SK}_{\mathcal{S}}$.
  2) *Decryption* query, on a set of attributes $\mathcal{S}$ and a ciphertext CT: The challenger runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ and $M \leftarrow \mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$. It then returns M to the adversary.
- $\mathsf{Challenge}$ The adversary $\mathcal{A}$ submits two (equal length) messages M_0, M_1 and an access structure $\mathbb{A}^*$, subject to the restriction that, for all $\mathcal{S} \in D$, $\mathbb{A}^*$ cannot be satisfied by $\mathcal{S}$. The challenger selects a random bit $\beta \in \{0, 1\}$, sets $CT^* = \mathsf{Encrypt}(\mathsf{PK}, M_\beta, \mathbb{A}^*)$ and sends CT^* to the adversary as its challenge ciphertext.
- $\mathsf{Query\ phase\ 2}$ The adversary continues to adaptively issue *Private key* and *Decryption* queries, as in Query phase 1, but with the restrictions that the adversary cannot
  1) issue a *Private key* query that would result in a set of attributes $\mathcal{S}$ which satisfies the access structure $\mathbb{A}^*$ being added to D.
  2) issue a *Decryption* query on a set of attributes $\mathcal{S}$ and a ciphertext CT such that $\mathcal{S}$ satisfies $\mathbb{A}^*$ and $CT = CT^*$.
- Guess The adversary $\mathcal{A}$ outputs its guess $\beta' \in \{0, 1\}$ for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $|\Pr[\beta = \beta'] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the adversary.

Definition 3: A CP-ABE scheme is CCA-secure if all polynomial time adversaries have at most a negligible advantage in this security game.

CPA Security: We say that a CP-ABE scheme is *CPA-secure* (or secure against chosen-plaintext attacks) if the adversary cannot make decryption queries.

Selective Security: We say that a CP-ABE scheme is *selectively secure* if we add an **Init** stage before **Setup** where the adversary commits to the challenge access structure $\mathbb{A}^*$.

III. New Model of CP-ABE With Outsourced Decryption

In the original model defined in [12], a CP-ABE scheme with outsourced decryption consists of five algorithms: Setup, $\mathsf{Encrypt}$, KeyGen_{out}, $\mathsf{Transform}$ and $\mathsf{Decrypt}_{out}$. A trusted party uses the algorithm Setup to generate the public parameters and a master secret key, and uses KeyGen_{out} to generate a private key and a transformation key for a user. Taking as input the transformation key given by a user and a ciphertext, the cloud can use the algorithm $\mathsf{Transform}$ to transform the ciphertext into a simple ciphertext if the user's attribute satisfies the access structure associated with the ciphertext; then the user uses the algorithm $\mathsf{Decrypt}_{out}$ to recover the plaintext from the transformed ciphertext. Note that in the definition of Green *et al.* [12], the input to the algorithm $\mathsf{Decrypt}_{out}$ includes only the private key of the user and the transformed ciphertext, but does not include the original ciphertext. Because of this omission of the original ciphertext, it is not possible to construct a CP-ABE scheme with verifiable outsourced decryption under the definition of [12]. This can be explained as follows. A malicious cloud could replace the ciphertext it supposes to transform with a ciphertext of a different message, and then transform the latter into a simple ciphertext using its transformation key. Obviously, the user cannot detect this malicious behavior of the cloud since the input to the algorithm $\mathsf{Decrypt}_{out}$ does not include the original ciphertext required to be transformed. In order to achieve verifiability, we need to modify the model of CP-ABE with outsourced decryption defined in [12]. We now formally describe our new model.

A CP-ABE scheme with outsourced decryption consists of the following seven algorithms:

- $\mathsf{Setup}(\lambda, U)$ takes as input a security parameter λ and attribute universe description U. It outputs the public parameters PK and a master secret key MSK.
- $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ takes as input the public parameters PK, the master secret key MSK and a set of attributes $\mathcal{S}$. It outputs a private key $\mathsf{SK}_{\mathcal{S}}$.
- $\mathsf{Encrypt}(\mathsf{PK}, M, \mathbb{A})$ takes as input the public parameters PK, a message M and an access structure $\mathbb{A}$. It outputs a ciphertext CT.
- $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$ takes as input the public parameters PK, a private key $\mathsf{SK}_{\mathcal{S}}$ for $\mathcal{S}$ and a ciphertext CT. It outputs a message M.
- $\mathsf{GenTK}_{out}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}})$ takes as input the public parameters PK and a private key $\mathsf{SK}_{\mathcal{S}}$ for $\mathcal{S}$. It outputs a transformation key $\mathsf{TK}_{\mathcal{S}}$ and the corresponding retrieving key $\mathsf{RK}_{\mathcal{S}}$.
- $\mathsf{Transform}_{out}(\mathsf{PK}, CT, \mathsf{TK}_{\mathcal{S}})$ takes as input the public parameters PK, a ciphertext CT and a transformation key $\mathsf{TK}_{\mathcal{S}}$ for $\mathcal{S}$. It outputs a partially decrypted ciphertext CT'.
- $\mathsf{Decrypt}_{out}(\mathsf{PK}, CT, CT', \mathsf{RK}_{\mathcal{S}})$ takes as input the public parameters PK, a ciphertext CT, a partially decrypted ciphertext CT' and a retrieving key $\mathsf{RK}_{\mathcal{S}}$ for $\mathcal{S}$. It outputs a message M.

Let $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^{\lambda})$, $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$, $CT \leftarrow \mathsf{Encrypt}(\mathsf{PK}, M, \mathbb{A})$, $(\mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}}) \leftarrow \mathsf{GenTK}_{out}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}})$ and $CT' \leftarrow \mathsf{Transform}_{out}(\mathsf{PK}, CT, \mathsf{TK}_{\mathcal{S}})$. For correctness, we require the following to hold:

1) If the set $\mathcal{S}$ of attributes satisfies the access structure $\mathbb{A}$, then $M \leftarrow \mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$ and $M \leftarrow \mathsf{Decrypt}_{out}(\mathsf{PK}, CT, CT', \mathsf{RK}_{\mathcal{S}})$;
2) Otherwise, $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$ and $\mathsf{Decrypt}_{out}(\mathsf{PK}, CT, CT', \mathsf{RK}_{\mathcal{S}})$ output the error symbol $\perp$.

In our new model, the algorithms Setup, KeyGen, Encrypt and Decrypt constitute a traditional CP-ABE scheme. The input to the algorithm $\mathsf{Decrypt}_{out}$ includes the original ciphertext and the transformed ciphertext. In fact, in our concrete scheme, a user only needs to know a small part of the original ciphertext to verify the correctness of the transformation done by the cloud in the algorithm $\mathsf{Decrypt}_{out}$. In addition, in our model, using the algorithm GenTK_{out} and his private key, the user generates the transformation key by himself, not by the trusted party as in [12]. Having either the trusted party or the user generate the transformation key does not have an effect on the security of the scheme. However, it is more flexible if we let the user himself generate the transformation key. Imagine that a user doesn't know whether he will outsource decryption of his stored files or not in the future. At the setup stage of our proposed ABE with verifiable outsourced decryption system, the user can just initialize an ordinary ABE system without outsourced decryption. Then, the user can generate the transformation key himself whenever he wants to outsource decryption, without having to resetup of the whole system. On the other hand, if the trusted party is responsible for the generation of transformation keys, the user is required to reinitialize the system for outsourced decryption.

Now, we formally describe the security and verifiability requirements of a CP-ABE scheme with outsourced decryption. Informally, security ensures that an adversary (including a malicious cloud) not be able to learn anything about the encrypted message and verifiability allows a user to check on the correctness of the transformation done by the cloud.

Security. Since the traditional notion of security against adaptive chosen-ciphertext attacks (CCA) does not allow any bit of the ciphertext to be altered, similar to [12], we adopt a relaxation due to Canetti *et al.* [32] called replayable CCA (RCCA) security, which allows modifications to the ciphertext provided they cannot change the underlying message in a meaningful way. The RCCA security for CP-ABE with outsourced decryption is described as a game between a challenger and an adversary. The RCCA security game proceeds as follows:

- Setup The challenger runs Setup algorithm to obtain the public parameters PK and a master secret key MSK. It gives the public parameters PK to the adversary $\mathcal{A}$ and keeps MSK to itself.
- Query phase 1 The challenger initializes an empty table T and an empty set D. The adversary $\mathcal{A}$ adaptively issues queries:
  1) *Private key* query, on input a set of attributes $\mathcal{S}$: The challenger runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ and sets $D = D \cup \{\mathcal{S}\}$. It then returns to the adversary the private key $\mathsf{SK}_{\mathcal{S}}$.
  2) *Transformation key* query, on input a set of attributes $\mathcal{S}$: The challenger searches the entry $(\mathcal{S}, \mathsf{SK}_{\mathcal{S}}, \mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}})$ in table T. If such entry exists, it returns the transformation key $\mathsf{TK}_{\mathcal{S}}$. Otherwise, it runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$, $(\mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}}) \leftarrow \mathsf{GenTK}_{out}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}})$ and stores in table T the entry $(\mathcal{S}, \mathsf{SK}_{\mathcal{S}}, \mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}})$. It then returns to the adversary the transformation key $\mathsf{TK}_{\mathcal{S}}$.
     Without of loss of generality, we assume that an adversary do not issue *Transformation key* query on a set of attributes $\mathcal{S}$, if it has already issued a *Private key* query on the same set of attributes $\mathcal{S}$. Since anyone can by himself generate a transformation key for a user using the algorithm GenTK_{out} and the user's private key, our assumption is reasonable.
  3) *Decryption* query, on input a set of attributes $\mathcal{S}$ and a ciphertext CT: The challenger runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ and $M \leftarrow \mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$. It then returns M to the adversary.
  4) $Decryption_{out}$ query, on input a set of attributes $\mathcal{S}$ and a pair of ciphertexts (CT, CT'): The challenger searches the entry $(\mathcal{S}, \mathsf{SK}_{\mathcal{S}}, \mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}})$ in table T. If such entry exists, it runs $M \leftarrow \mathsf{Decrypt}_{out}(\mathsf{PK}, CT, CT', \mathsf{RK}_{\mathcal{S}})$ and returns to the adversary M; otherwise, it returns $\perp$.
- Challenge The adversary $\mathcal{A}$ submits two (equal length) messages M_0, M_1 and an access structure $\mathbb{A}$, subject to the restriction that, for all $\mathcal{S} \in D$, $\mathbb{A}$ cannot be satisfied by $\mathcal{S}$. The challenger selects a random bit $\beta \in \{0, 1\}$, sets $CT^* = \mathsf{Encrypt}(\mathsf{PK}, M_{\beta}, \mathbb{A})$ and sends CT^* to the adversary as its challenge ciphertext.
- Query phase 2 The adversary continues to adaptively issue *Private key, Transformation key, Decryption* and

$Decryption_{out}$ queries, as in Query phase 1, but with the restrictions that the adversary cannot

1) issue a *Private key* query that would result in a set of attributes $\mathcal{S}$ which satisfies the access structure $\mathbb{A}$ being added to D.
2) issue a trivial decryption query. That is, *Decryption* and $Decryption_{out}$ queries will be answered as in Query phase 1, except that if the response would be either M_0 or M_1, then the challenger responds with the error symbol $\perp$.

- Guess The adversary $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$ for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $|\Pr[\beta = \beta'] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the adversary.

Definition 4: A CP-ABE scheme with outsourced decryption is RCCA-secure if all polynomial time adversaries have at most a negligible advantage in this security game.

CPA Security: We say that a CP-ABE scheme with outsourced decryption is *CPA-secure* (or secure against chosen-plaintext attacks) if the adversary cannot make decryption queries.

Selective Security: We say that a CP-ABE scheme with outsourced decryption is *selectively secure* if we add an **Init** stage before **Setup** where the adversary commits to the challenge access structure $\mathbb{A}^*$.

Verifiability. Verifiability of CP-ABE with outsourced decryption is also described by a game between a challenger and an adversary. The game proceeds as follows:

- Setup The challenger runs Setup algorithm to obtain the public parameters PK and a master secret key MSK. It gives the public parameters PK to the adversary $\mathcal{A}$ and keeps MSK to itself.
- Query phase 1 The challenger initializes an empty table T. The adversary $\mathcal{A}$ adaptively issues queries:
  1) *Private key* query, on input a set of attributes $\mathcal{S}$: The challenger runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ and returns to the adversary the private key $\mathsf{SK}_{\mathcal{S}}$.
  2) *Transformation key* query, on input a set of attributes $\mathcal{S}$: The challenger runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$, $(\mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}}) \leftarrow \mathsf{GenTK}_{out}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}})$ and stores in table T the entry $(\mathcal{S}, \mathsf{SK}_{\mathcal{S}}, \mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}})$. It then returns to the adversary the transformation key $\mathsf{TK}_{\mathcal{S}}$.
     Without of loss of generality, we assume that an adversary does not issue *Transformation key* query on a set of attributes $\mathcal{S}$, if it has already issued a *Private key* query on the same set of attributes $\mathcal{S}$. Since anyone can by himself generate a transformation key for a user using the algorithm GenTK_{out} and the user's private key, our assumption is reasonable.
  3) *Decryption* query, on input a set of attributes $\mathcal{S}$ and a ciphertext CT: The challenger runs $\mathsf{SK}_{\mathcal{S}} \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ and $M \leftarrow \mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}}, CT)$. It then returns to the adversary M.
  4) $Decryption_{out}$ query, on input a set of attributes $\mathcal{S}$ and a pair of ciphertexts (CT, CT'): The challenger searches the entry $(\mathcal{S}, \mathsf{SK}_{\mathcal{S}}, \mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}})$ in table T. If such entry exists, it runs $M \leftarrow \mathsf{Decrypt}_{out}(\mathsf{PK}, CT, CT', \mathsf{RK}_{\mathcal{S}})$ and returns M to the adversary; otherwise, it returns $\perp$.
- Challenge The adversary $\mathcal{A}$ submits a message M^* and an access structure $\mathbb{A}$. The challenger sets $CT^* = \mathsf{Encrypt}(\mathsf{PK}, M^*, \mathbb{A})$ and sends CT^* to the adversary.
- Query phase 2 The adversary continues to adaptively issue *Private key, Transformation key, Decryption* and $Decryption_{out}$ queries, as in Query phase 1.
- Output The adversary $\mathcal{A}$ outputs a set of attributes $\mathcal{S}^*$ and a transformed ciphertext $CT^{*\prime}$. We assume that entry $(\mathcal{S}^*, \mathsf{SK}_{\mathcal{S}^*}, \mathsf{TK}_{\mathcal{S}^*}, \mathsf{RK}_{\mathcal{S}^*})$ exists in table T (If not, the challenger can generate the entry as in the response of *Transformation key* query). The adversary wins the game if $\mathsf{Decrypt}_{out}(\mathsf{PK}, CT^*, CT^{*\prime}, \mathsf{RK}_{\mathcal{S}^*}) \notin \{M^*, \perp\}$.

The advantage of the adversary in this game is defined as $\Pr[\mathcal{A}\ wins]$ where the probability is taken over the random bits used by the challenger and the adversary.

Definition 5: A CP-ABE scheme with outsourced decryption is verifiable if all polynomial time adversaries have at most a negligible advantage in the above game.

One stronger notion of verifiability is that, even if the trusted party who setups the system is malicious, a user still can check on the correctness of the transformation done by the cloud. That is, the adversary generates the system's public parameters and master secret key by himself in the above game. Nevertheless, it is difficult to construct a CP-ABE scheme with outsourced decryption which is verifiable in such stronger model, since most existing techniques of provable security need to generate the system's public parameters *elaborately* by the challenger. This challenging problem will be left as one of our future research topics.

IV. Proposed CP-ABE Scheme With Verifiable Outsourced Decryption

In this section, we first propose a new CP-ABE scheme utilizing Waters' CP-ABE scheme [4], which is proven to be selectively CPA-secure. Then, based on the scheme, we propose a CP-ABE scheme with outsourced decryption and prove that it is selectively CPA-secure and verifiable in the standard model. Recently, the first CP-ABE scheme that achieved full security was proposed by Lewko *et al.* [5]. Since the underlying structure of the CP-ABE scheme presented by Lewko *et al.* [5] is almost identical to the underlying Waters' CP-ABE scheme [4] we use, one can adapt our construction techniques to the CP-ABE scheme proposed in [5] to achieve fully secure (i.e., RCCA secure) CP-ABE scheme with verifiable outsourced decryption in the standard model.

A. New CP-ABE Scheme

Before presenting our new CP-ABE scheme, we give some intuitions of our construction. Based on Waters' CP-ABE scheme [4], we add to the ciphertext the encryption of an extra random message and a checksum value, which is computed with this random message and the actual plaintext. We regard this checksum value as a commitment of the actual plaintext, which can be used to check if the transformation is done

correctly in our CP-ABE Scheme with verifiable outsourced decryption. In fact, using our techniques, we can modify unbounded ABE schemes [7], [33] to unbounded ABE scheme with verifiable outsourced decryption.

Our new CP-ABE scheme consists of the following algorithms:

- $\mathsf{Setup}(\lambda, U)$ The setup algorithm takes as input a security parameter λ and a small universe description $U = \{1, 2, \ldots, \ell\}$. It first runs $\mathcal{G}(\lambda)$ to obtain $(p, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p. It then chooses $g, u, v, d \in \mathbb{G}$, and $\alpha, a \in \mathbb{Z}_p^*$ uniformly at random. For each attribute $i \in U$, it chooses a random value $s_i \in \mathbb{Z}_p^*$. Finally, it chooses a collision-resistant hash function $H : \mathbb{G} \to \mathbb{Z}_p^*$. The public parameters are published as $\mathsf{PK} = (\mathbb{G}, \mathbb{G}_T, e, g, u, v, d, g^a, e(g,g)^\alpha, T_i = g^{s_i} \forall i,\ H)$. The master secret key is $\mathsf{MSK} = \alpha$.
- $\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathcal{S})$ The key generation algorithm randomly picks $t \in \mathbb{Z}_p^*$. The secret key $\mathsf{SK}_\mathcal{S} = (\mathcal{S},\ K, K_0, K_i)$ is computed as $K = g^\alpha g^{at},\ K_0 = g^t,\ K_i = T_i^t\ \forall i \in \mathcal{S}$.
- $\mathsf{Encrypt}(\mathsf{PK}, M, \mathbb{A})$ The encryption algorithm takes as input the public parameters PK, a message $M \in \mathbb{G}_T$ to encrypt and an LSSS access structure $\mathbb{A} = (\mathbf{A}, \rho)$, where $\mathbf{A}$ is an $\ell \times n$ matrix and ρ is a map from each row A_i of $\mathbf{A}$ to an attribute $\rho(i)$. It chooses two random vectors $\vec{v}, \vec{v}' \in \mathbb{Z}_p^{*n}$, denoted $\vec{v} = (s, v_2, \ldots, v_n)$ and $\vec{v}' = (s', v_2', \ldots, v_n')$. For each row A_i of $\mathbf{A}$, it chooses $r_{1,i}, r_{2,i} \in \mathbb{Z}_p^*$ uniformly at random. Finally, it chooses a random message $\tilde{M} \in \mathbb{G}_T$. The ciphertext is $CT = ((\mathbf{A}, \rho),\ \hat{C},\ C_1, C_1', C_{1,i}, D_{1,i}, C_2, C_2', C_{2,i}, D_{2,i})$, where

$$\hat{C} = u^{H(M)} v^{H(\tilde{M})} d,$$
$$C_1 = M \cdot e(g,g)^{\alpha s},\ C_1' = g^s,$$
$$C_{1,i} = g^{aA_i \cdot v} T_{\rho(i)}^{-r_{1,i}},\ D_{1,i} = g^{r_{1,i}}\ \forall i \in \{1, 2, \ldots, \ell\},$$
$$C_2 = \tilde{M} \cdot e(g,g)^{\alpha s'},\ C_2' = g^{s'},$$
$$C_{2,i} = g^{aA_i \cdot v'} T_{\rho(i)}^{-r_{2,i}},\ D_{2,i} = g^{r_{2,i}}\ \forall i \in \{1, 2, \ldots, \ell\}.$$

- $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{SK}_\mathcal{S}, CT)$ The decryption algorithm takes as input the public parameters PK, a private key $\mathsf{SK}_\mathcal{S} = (\mathcal{S},\ K,\ K_0, K_i)$ for a set of attributes $\mathcal{S}$ and a ciphertext $CT = ((\mathbf{A}, \rho),\ \hat{C},\ C_1, C_1', C_{1,i}, D_{1,i}, C_2, C_2', C_{2,i}, D_{2,i})$ for an access structure $\mathbb{A} = (\mathbf{A}, \rho)$. If $\mathcal{S}$ does not satisfy the access structure $\mathbb{A}$, it outputs $\perp$. Suppose that $\mathcal{S}$ satisfies the access structure and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in \mathcal{S}\}$. It computes constant $\omega_i \in \mathbb{Z}_p^*$ such that $\sum_{i \in I} \omega_i A_i = (1, 0, \ldots, 0)$. The decryption algorithm then computes:

$$C_1 \cdot \frac{\left(\prod_{i \in I} (e(C_{1,i}, K_0) \cdot e(K_{\rho(i)}, D_{1,i}))^{\omega_i}\right)}{e(C_1', K)}$$
$$= M \cdot e(g,g)^{\alpha s} \cdot \frac{\left(\prod_{i \in I} e(g,g)^{atA_i \cdot v \cdot \omega_i}\right)}{(e(g,g)^{\alpha s} e(g,g)^{ats})} = M,$$
$$C_2 \cdot \frac{\left(\prod_{i \in I} (e(C_{2,i}, K_0) \cdot e(K_{\rho(i)}, D_{2,i}))^{\omega_i}\right)}{e(C_2', K)}$$
$$= \tilde{M} \cdot e(g,g)^{\alpha s'} \cdot \frac{\left(\prod_{i \in I} e(g,g)^{atA_i \cdot v' \cdot \omega_i}\right)}{(e(g,g)^{\alpha s'} e(g,g)^{ats'})} = \tilde{M}.$$

If $\hat{C} = u^{H(M)} v^{H(\tilde{M})} d$, it outputs the message M; otherwise, it outputs $\perp$.

Obviously, the above CP-ABE scheme satisfies correctness. Observe that, in our construction, a ciphertext includes three parts: $(C_1, C_1', C_{1,i}, D_{1,i})$, $(C_2, C_2', C_{2,i}, D_{2,i})$ and $\hat{C}$. The first and second parts are encryptions of message M and a random message $\tilde{M}$, respectively, using the encryption algorithm of Waters' CP-ABE scheme [4]. In fact, the second and third parts are redundant. However, the redundant parts are the point that we can construct a CP-ABE with verifiable outsourced decryption from the above CP-ABE scheme.

Theorem 1: Suppose that the construction of Waters [4] is a selectively CPA-secure CP-ABE scheme, then the above construction of CP-ABE scheme is also selectively CPA-secure.

Proof: To prove the selective CPA security of our CP-ABE scheme, we consider the following two games.

- Game_0 The original selectively CPA-secure game of CP-ABE.
- Game_1 Same as Game_0 except for the way that the challenger generates the challenge ciphertext $CT^* = ((\mathbf{A}, \rho),\ \hat{C},\ C_1, C_1', C_{1,i}, D_{1,i}, C_2, C_2', C_{2,i}, D_{2,i})$, where the challenger picks $\hat{C} \in \mathbb{G}$ randomly and the rest parts of the challenge ciphertext are generated properly as in Game_0.

We prove this theorem by the following two lemmas. Lemma 1 states that Game_0 and Game_1 are indistinguishable; and Lemma 2 states that the advantage of the adversary in Game_1 is negligible. Therefore, we conclude that the advantage of the adversary in Game_0 (i.e., the original selectively CPA-secure game) is negligible. This completes the proof of Theorem 1. ■

Lemma 1: Suppose that the construction of Waters [4] is a selectively CPA-secure CP-ABE scheme, Game_0 and Game_1 are computationally indistinguishable.

Proof: Suppose there exists an adversary $\mathcal{A}$ that can distinguish Game_0 and Game_1 with nonnegligible advantage. We build an algorithm $\mathcal{B}$ that can attack the Waters' CP-ABE scheme [4] in the selectively CPA-secure model with nonnegligible advantage.

Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the selectively CPA-secure game of Waters' CP-ABE scheme. $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

- Init The adversary $\mathcal{A}$ gives $\mathcal{B}$ its challenge access structure $\mathbb{A}^*$. $\mathcal{B}$ sends $\mathbb{A}^*$ to $\mathcal{C}$ as its challenge access structure and is given the public parameters $\mathsf{PK}' = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^a, e(g,g)^\alpha, T_i = g^{s_i} \forall i)$ of the Waters' CP-ABE scheme.
- Setup $\mathcal{B}$ chooses random exponents $x, y, z \in \mathbb{Z}_p^*$ and sets $u = g^x$, $v = g^y$, and $d = g^z$. $\mathcal{B}$ also chooses a collision-resistant hash function $H : \mathbb{G} \to \mathbb{Z}_p^*$. Then, $\mathcal{B}$ sends the public parameters $\mathsf{PK} = (p, \mathbb{G}, \mathbb{G}_T, e,\ g, u, v, d, g^a, e(g,g)^\alpha, T_i = g^{s_i} \forall i,\ H)$ to the adversary $\mathcal{A}$.
- $\mathsf{Query\ phase\ 1}$ When the adversary $\mathcal{A}$ adaptively issues a private key query for a set of attributes $\mathcal{S}_i$, $\mathcal{B}$ calls the key generation oracle of $\mathcal{C}$ on $\mathcal{S}_i$ to obtain the private key $\mathsf{SK}_{\mathcal{S}_i}$. Then, $\mathcal{B}$ returns to $\mathcal{A}$ the private key $\mathsf{SK}_{\mathcal{S}_i}$.
- $\mathsf{Challenge}$ The adversary $\mathcal{A}$ submits two (equal length) messages M_0, M_1. $\mathcal{B}$ chooses a random bit $\beta \in \{0, 1\}$ and two random messages $\tilde{M}_0, \tilde{M}_1 \in \mathbb{G}_T$. Then, $\mathcal{B}$ sends $\tilde{M}_0, \tilde{M}_1$ and $\mathbb{A}^*$ to $\mathcal{C}$. $\mathcal{C}$ selects a random bit $\gamma \in \{0, 1\}$, encrypts the message $\tilde{M}_\gamma$ under PK' and $\mathbb{A}^*$ using the

encryption algorithm of Waters' CP-ABE scheme, and sends the resulting ciphertext $CT^{*\prime}$ to $\mathcal{B}$. $\mathcal{B}$ parses $CT^{*\prime}$ as $(\mathbb{A}^* = (\mathbf{A}, \rho),\ C_2, C_2', C_{2,i}, D_{2,i})$. Then, $\mathcal{B}$ chooses a random vector $\vec{v} \in \mathbb{Z}_p^{*n}$, denoted $\vec{v} = (s, v_2, \ldots, v_n)$. For each row A_i of $\mathbf{A}$, $\mathcal{B}$ chooses $r_{1,i} \in \mathbb{Z}_p^*$ uniformly at random. Finally, $\mathcal{B}$ sets

$$\hat{C} = u^{H(M_\beta)} v^{H(\tilde{M}_\beta)} d,\ C_1 = M_\beta \cdot e(g,g)^{\alpha s},\ C_1' = g^s,$$
$$C_{1,i} = g^{aA_i \cdot v} T_{\rho(i)}^{-r_{1,i}},\ D_{1,i} = g^{r_{1,i}}\ \forall i \in \{1, 2, \ldots, \ell\},$$

and sends $CT^* = ((\mathbf{A}, \rho),\ \hat{C},\ C_1, C_1', C_{1,i}, D_{1,i}, C_2, C_2', C_{2,i}, D_{2,i})$ to $\mathcal{A}$ as its challenge ciphertext.

- Query phase 2 $\mathcal{A}$ continues to adaptively issue private key queries as in Query phase 1, and $\mathcal{B}$ responds the queries as in Query phase 1.
- Guess $\mathcal{A}$ outputs its guess $\beta' \in \{0, 1\}$ for β. $\mathcal{B}$ also outputs β' as its guess for γ.

Observe that, if $\beta = \gamma$, then $\mathcal{B}$ has properly simulated Game_0; otherwise, $\mathcal{B}$ has properly simulated Game_1. Thus, if $\mathcal{A}$ can distinguish Game_0 and Game_1 with nonnegligible advantage, we can build an algorithm $\mathcal{B}$ that attacks the selectively CPA-secure Waters' CP-ABE scheme [4] with nonnegligible advantage. ■

Lemma 2: Suppose that the construction of Waters [4] is a selectively CPA-secure CP-ABE scheme, the advantage of the adversary in Game_1 is negligible.

Proof: Suppose there exists an adversary $\mathcal{A}$ that has a nonnegligible advantage in Game_1. We build an algorithm $\mathcal{B}$ that can attack the Waters' CP-ABE scheme [4] in the selectively CPA-secure model with a nonnegligible advantage.

Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the selectively CPA-secure game of Waters' CP-ABE scheme. $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

- Init The adversary $\mathcal{A}$ gives $\mathcal{B}$ its challenge access structure $\mathbb{A}^*$. $\mathcal{B}$ sends $\mathbb{A}^*$ to $\mathcal{C}$ as its challenge access structure and is given the public parameters $\mathsf{PK}' = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^a, e(g,g)^\alpha, T_i = g^{s_i} \forall i)$ of the Waters' CP-ABE scheme.
- Setup $\mathcal{B}$ chooses random exponents $x, y, z \in \mathbb{Z}_p^*$ and sets $u = g^x$, $v = g^y$, and $d = g^z$. $\mathcal{B}$ also chooses a collision-resistant hash function $H : \mathbb{G} \to \mathbb{Z}_p^*$. Then, $\mathcal{B}$ sends the public parameters $\mathsf{PK} = (p, \mathbb{G}, \mathbb{G}_T, e,\ g, u, v, d, g^a, e(g,g)^\alpha, T_i = g^{s_i} \forall i,\ H)$ to the adversary $\mathcal{A}$.
- Query phase 1 When the adversary $\mathcal{A}$ adaptively issues a private key query for a set of attributes $\mathcal{S}_i$, $\mathcal{B}$ calls the key generation oracle of $\mathcal{C}$ on $\mathcal{S}_i$ to obtain the private key $\mathsf{SK}_{\mathcal{S}_i}$. Then, $\mathcal{B}$ returns to $\mathcal{A}$ the private key $\mathsf{SK}_{\mathcal{S}_i}$.
- Challenge The adversary $\mathcal{A}$ submits two (equal length) messages M_0, M_1. $\mathcal{B}$ sends M_0, M_1 and $\mathbb{A}^*$ to $\mathcal{C}$. $\mathcal{C}$ selects a random bit $\gamma \in \{0, 1\}$, encrypts the message M_γ under PK' and $\mathbb{A}^*$ using the encryption algorithm of Waters' CP-ABE scheme, and sends the resulting ciphertext $CT^{*\prime}$ to $\mathcal{B}$. $\mathcal{B}$ parses $CT^{*\prime}$ as $(\mathbb{A}^* = (\mathbf{A}, \rho),\ C, C_1', C_{1,i}, D_{1,i})$. Then, $\mathcal{B}$ chooses a random message $\tilde{M} \in \mathbb{G}_T$ and a random vector $\vec{v}' \in \mathbb{Z}_p^{*n}$, denoted $\vec{v}' = (s', v_2', \ldots, v_n')$. For each row A_i of $\mathbf{A}$, $\mathcal{B}$ chooses $r_{2,i} \in \mathbb{Z}_p^*$ uniformly at random. Finally, $\mathcal{B}$ chooses a random element $\hat{C}$ in $\mathbb{G}_{p_1}$ and sets

$$C_2 = \tilde{M} \cdot e(g,g)^{\alpha s'},\ C_2' = g^{s'},$$
$$C_{2,i} = g^{aA_i \cdot v'} T_{\rho(i)}^{-r_{2,i}},\ D_{2,i} = g^{r_{2,i}}\ \forall i \in \{1, 2, \ldots, \ell\},$$

and sends $CT^* = ((\mathbf{A}, \rho),\ \hat{C},\ C_1, C_1', C_{1,i}, D_{1,i}, C_2, C_2', C_{2,i}, D_{2,i})$ to $\mathcal{A}$ as its challenge ciphertext.

- Query phase 2 $\mathcal{A}$ continues to adaptively issue private key queries as in Query phase 1, and $\mathcal{B}$ responds the queries as in Query phase 1.
- Guess $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ also takes β' as its output.

Obviously, $\mathcal{B}$ has properly simulated in Game_1. Thus, if $\mathcal{A}$ has a nonnegligible advantage in Game_1, then $\mathcal{B}$ attacks the selectively CPA-secure Waters' CP-ABE scheme [4] with a nonnegligible advantage. ■

B. Our CP-ABE Scheme With Verifiable Outsourced Decryption

For notational purposes in the below, we denote the above CP-ABE scheme as $\mathsf{BasicCP{-}ABE}$. Based on $\mathsf{BasicCP{-}ABE}$, we present a CP-ABE scheme with verifiable outsourced decryption. The Setup, KeyGen, Encrypt and Decrypt algorithms operate exactly as in $\mathsf{BasicCP{-}ABE}$. We now describe the remaining algorithms:

- $\mathsf{GenTK}_{out}(\mathsf{PK}, \mathsf{SK}_{\mathcal{S}})$ This algorithm takes as input the public parameters PK and a private key $\mathsf{SK}_{\mathcal{S}} = (\mathcal{S},\ K,\ K_0, K_i)$ for a set of attributes $\mathcal{S}$. It chooses a random value $z \in \mathbb{Z}_p^*$. Then, it sets the transformation key as $\mathsf{TK}_{\mathcal{S}} = (\mathcal{S},\ K' = K^{1/z}, K_0' = K_0^{1/z}, K_i' = K_i^{1/z})$ and the retrieving key as $\mathsf{RK}_{\mathcal{S}} = z$. Note that, with overwhelming probability, z has multiplicative inverse.
- $\mathsf{Transform}_{out}(\mathsf{PK}, CT, \mathsf{TK}_{\mathcal{S}})$ This algorithm takes as input the public parameters PK, a ciphertext $CT = (\mathbb{A} = (\mathbf{A}, \rho),\ \hat{C},\ C_1, C_1', C_{1,i}, D_{1,i},\ C_2, C_2', C_{2,i}, D_{2,i})$ for an access structure $\mathbb{A} = (\mathbf{A}, \rho)$, and a transformation key $\mathsf{TK}_{\mathcal{S}} = (\mathcal{S},\ K',\ K_0', K_i')$ for a set of attributes $\mathcal{S}$. It then computes:

$$T_1' = \frac{e(C_1', K')}{\left(\prod_{i \in I} \left(e(C_{1,i}, K_0') \cdot e\left(K'_{\rho(i)}, D_{1,i}\right)\right)^{\omega_i}\right)}$$
$$= \frac{e(g,g)^{\alpha s/z} e(g,g)^{ats/z}}{\left(\prod_{i \in I} e(g,g)^{atA_i \cdot v \cdot \omega_i / z}\right)} = e(g,g)^{\alpha s/z},$$

and

$$T_2' = \frac{e(C_2', K')}{\left(\prod_{i \in I} \left(e(C_{2,i}, K_0') \cdot e\left(K'_{\rho(i)}, D_{2,i}\right)\right)^{\omega_i}\right)}$$
$$= \frac{e(g,g)^{\alpha s'/z} e(g,g)^{ats'/z}}{\left(\prod_{i \in I} e(g,g)^{atA_i \cdot v' \cdot \omega_i / z}\right)}$$
$$= e(g,g)^{\alpha s'/z},$$

and outputs the transformed ciphertext as $CT' = (\hat{T} = \hat{C}, T_1 = C_1, T_1', T_2 = C_2, T_2')$.

- $\mathsf{Decrypt}_{out}(\mathsf{PK}, CT, CT', \mathsf{RK}_{\mathcal{S}})$ This algorithm takes as input the public parameters PK, a ciphertext $CT = (\mathbb{A} = (\mathbf{A}, \rho),\ \hat{C},\ C_1, C_1', C_{1,i}, D_{1,i},\ C_2, C_2', C_{2,i}, D_{2,i})$, a transformed ciphertext $CT' = (\hat{T}, T_1, T_1', T_2, T_2')$ and a retrieving key $\mathsf{RK}_{\mathcal{S}} = z$ for a set of attributes $\mathcal{S}$. If $\hat{T} \neq \hat{C}$ or $T_1 \neq C_1$ or $T_2 \neq C_2$, it outputs $\perp$. Then, it computes $M = T_1 / T_1'^z$ and $\tilde{M} = T_2 / T_2'^z$. If $\hat{T} = u^{H(M)} v^{H(\tilde{M})} d$, it outputs the message M; otherwise, it outputs $\perp$.

Obviously, the above CP-ABE scheme with outsourced decryption satisfies correctness. In the above construction, a user runs

the algorithm $\mathsf{Decrypt}_{out}$ to recover the plaintext from the transformed ciphertext and computation cost incurred by the user is about three exponentiations, which is far less than the cost of running the algorithm Decrypt to recover the plaintext from the original ciphertext directly. The input of algorithm $\mathsf{Decrypt}_{out}$ includes the original ciphertext $CT = (\mathbb{A}' = (\mathbf{A}, \rho),\ vk,\ \hat{C} = (\hat{C},\ C_1, C_1', C_{1,i}, D_{1,i},\ C_2, C_2', C_{2,i},\ D_{2,i}),\ \sigma)$ and the transformed ciphertext. In fact, the user only needs to know $\hat{C}$ to verify the correctness of the transformation done by the cloud.

Theorem 2: Assume that $\mathsf{BasicCP-ABE}$ is selectively CPA-secure. Then the above construction of CP-ABE scheme with outsourced decryption is selectively CPA-secure.

Proof: Suppose there exists an adversary $\mathcal{A}$ that can attack the above CP-ABE scheme with outsourced decryption in the selectively CPA-secure model with nonnegligible advantage. We build an algorithm $\mathcal{B}$ that can attack the CP-ABE scheme $\mathsf{BasicCP-ABE}$ in the selectively CPA-secure model with nonnegligible advantage.

Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the selectively CPA-secure game of the CP-ABE scheme $\mathsf{BasicCP-ABE}$. $\mathcal{B}$ runs $\mathcal{A}$ to execute the following steps.

- Init The adversary $\mathcal{A}$ gives $\mathcal{B}$ its challenge access structure $\mathbb{A}^*$. $\mathcal{B}$ sends $\mathbb{A}^*$ to $\mathcal{C}$ as its challenge access structure and is given the public parameters $\mathsf{PK} = (N, \mathbb{G}, \mathbb{G}_T, e, g, u, v, d, g^a, e(g,g)^\alpha, T_i = g^{s_i} \forall i,\ H)$ of $\mathsf{BasicCP-ABE}$.
- Setup $\mathcal{B}$ sends the public parameters PK to the adversary $\mathcal{A}$.
- Query phase 1 $\mathcal{B}$ initializes an empty table T and an empty set D. The adversary $\mathcal{A}$ adaptively issues queries:
  1) *Private key* query for a set of attributes $\mathcal{S}$: $\mathcal{B}$ calls the key generation oracle of $\mathcal{C}$ on $\mathcal{S}$ to obtain the private key $\mathsf{SK}_{\mathcal{S}}$. Then, $\mathcal{B}$ sets $D = D \cup \{\mathcal{S}\}$ and returns to $\mathcal{A}$ the private key $\mathsf{SK}_{\mathcal{S}}$.
  2) *Transformation key* query for a set of attributes $\mathcal{S}$: $\mathcal{B}$ searches the entry $(\mathcal{S}, \mathsf{SK}_{\mathcal{S}}, \mathsf{TK}_{\mathcal{S}}, \mathsf{RK}_{\mathcal{S}})$ in table T. If such entry exists, it returns the transformation key $\mathsf{TK}_{\mathcal{S}}$. Otherwise, $\mathcal{B}$ chooses random exponents $z, t \in \mathbb{Z}_p^*$. Then, $\mathcal{B}$ sets

$$K' = g^z g^{at},\ K_0' = g^t,\ K_i' = T_i^t\ \forall i \in \mathcal{S}.$$

  Finally, $\mathcal{B}$ stores in table T the entry $(\mathcal{S}, *, \mathsf{TK}_{\mathcal{S}} = (\mathcal{S},\ K', K_0', K_i'), z)$ and returns to $\mathcal{A}$ the transformation key $\mathsf{TK}_{\mathcal{S}}$. Note that, $\mathcal{B}$ does not know the actual retrieving key $\mathsf{RK}_{\mathcal{S}} = \alpha/z$.
- Challenge The adversary $\mathcal{A}$ submits two (equal length) messages M_0, M_1 and an access structure $\mathbb{A}$. $\mathcal{B}$ sends M_0, M_1 and $\mathbb{A}$ to $\mathcal{C}$ to obtain the challenge ciphertext CT^*. Then, $\mathcal{B}$ sends CT^* to the adversary $\mathcal{A}$ as its challenge ciphertext.
- Query phase 2 $\mathcal{A}$ continues to adaptively issue private key queries as in Query phase 1, and $\mathcal{B}$ responds the queries as in Query phase 1.
- Guess The adversary $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ also outputs β'.

So, we build an algorithm $\mathcal{B}$ that can attack $\mathsf{BasicCP-ABE}$ in the selectively CPA-secure model with nonnegligible advantage, if $\mathcal{A}$ can attack the above CP-ABE scheme with outsourced decryption in the selectively CPA-secure model with nonnegligible advantage. ■

Theorem 3: Suppose that the DL assumption holds in the prime order bilinear group system. Then the above construction of CP-ABE scheme with outsourced decryption is verifiable.

Proof: Suppose there exists an adversary $\mathcal{A}$ that can attack the verifiability of the above CP-ABE scheme with outsourced decryption with nonnegligible advantage. We build an algorithm $\mathcal{B}$ that can solve the DL problem in the prime order bilinear group system with nonnegligible advantage.

$\mathcal{B}$ is given $(p, \mathbb{G}, \mathbb{G}_T, e,\ g, g^x)$ and runs $\mathcal{A}$ executing the following steps.

- Setup $\mathcal{B}$ chooses $\alpha, a, y, z \in \mathbb{Z}_p^*$ uniformly at random. For each attribute $i \in U$, $\mathcal{B}$ chooses a random value $s_i \in \mathbb{Z}_p^*$. $\mathcal{B}$ also chooses a collision-resistant hash function $H : \mathbb{G} \to \mathbb{Z}_p^*$. Then, $\mathcal{B}$ sets the public parameters as $\mathsf{PK} = (p, \mathbb{G}, \mathbb{G}_T, e, g, u = g^x, v = g^y, d = g^z, g^a, e(g,g)^\alpha, T_i = g^{s_i} \forall i,\ H)$. The master secret key is $\mathsf{MSK} = \alpha$. Finally, $\mathcal{B}$ sends the public parameters PK to the adversary $\mathcal{A}$.
- Query phase 1 The adversary $\mathcal{A}$ adaptively issues *Private key*, *Transformation Key*, *Decryption* and $Decryption_{out}$ queries. Since $\mathcal{B}$ knows the master secret key MSK, it can answers the adversary's queries properly.
- Challenge The adversary $\mathcal{A}$ submits a messages M^* and an access structure $\mathbb{A}$. $\mathcal{B}$ sets $CT^* = \mathsf{Encrypt}\ (\mathsf{PK}, M^*, \mathbb{A})$ and sends CT^* to $\mathcal{A}$. Let $CT^* = (\mathbb{A}' = (\mathbf{A}, \rho),\ vk,\ \hat{C} = (\hat{C},\ C_1, C_1', C_{1,i}, D_{1,i},\ C_2, C_2', C_{2,i},\ D_{2,i}),\ \sigma)$, where $\hat{C} = u^{H(M^*)} v^{H(\tilde{M}^*)} d$ and $\tilde{M}^* \in \mathbb{G}_T$ is chosen by $\mathcal{B}$ randomly.
- Query phase 2 $\mathcal{A}$ continues to adaptively issue private key queries as in Query phase 1, and $\mathcal{B}$ responds the queries as in Query phase 1.
- Output The adversary $\mathcal{A}$ outputs a set of attributes $\mathcal{S}^*$ and a transformed ciphertext $CT^{*\prime} = (\hat{T}, T_1, T_1', T_2, T_2')$.

$\mathcal{B}$ computes $T_1 / T_1'^{z_{\mathcal{S}^*}} = M$ and $T_2 / T_2'^{z_{\mathcal{S}^*}} = \tilde{M}$, where $z_{\mathcal{S}^*}$, known by $\mathcal{B}$, is the retrieving key for the attributes set $\mathcal{S}^*$. If the adversary $\mathcal{A}$ wins the above game, then $\mathcal{B}$ can obtain

$$\begin{aligned} & g^{xH(M^*)+yH(\tilde{M}^*)+z} \\ &= u^{H(M^*)} v^{H(\tilde{M}^*)} d = \hat{C} = \hat{T} = u^{H(M)} v^{H(\tilde{M})} d \\ &= g^{xH(M)+yH(\tilde{M})+z}, \end{aligned}$$

where $M \neq M^*$ and $M^*, \tilde{M}^*, M, \tilde{M}, y, z$ are known by $\mathcal{B}$. Since H is a collision-resistant hash function, with overwhelming probability, $H(M^*)$ is not equal to $H(M)$. Then, $\mathcal{B}$ outputs $x = \frac{y(H(\tilde{M}) - H(\tilde{M}^*))}{H(M^*) - H(M)}$ as the solution of the DL problem $(p, \mathbb{G}, \mathbb{G}_T, p_1, p_2, p_3, g, g^x)$. ■

V. Performance

In order to evaluate the performance of our CP-ABE scheme with verifiable outsourced decryption presented in Section IV, we implement our scheme in software based on the libfenc library [18] and using a 224-bit MNT elliptic curve from the Stanford Pairing-Based Crypto library [34]. Although our implementation based the MNT curve implies the use of asymmetric pairing, only a small change need to be made on our scheme of symmetric setting in the implementation. Specifically, suppose that an asymmetric pairing e takes elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ as inputs. Then, according to the description of our scheme in Section IV, we generate two g's, one from

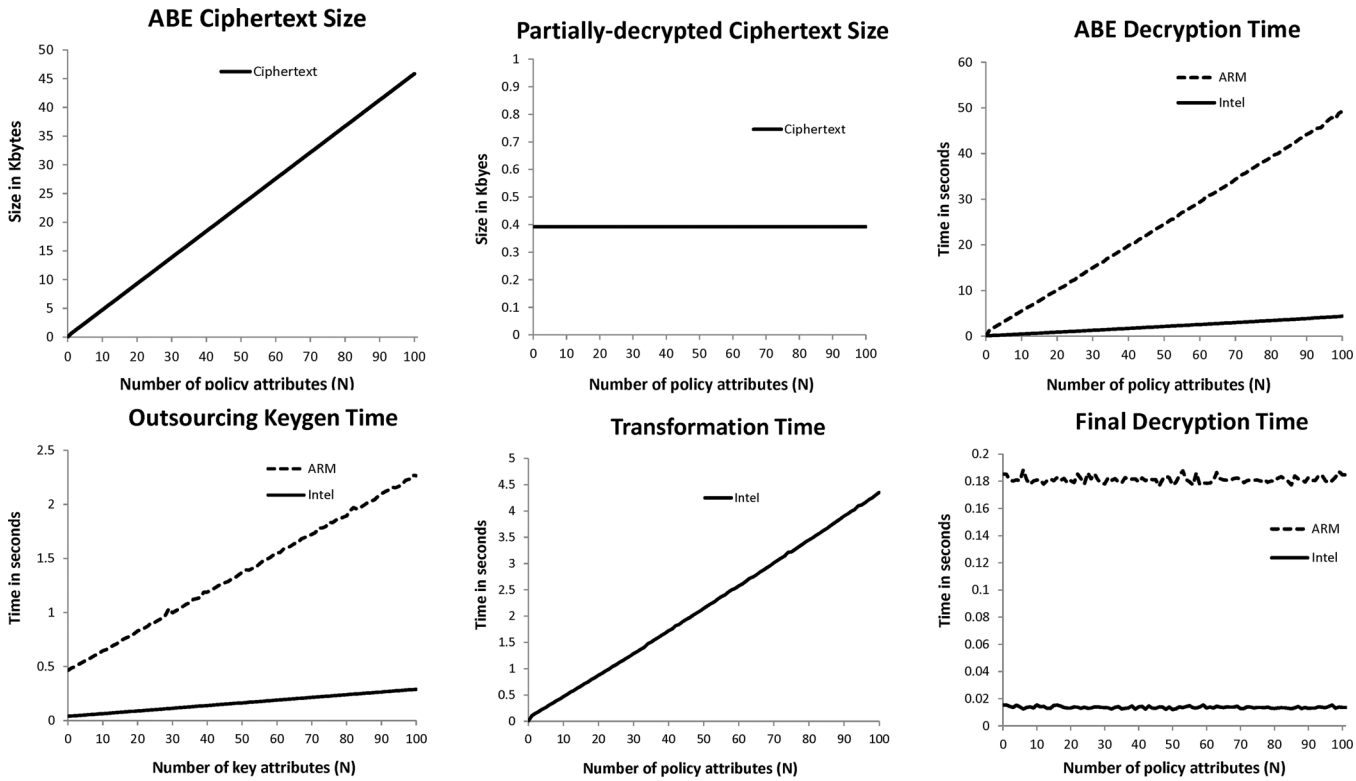

Fig. 2. Performance of our CP-ABE scheme with verifiable outsourced decryption.

$\mathbb{G}_1$ and another from $\mathbb{G}_2$, and compute two corresponding g^a's. We further set u, v, d, T_i as group elements in $\mathbb{G}_1$. As a consequence, among the ciphertext and private key components, $\hat{C}, C'_1, C'_2, C_{1,i}, C_{2,i}, K_i$ are group elements in $\mathbb{G}_1$ while $D_{1,i}, D_{2,i}, K, K_0$ are group elements in $\mathbb{G}_2$. The reason why we implement our proposed scheme using asymmetric pairing is that: compared to symmetric pairings, asymmetric pairings are much faster and more compact to implement [35]–[37]. We compile our code on two dedicated hardware platforms: a 2.53 GHz Intel Core CPU with 4 GB of RAM running 32-bit Linux Kernel version 2.6.32, and a 800 MHz ARM-based Samsung GT-S5830 with 278 MB of RAM running Android OS.

As in [18], our implementation adopts the key encapsulation mechanism, where the ABE ciphertext is the encryption of a symmetric key k and the message is encrypted separately using a symmetric encryption scheme under this k. The symmetric key is computed as $k = e(g, g)^{\alpha s}$, and we omit the components $C_1 = M \cdot e(g, g)^{\alpha s}$ and $C_2 = \hat{M} e(g, g)^{\alpha s'}$ in the ABE ciphertext. Note that in our scheme presented in Section IV, the verification step involves M and $\hat{M}$, while in our implementation, we use the two hash values of $e(g, g)^{\alpha s}$ and $e(g, g)^{\alpha s'}$, instead. These modifications reduce the sizes of the ABE ciphertext and the partially-decrypted ciphertext by two elements in $\mathbb{G}_T$, respectively, without sacrificing security and verifiability.

Experimental Setup: In a CP-ABE scheme, the complexity of ciphertext policy impacts both the decryption time and the ciphertext size. To illustrate this, we generate ciphertext policies in the form of (A_1 and A_2 and ... and A_N) (i.e., the worst situation over the policy), where each A_i is an attribute. This approach ensures that all the ciphertext components are involved the decryption computation. We generate 100 distinct policies in this form with N increasing from 1 to 100. In each case, we construct a corresponding standard decryption key that contains exact N attributes.

In our experiments, we do not consider the effect of symmetric encryption. Thus, all the datum on decryption time and ciphertext size presented in Fig. 2 are only associated with the key encapsulation variant of our ABE scheme. For each ciphertext policy, we repeat our experiment 100 times on the PC and 30 times on the ARM device and we take the average values as the experimental results. In Fig. 2, we show the size of standard ABE ciphertext and partially-decrypted ciphertext, the standard ABE decryption time on the Intel and the ARM platforms, the time of generating an outsourcing key, the time of transforming the ABE ciphertext, and the time of decrypting the transformed ciphertext on the Intel and the ARM platforms.

Discussion: The ABE ciphertext size and decryption/transformation time increase linearly as the ciphertext policy's complexity grows. An encryption under a ciphertext policy with 100 attributes results in an ABE ciphertext of nearly 46 KB and it takes about 5 seconds for the Intel platform to decrypt this ciphertext. On the other hand, decryption time degrades considerably on the ARM platform: it requires more than 1 second to decrypt a ciphertext under a policy with one attribute, 5 seconds under a policy with ten attributes and almost 50 seconds under a policy with one hundred attributes.

As expected, outsourcing substantially reduces the computation time required for devices with limited computing resource to recover the plaintext. The bulk of the decryption operation is now handled by the proxy. The transformed ciphertext is not

only much efficient to decrypt but also much smaller in size. In our implementation, each partially-decrypted ciphertext has a constant size of 392 bytes, regardless the complexity of its corresponding ciphertext policy. The final decryption and verification of the transformed ciphertext requires only 13 milliseconds on the Intel platform and approximately 180 milliseconds on the ARM platform.

VI. Conclusion

In this paper, we considered a new requirement of ABE with outsourced decryption: verifiability. We modified the original model of ABE with outsourced decryption proposed by Green *et al.* [12] to include verifiability. We also proposed a concrete ABE scheme with verifiable outsourced decryption and proved that it is secure and verifiable. Our scheme does not rely on random oracles. To assess the practicability of our scheme, we implemented it and conducted experiments in a simulated outsourcing environment. As expected, the scheme substantially reduced the computation time required for resource-limited devices to recover plaintexts.

References

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM Conf. Computer and Communications Security*, 2006, pp. 89–98.

[3] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. ACM Conf. Computer and Communications Security*, 2007, pp. 195–203.

[4] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Public Key Cryptography*, 2011, pp. 53–70.

[5] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. EUROCRYPT*, 2010, pp. 62–91.

[6] T. Okamoto and K. Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," in *Proc. CRYPTO*, 2010, pp. 191–208.

[7] A. B. Lewko and B. Waters, "Unbounded HIBE and attribute-based encryption," in *Proc. EUROCRYPT*, 2011, pp. 547–567.

[8] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security and Privacy*, 2007, pp. 321–334.

[9] L. Cheung and C. C. Newport, "Provably secure ciphertext policy ABE," in *Proc. ACM Conf. Computer and Communications Security*, 2007, pp. 456–465.

[10] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Ràfols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theor. Comput. Sci.*, vol. 422, pp. 15–38, 2012.

[11] S. Hohenberger and B. Waters, "Attribute-based encryption with fast decryption," in *Proc. Public Key Cryptography*, 2013, pp. 162–179.

[12] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Security Symp.*, San Francisco, CA, USA, 2011.

[13] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. ACM Conf. Computer and Communications Security*, 1993, pp. 62–73.

[14] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited (preliminary version)," in *Proc. STOC*, 1998, pp. 209–218.

[15] J. B. Nielsen, "Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case," in *Proc. CRYPTO*, 2002, pp. 111–126.

[16] S. Goldwasser and Y. T. Kalai, "On the (in)security of the fiat-shamir paradigm," in *Proc. FOCS*, 2003, pp. 102–113.

[17] M. Bellare, A. Boldyreva, and A. Palacio, "An uninstantiable random-oracle-model scheme for a hybrid-encryption problem," in *Proc. EUROCRYPT*, 2004, pp. 171–188.

[18] M. Green, A. Akinyele, and M. Rushanan, Libfenc: The Functional Encryption Library.

[19] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. CRYPTO*, 2010, pp. 465–482.

[20] K.-M. Chung, Y. T. Kalai, and S. P. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. CRYPTO*, 2010, pp. 483–501.

[21] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. STOC*, 2009, pp. 169–178.

[22] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," in *Proc. EUROCRYPT*, 2011, pp. 129–148.

[23] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proc. TCC*, 2012, pp. 422–439.

[24] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Succinct functional encryption and applications: Reusable garbled circuits and beyond," *IACR Cryptology ePrint Archive*, vol. 2012, p. 733, 2012.

[25] B. Chevallier-Mames, J.-S. Coron, N. McCullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Proc. CARDIS*, 2010, pp. 24–35.

[26] B. G. Kang, M. S. Lee, and J. H. Park, "Efficient delegation of pairing computation," *IACR Cryptology ePrint Archive*, vol. 2005, p. 259, 2005.

[27] P. P. Tsang, S. S. M. Chow, and S. W. Smith, "Batch pairing delegation," in *Proc. IWSEC*, 2007, pp. 74–90.

[28] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. EUROCRYPT*, 1998, pp. 127–144.

[29] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *Proc. NDSS*, San Diego, CA, USA, 2005.

[30] A. Beimel, "Secure Schemes for Secret Sharing and Key Distribution," Ph.D. dissertation, Israel Inst. of Technology, Technion City, Haifa, Israel, 1996.

[31] A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. EUROCRYPT*, 2011, pp. 568–588.

[32] R. Canetti, H. Krawczyk, and J. B. Nielsen, "Relaxing chosen-ciphertext security," in *Proc. CRYPTO*, 2003, pp. 565–582.

[33] T. Okamoto and K. Takashima, "Fully secure unbounded inner-product and attribute-based encryption," in *Proc. ASIACRYPT*, 2012, pp. 349–366.

[34] B. Lynn, The Stanford Pairing Based Crypto Library.

[35] S. Chatterjee and A. Menezes, "On cryptographic protocols employing asymmetric pairings—The role of revisited," *Discrete Appl. Math.*, vol. 159, no. 13, pp. 1311–1322, 2011.

[36] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Appl. Math.*, vol. 156, no. 16, pp. 3113–3121, 2008.

[37] N. P. Smart and F. Vercauteren, "On computable isomorphisms in efficient asymmetric pairing-based systems," *Discrete Appl. Math.*, vol. 155, no. 4, pp. 538–547, 2007.

Junzuo Lai received the B.S. and M.S. degrees in computer science and technology from Jingdezhen Ceramic Institute, China, in 2002 and 2005, respectively. In 2010, he received the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University, China.

From August 2008 to February 2010, he was a research assistant with Singapore Management University. Currently, he is a postdoc with Singapore Management University and a research professor with Jinan University, China. His research interests include cryptography and information security.

Robert H. Deng received the Bachelor's degree from National University of Defense Technology, China, and the M.Sc. and Ph.D. degrees from Illinois Institute of Technology, USA.

He has been with the Singapore Management University since 2004, and is currently professor, associate dean for Faculty and Research, School of Information Systems. Prior to this, he was principal scientist and manager of the Infocomm Security Department, Institute for Infocomm Research, Singapore. He has 26 patents and more than 200 technical publications in international conferences and journals in the areas of computer networks, network security, and information security. He has served as general chair, program committee chair, and program committee member of numerous international conferences. He is an associate editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, associate editor of *Security and Communication Networks Journal* (Wiley), and member of Editorial Board of *Journal of Computer Science and Technology* (the Chinese Academy of Sciences). He received the University Outstanding Researcher Award from the National University of Singapore in 1999 and the Lee Kuan Yew Fellow for Research Excellence from the Singapore Management University in 2006.

Chaowen Guan received the B.Eng. degree from Jinan University, China, in 2011.

He was a research engineer with Singapore Management University from June 2012 to June 2013. His primary research interest lies in the field of cryptography and information security.

Jian Weng received the M.S. and B.S. degrees in computer science and engineering from South China University of Technology, in 2004 and 2000, respectively, and the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, in 2008.

From April 2008 to March 2010, he was a postdoc in the School of Information Systems, Singapore Management University. Currently, he is a professor and vice dean with the School of Information Technology, Jinan University. He has published more than 40 papers in cryptography conferences and journals, such as PKC, CT-RSA, ACSAC, SCN, Designs, Codes and Cryptography, Algorithmica, etc. He served as PC cochairs or PC member for more than 10 international conferences, such as ISPEC 2011, RFIDsec 2013 Asia, ISC 2011, IWSEC 2012, etc.