Revisiting Attribute-Based Encryption With Verifiable Outsourced Decryption

Suqing Lin, Rui Zhang, Hui Ma, and Mingsheng Wang

***Abstract*— Attribute-based encryption (ABE) is a promising technique for fine-grained access control of encrypted data in a cloud storage, however, decryption involved in the ABEs is usually too expensive for resource-constrained front-end users, which greatly hinders its practical popularity. In order to reduce the decryption overhead for a user to recover the plaintext, Green *et al.* suggested to outsource the majority of the decryption work without revealing actually data or private keys. To ensure the third-party service honestly computes the outsourced work, Lai *et al.* provided a requirement of verifiability to the decryption of ABE, but their scheme doubled the size of the underlying ABE ciphertext and the computation costs. Roughly speaking, their main idea is to use a parallel encryption technique, while one of the encryption components is used for the verification purpose. Hence, the bandwidth and the computation cost are doubled. In this paper, we investigate the same problem. In particular, we propose a more efficient and generic construction of ABE with verifiable outsourced decryption based on an attribute-based key encapsulation mechanism, a symmetric-key encryption scheme and a commitment scheme. Then, we prove the security and the verification soundness of our constructed ABE scheme in the standard model. Finally, we instantiate our scheme with concrete building blocks. Compared with Lai *et al.*'s scheme, our scheme reduces the bandwidth and the computation costs almost by half.**

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

I. Introduction

With the rapid development of cloud computing, growing data is being centralized into the cloud for sharing. To keep the data security and privacy for data owners, the sharing data needs to be encrypted before being uploaded and fine-grained access control is required. Attribute-based encryption (ABE) [1] was thus proposed to have flexible access control of encrypted data utilizing access policies and ascribed attributes associated with private keys and ciphertexts respectively. In an ABE scheme, a specified private key can decrypt a particular ciphertext only if associated attributes and policy are matched. According to the ciphertext associated with an access policy or containing a set of attributes, ABE schemes are divided into two kinds: ciphertext-policy (CP) ABE [4]–[7] and key-policy (KP) ABE [8], [9].

The functionality of access control is very powerful, however, expensive. For most of the existing pairing-based ABE schemes (see [6], [8]), the number of pairing operations to decrypt a ciphertext is linear to the complexity of the access policy. It would be a significant challenge for users to complete the decryption independently on resource-constrained devices, e.g., mobile phones. In order to reduce the number of pairing operations for users when executing the decryption algorithm, Green *et al.* [2] considered outsourcing the heavy computation of decryption to a third-party service, which helps to implement "thin clients." They proposed a key blinding technique to outsource the decryption without leaking data or secret keys as a precaution against maliciously detecting from the third-party service. A user provides a transformed key to the service to outsource an ABE ciphertext and obtains a constant-size ElGamal-style ciphertext, then utilizes the secret retrieving key to recover the plaintext.

To guarantee the third-party service honestly executes the outsourced computation, Lai *et al.* (LDGW) [3] introduced verifiability to the outsourced decryption of ABE. Actually, they added an extra instance to the underlying ABE scheme [6] in the encryption/decryption algorithms, which is used for verification. The technique added noticeable overhead to the underlying ABE scheme: encryption requires the data sender to encrypt an extra random message and compute a checksum value related to two messages; decryption requires the third-party service to execute the underlying decryption algorithm twice and the data receiver to verify the outsourced computation with respect to the encrypted messages. Although the LDGW-scheme [3] is easy to understand, it works not so well in practice: First, the scheme doubles the computation costs of encryption and decryption compared to the underlying ABE scheme. Second, the length of the ciphertext is twice of that of the underlying ABE ciphertext. Therefore, the following questions arise naturally:

1) Whether there exists a generic construction to introduce verification to the outsourced decryption of ABE?
2) How to construct an ABE with verifiable outsourced decryption more efficiently?

Manuscript received September 22, 2014; revised February 5, 2015 and June 11, 2015; accepted June 11, 2015. Date of publication June 23, 2015; date of current version August 6, 2015. This work was supported in part by the Foundation of Science and Technology on Information Assurance Laboratory Under Grant KJ-14-002, in part by the Strategic Priority Research Program through the Chinese Academy of Sciences, Beijing, China, (CAS), under Grant XDA06010703 and Grant XDA06010701, in part by the One Hundred Talents Project through CAS, in part by the National Natural Science Foundation of China under Grant 61272478, Grant 61472416, Grant 61379142, and Grant 61402468, and in part by the Shenzhen Governmental Research under Grant ZDS Y20130402095348589 and Grant JSGG20130624154032565. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sen-Ching S. Cheung. (*Corresponding author: Rui Zhang.)*

S. Lin and H. Ma are with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: linsuqing@iie.ac.cn; mahui@iie.ac.cn).

R. Zhang and M. Wang are with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: r-zhang@iie.ac.cn; wangmingsheng@iie.ac.cn).

Digital Object Identifier 10.1109/TIFS.2015.2449264

In this paper, we revisit ABE with verifiable outsourced decryption (VO-ABE), and try to solve these problems. We first present a generic construction of VO-ABE, based on an attribute-based key encapsulation mechanism (AB-KEM), a symmetric-key encryption scheme and a commitment scheme. In our opinion, hybrid encryption and a commitment can be used to add verification to the outsourced decryption more efficiently and a proper verification algorithm should be defined as a constraint during the final decryption for the data receiver. Similar to the idea of blinding technique in [2], we propose an appropriate transform for the actual secret key to achieve outsourcing the decryption. In fact, the transform we used here may be thought as a subclass of all-or-nothing transforms (AONTs) [10], [11] with specific properties ensuring secure outsourced computation. We insist that our construction of VO-ABE is comprehensive and can be operated easily and as secure as [3].

A. Our Contributions

In this paper, we propose a novel technique to build an ABE with verifiable outsourced decryption (VO-ABE) based on an AB-KEM, a symmetric-key encryption scheme and a commitment scheme. We provide a unified model of VO-ABE, which can be considered in both key-policy (KP) and ciphertext-policy (CP) settings.

In [3], Lai *et al.* considered to introduce verification to the outsourced decryption of ABE by adding an extra instance in the encryption/decryption algorithms, which duplicates the computation and communication overhead. Instead of two parallel instances in the encryption/decryption algorithms [3], we combine a hybrid encryption and a commitment together to bundle the randomness to the ciphertext, so that one can verify the outsourced computation easily. Decryption is done in the natural way, but note that the outsourced transform key is obtained by an appropriate transform of the actual secret key with specific properties ensuring secure outsourced computation. We define a verification algorithm for the data receiver to check the correctness of the outsourced computation. Only if the verification is passed, the data can be recovered in the decryption algorithm. We prove that our constructed ABE scheme is secure and meets the verification soundness in the standard model if the underlying building blocks are secure.

In addition to the general method discussed above, we also provide an instantiation of a VO-ABE scheme. We implement our CP-ABE scheme with verifiable outsourced decryption and show that the size of the ABE ciphertext and the encryption cost of our scheme are both almost half of the scheme in [3]. On the other hand, since we introduce an additional commitment scheme, the final decryption cost is slightly more than half of that in [3].

B. Related Work

Since the introduction of attribute-based encryption, most of ABE systems are constructed with pairings while the computation cost in the decryption phase grows along with the size of the access policy. Attrapadung *et al.* [12] used novel applications of aggregation techniques to achieve very short ciphertexts so as to control the cost of decryption. Hohenberger and Waters [13] succeeded in reducing the decryption requirements to two pairings and two exponentiations by making tradeoffs in the private key size. From another point of view, Green *et al.* [2] introduced outsourced decryption into ABE systems such that most of complex computation of decryption algorithms is outsourced to an untrusted third-party service, leaving only a smaller overhead for users to recover the plaintext. For ABE systems with outsourced decryption, a third-party service is given a transform key to translate an ABE ciphertext into a constant-size ciphertext on the same message without learning any information about the message. The main effect of outsourcing the decryption of ABE ciphertexts is to delegate pairing operations to a powerful device. Pairing delegation has been proposed in [14]–[16], but applying pairing delegation to the decryption does not overcome the disadvantage of growing computational amount with the complexity of the access policy. Outsourcing decryption is similar to proxy re-encryption [17], [18], where there is no way to verify the proxy's transform. Since the proxy is untrusted, verifiable outsourced computation is needed. Although verifiable computation has been considered in [19] and [20] based on fully homomorphic encryption or ABE schemes, all of them are impractical here. Lai *et al.* [3] provided a feasible method to verify the outsourced decryption and built a concrete ABE scheme with verifiable outsourced decryption. In addition to outsourcing the decryption, Li *et al.* [21], [22] also considered to outsource key-issuing for ABE schemes by introducing two cloud service providers to perform the outsourced key-issuing and decryption.

II. Preliminaries

Notations. Let $\mathbf{Alg}(u, v, \cdots) \rightarrow w$ (sometimes $w \leftarrow \mathbf{Alg}(u, v, \cdots)$) denote the operation of running an algorithm **Alg** with inputs $(u, v, \cdots)$ and output w. If $\mathbf{S}$ is a set, $|\mathbf{S}|$ denotes its size and $b \xleftarrow{R} \mathbf{S}$ denotes the operation of selecting an element b uniformly at random from $\mathbf{S}$. If x and y are two strings, $|x|$ denotes the length of x, and $x \parallel y$ denotes the concatenation of x and y. Let $\mathbb{N}$ be the set of natural numbers. 1^λ ($\lambda \in \mathbb{N}$) denotes the string of λ ones. Denote λ as a security parameter. $\mathrm{negl}(\lambda)$ denotes a negligible function (in λ), i.e., $\forall n > 0$, there exists $\lambda_0 \in \mathbb{N}$, s.t., $\mathrm{negl}(\lambda) < 1/\lambda^n$, $\forall \lambda > \lambda_0$.

A. Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, and g is a generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be an efficiently-computable map satisfying that: 1) Non-degeneracy: $e(g, g) \neq 1$; 2) Bilinearity: $\forall u, v \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$. We say $(\mathbb{G}, \mathbb{G}_T)$ is a bilinear group pair and e is a bilinear map from $\mathbb{G}$ into $\mathbb{G}_T$.

B. Definitions for ABE and AB-KEM

There are two kinds of ABE schemes: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE).

In KP-ABE, each ciphertext is associated with a set of attributes and each private key with an access policy. Alternatively, in CP-ABE, the roles are flipped: attributes sets are associated with private keys and access policies with ciphertexts. Next, we present a unified definition for both KP-ABE and CP-ABE schemes.

Definition 1 [23]: Let S represent an attribute set and $\mathbb{A}$ an access structure. We set I_{key} and I_{enc} as the inputs to key generation and encryption algorithms respectively. In a KP-ABE scheme $(I_{key}, I_{enc}) = (\mathbb{A}, S)$, while in a CP-ABE scheme $(I_{key}, I_{enc}) = (S, \mathbb{A})$. The function f is defined as follows:

$$f(I_{key}, I_{enc}) = \begin{cases} 1, & \text{if } I_{enc} \in I_{key} \text{ in KP-ABE setting} \\ 1, & \text{if } I_{key} \in I_{enc} \text{ in CP-ABE setting} \\ 0, & \text{otherwise.} \end{cases}$$

Definition 2 (ABE): An ABE scheme with an attribute universe U for an access structure space $\mathcal{P}$ is defined by the following polynomial-time algorithms:

Setup$(1^\lambda, U) \rightarrow (PK, MSK)$*: The setup algorithm takes as input a security parameter λ and an attribute universe U, then outputs a public key PK and a master secret key MSK.*

KeyGen$(PK, MSK, I_{key}) \rightarrow SK$*: The key generation algorithm takes as input a public key PK, the master secret key MSK and an access structure $I_{key} \in \mathcal{P}$ for KP-ABE ($I_{key} \subseteq U$ for CP-ABE), then outputs a private key SK.*

Encrypt$(PK, M, I_{enc}) \rightarrow CT$*: The encryption algorithm takes as input a public key PK, a message M, and an attribute set $I_{enc} \subseteq U$ for KP-ABE ($I_{enc} \in \mathcal{P}$ for CP-ABE), then outputs a ciphertext CT.*

Decrypt$(SK, CT) \rightarrow M$*: The decryption algorithm takes a private key SK and a ciphertext CT as input, then outputs a message M if $f(I_{key}, I_{enc}) = 1$ and $\perp$ otherwise.*

Correctness. $\forall$ $(PK, MSK) \leftarrow$**Setup**$(1^\lambda, U)$, $SK \leftarrow$ **KeyGen**(PK, MSK, I_{key}), $\forall$ M in the message space, if $f(I_{key}, I_{enc}) = 1$, $M =$**Decrypt**$(SK,$ **Encrypt**$(PK, M, I_{enc}))$.

Similarly, a unified definition for AB-KEM in both KP and CP settings is presented below. Here we omit descriptions of inputs for simplicity.

Definition 3 (AB-KEM): An AB-KEM with an attribute universe U for an access structure space $\mathcal{P}$ is a tuple of the following polynomial-time algorithms:

Setup$(1^\lambda, U) \rightarrow (PK, MSK)$*: The setup algorithm returns a public key and a master secret key (PK, MSK).*

KeyGen$(PK, MSK, I_{key}) \rightarrow SK$*: The key generation algorithm returns a private key SK.*

Encrypt$(PK, I_{enc}) \rightarrow (\mathsf{DK}, CT)$*: The encryption algorithm returns a session key DK and a ciphertext CT.*

Decrypt$(SK, CT) \rightarrow \mathsf{DK}$*: The decryption algorithm returns the session key DK if $f(I_{key}, I_{enc}) = 1$, and $\perp$ otherwise.*

Correctness. $\forall$ $(PK, MSK) \leftarrow$ **Setup**$(1^\lambda, U)$, $SK \leftarrow$ **KeyGen**(PK, MSK, I_{key}), and $(\mathsf{DK}, CT) \leftarrow$ **Encrypt**(PK, I_{enc}), if $f(I_{key}, I_{enc}) = 1$, **Decrypt**(SK, CT) outputs DK.

Subsequently, we describe security models for an ABE scheme Π_{ABE} and an AB-KEM Π_{KEM}, then provide the formal definitions. Before that, we need to introduce the key generation oracle defined as follows.

Key Generation Oracle. Given access to a public key PK and a master secret key MSK, the key generation oracle $\mathcal{O}^{\text{KeyGen}}(\cdot)$ takes as input I_{key} and returns a private key SK generated from: **KeyGen**$(PK, MSK, I_{key}) \rightarrow SK$.

$Exp^{\text{IND}}_{\mathcal{A}, \Pi_{\text{ABE}}}(1^\lambda, U)$:

- **Setup.** The challenger runs the setup algorithm to obtain (PK, MSK), and returns PK to the adversary $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ is given access to the oracle $\mathcal{O}^{\text{KeyGen}}(\cdot)$.
- **Challenge.** $\mathcal{A}$ submits two (equal length) messages M_0, M_1 and I^*_{enc} to the challenger, where $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} issued to the oracle $\mathcal{O}^{\text{KeyGen}}(\cdot)$ in Phase 1. The challenger picks $b \xleftarrow{R} \{0, 1\}$ and sends to $\mathcal{A}$ the challenge ciphertext CT^* generated from: **Encrypt**$(PK, M_b, I^*_{enc}) \rightarrow CT^*$.
- **Phase 2.** $\mathcal{A}$ is given oracle access to $\mathcal{O}^{\text{KeyGen}}(\cdot)$ with the restriction that $f(I_{key}, I^*_{enc}) \neq 1$ for all issued I_{key}.
- **Guess.** $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$. The experiment returns 1 if and only if $b' = b$.

Definition 4 (Data Privacy of ABE): An ABE scheme is indistinguishable under chosen plaintext attack (IND-CPA secure) if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage in this security game:

$$\left| \Pr\left[Exp^{\text{IND}}_{\mathcal{A}, \Pi_{\text{ABE}}}(1^\lambda, U) = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

Selective Security*: An ABE scheme is selectively secure if we add an* **Init** *stage before* **Setup** *where the adversary commits to the challenge I^*_{enc}.*

$Exp^{\text{IND}}_{\mathcal{A}, \Pi_{\text{KEM}}}(1^\lambda, U)$:

- **Setup.** The challenger runs the setup algorithm to obtain (PK, MSK) and sends PK to the adversary $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ is given oracle access to $\mathcal{O}^{\text{KeyGen}}(\cdot)$.
- **Challenge.** $\mathcal{A}$ submits I^*_{enc} to the challenger satisfying that $f(I_{key}, I^*_{enc}) \neq 1$ for all I_{key} issued to the oracle in Phase 1. The challenger runs **Encrypt**$(PK, I^*_{enc}) \rightarrow (\mathsf{DK}, C^*)$. It selects $b \xleftarrow{R} \{0, 1\}$. If $b = 0$, it returns (DK, C^*) to $\mathcal{A}$. If $b = 1$, it chooses a random session key RK from the session key space and returns (RK, C^*).
- **Phase 2.** $\mathcal{A}$ is given access to $\mathcal{O}^{\text{KeyGen}}(\cdot)$ with the restriction that $f(I_{key}, I^*_{enc}) \neq 1$ for all issued I_{key}.
- **Guess.** $\mathcal{A}$ outputs a guess b' of b. The experiment returns 1 if and only if $b' = b$.

Definition 5 (Data Privacy of AB-KEM): An AB-KEM is IND-CPA secure if for any PPT adversary $\mathcal{A}$, the advantage in this security game:

$$\left| \Pr\left[Exp^{\text{IND}}_{\mathcal{A}, \Pi_{\text{KEM}}}(1^\lambda, U) = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

Selective Security*: An AB-KEM is selectively secure if we add an* **Init** *stage before* **Setup** *where the adversary outputs the challenge I^*_{enc}.*

C. Symmetric-Key Encryption

Definition 6 [24]: A symmetric-key encryption scheme is a tuple of probabilistic polynomial-time (PPT) algorithms:
Gen$(1^\lambda) \rightarrow \mathsf{NK}$: The key generation algorithm takes as input a security parameter λ and outputs a key NK.

Enc$(\mathsf{NK}, M) \rightarrow C$: The encryption algorithm takes as input a key NK and a plaintext M, then outputs a ciphertext C.
Dec$(\mathsf{NK}, C) \rightarrow M$: The decryption algorithm takes as input a key NK and a ciphertext C, then outputs a plaintext M.
Correctness. $\forall\ \mathsf{NK} \leftarrow \mathbf{Gen}(1^\lambda)$, and $\forall\ M$ in the message space, $M = \mathbf{Dec}(\mathsf{NK}, \mathbf{Enc}(\mathsf{NK}, M))$.

Definition 7: A symmetric-key encryption scheme is semantically secure if for any PPT adversary $\mathcal{A}$ the following advantage is negligible:

$$\left|\Pr\left[\begin{array}{l}(M_0, M_1) \leftarrow \mathcal{A}(1^\lambda);\\ \mathsf{NK} \leftarrow \mathbf{Gen}(1^\lambda);\\ b \overset{R}{\leftarrow} \{0,1\};\\ C^* \leftarrow \mathbf{Enc}(\mathsf{NK}, M_b)\end{array} : \mathcal{A}(1^\lambda, M_0, M_1, C^*) = b\right] - \frac{1}{2}\right|.$$

D. Commitment Scheme

A commitment scheme is defined by a pair of probabilistic polynomial-time (PPT) algorithms $(\mathsf{Commit}, \mathsf{Decommit})$ between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$. During the commitment phase, $\mathcal{S}$ commits to a message M with a random coin r and sends $\widehat{C}$ to $\mathcal{R}$, where $\widehat{C} = \mathsf{Commit}_r(M)$. During the decommitment phase, $\mathcal{S}$ sends (M, r) to $\mathcal{R}$ and $\mathcal{R}$ verifies $\mathsf{Decommit}_r(\widehat{C}) \overset{?}{=} M$. $\mathcal{S}$ and $\mathcal{R}$ are given common parameters which are omitted here for simplicity. The security requirements of a commitment scheme are defined as follows.
Completeness: For any message M,

$$\Pr[\mathsf{Decommit}_r(\mathsf{Commit}_r(M))] = M.$$

(Computational) Hiding: For all (PPT) receiver $\mathcal{R}^*$, it holds that the following is negligible ($\forall M \neq M'$):

$$\left|\Pr\left[\mathcal{R}^*(\mathsf{Commit}_r(M)) = 1\right] - \Pr[\mathcal{R}^*(\mathsf{Commit}_{r'}(M')) = 1]\right|$$

(Computational) Binding: For all (PPT) sender $\mathcal{S}^*$, it holds that the following is negligible ($\forall M \neq M'$):

$$\Pr\left[\begin{pmatrix}M, r,\\ M', r'\end{pmatrix} \leftarrow \mathcal{S}^*(\widehat{C}) : \mathcal{R}(\widehat{C}, M, r) = \mathcal{R}(\widehat{C}, M', r') = 1\right]$$

Throughout this paper, we only require the commitment scheme to be computationally hiding and binding.

E. Key Derivation Function (KDF)

A key derivation function (KDF) [25] takes an initial keying material, which contains some good amount of randomness but is not distributed uniformly or has leaked some information to an adversary, and derives pseudorandom secret keys. The pseudorandom secret keys are indistinguishable from a random uniform distribution of the same length by polynomial-time computation. Especially, part of the bits or keys output by a KDF should not leak information on the other generated bits. The definitions of a KDF and its security are provided as follows.

Definition 8 (KDF): A key derivation function (KDF) takes as input a secret key DK and a length ℓ, and outputs a string of ℓ bits.

Definition 9 (Security of KDF): A key derivation function KDF is secure if for any PPT adversary $\mathcal{A}$, the following is negligible:

$$\left|\Pr\left[\mathcal{A}(\mathsf{KDF}(\mathsf{DK}, \ell)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{R}) = 1\right]\right|.$$

Here DK denotes an initial secret key and ℓ is an output length, and R is chosen uniformly at random from $\{0,1\}^\ell$.

F. Endomorphism Transform

Motivated by all-or-nothing transform (AONT) [10], [11], we define a new transform, called endomorphism transform, which may be viewed as a subclass of AONT with the property of endomorphism. In terms of functionality, an endomorphism transform changes an input into two parts: one part is the secret output that maintains completely confidential from any adversary as opposed to AONT on the sufficient bits secrecy; the other is the public output that can be obtained by any adversary similar to AONT, but with endomorphism requirement on the input domain. In terms of security, an endomorphism transform has the same security requirements as AONT, i.e., every PPT adversary should have no information about the input of the transform assuming it obtains the public output.

Specifically, the formal definition of an endomorphism transform on a multiplicative group is provided below.

Definition 10 (Endomorphism Transform): Let $\mathbb{X}$ be a multiplicative group and $\mathbb{Z}$ denotes the set of integers. A randomized polynomial-time computable function $T : \mathbb{X} \rightarrow \mathbb{Z} \times \mathbb{X}$ is an endomorphism transform if it satisfies that:
1) Invertibility : *there is a polynomial-time machine T' such that for any $x \in \mathbb{X}$ and any $(T_1, T_2) \in T(x)$, we have $T'(T_1, T_2) = x$. Here T_1 denotes the secret output and T_2 denotes the public output.*
2) Indistinguishability : *the public outputs $\{T_2(x) : x \in \mathbb{X}\}$ are all indistinguishable from each other, i.e., $T_2(x_0) \approx T_2(x_1), \forall x_0,\ x_1 \in \mathbb{X}$. Here $\approx$ denotes computational indistinguishability.*
3) Endomorphism : *the public output $T_2 : \mathbb{X} \rightarrow \mathbb{X}$ is an endomorphism on the multiplicative group $\mathbb{X}$, i.e.,*

$$T_2(u \cdot v) = T_2(u) \cdot T_2(v), \qquad \forall u,\ v \in \mathbb{X}.$$

Consider an endomorphism transform T defined on a multiplicative cyclic group $\mathbb{X}$ of prime order p. Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map. Up to isomorphism, there is just one cyclic group of prime order p. Thus the public output $T_2 : \mathbb{X} \rightarrow \mathbb{X}$ meets the same multiplicative homomorphism either on $\mathbb{G}$ or $\mathbb{G}_T$. We note that the construction where $e(g, T_2(h)) = e(T_2(g), h) = T_2(e(g, h)), \forall\, g, h \in \mathbb{G}$, holds for both prime order and composite order bilinear groups [26].

In fact, for any generator $c \in \mathbb{G}$, suppose that $T_2(c) = c^K$, $k \in \mathbb{Z}_p$. Since T_2 is a multiplicative homomorphism on $\mathbb{G}$, it holds that $T_2(g) = g^k$ for any $g \in \mathbb{G}$. The non-degeneracy of the bilinear map e implies that $e(c, c)$ is a generator of $\mathbb{G}_T$. With the same multiplicative homomorphism T_2 on $\mathbb{G}_T$, we have $T_2(e(c, c)) = e(c, c)^k$. Then $T_2(e(g, h)) = e(g, h)^k$, $\forall\, g, h \in \mathbb{G}$.

Let us review the key blinding technique proposed in [2]: Choose a blinding factor exponent $z \in \mathbb{Z}_p^*$, where $p \in \Theta(2^\lambda)$, and produce the transform key $TK = SK^z$ for the private key SK, while the blinding factor z is kept as the secret retrieving key. Actually, this implicitly defines an endomorphism transform T on a cyclic group $\mathbb{X}$ of prime order p as $T(d) = (T_1, T_2) = (z, d^z)$, $\forall z \in \mathbb{Z}_p^*, \forall d \in \mathbb{X}$. Here $T_2(d) = d^z$, $\forall d \in \mathbb{X}$ is a multiplicative homomorphism on $\mathbb{X}$ and $\mathbb{X}$ can be $\mathbb{G}$ or $\mathbb{G}_T$. When the secret output $T_1 = z$ is kept hidden, all the public outputs $\{d^z : d \in \mathbb{X}\}$ are computationally indistinguishable from each other supposing that the Discrete-Logarithm assumption holds. The inverse transform of T is defined as $T'(T_1, T_2) = T_2^{1/T_1}$ for any $T(d)$ to recover the input d.

Generally, an endomorphism transform defined on a multiplicative group can be used to outsource decryption of AB-KEM ciphertexts for most pairing-based AB-KEMs of which the decryption algorithm merely involves multiplication and pairing evaluation. Taking a private key SK of an AB-KEM as input of the endomorphism transform, a user obtains two outputs $(T_1, T_2(SK))$, where $T_2(SK)$ denotes the result of operating T_2 on each key component of SK, and sends the public output $T_2(SK)$ to a proxy for outsourcing the decryption while keeps the output T_1 secret. Since the proxy does not get any knowledge about the secret output, it can not learn any information about the original private key even if the public output is provided. The proxy runs the decryption algorithm $\mathbf{Decrypt}(T_2(SK), CT)$ of the AB-KEM by using the transform key $T_2(SK)$ and sends the consequence to the user. For any AB-KEM of which the decryption algorithm just contains multiplication and pairing evaluation, since the public output is an endomorphism on the input group, it is not difficult to infer that the result from the proxy is the same as applying the endomorphism transform to the decrypted result by using the original private key, i.e., $\mathbf{Decrypt}(T_2(SK), CT) = T_2(\mathbf{Decrypt}(SK, CT))$. Because the endomorphism transform is efficiently computable and invertible, the user can recover the message successfully by applying the inverse transform to the result received from the proxy.

III. Model of VO-ABE

In [3], the model of CP-ABE with outsourced decryption consists of seven algorithms ($\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Encrypt}, \mathbf{Decrypt}, \mathbf{GenTK}_{out}$, $\mathbf{Transform}_{out}$, $\mathbf{Decrypt}_{out}$). A trusted party generates the public parameters and a master secret key by running the algorithm **Setup**, and a user obtains a private key generated by the trusted party running **KeyGen**. After taking the ciphertext, the user decides whether to outsource decryption of the ciphertext. If the user wants to outsource decryption, he can execute the algorithm $\mathbf{GenTK}_{out}$ and use his private key to generate the transform key by himself. Taking as input the transform key and a ciphertext, the proxy is able to change the ciphertext into a constant-size ciphertext by the algorithm $\mathbf{Transform}_{out}$ if the set of attributes associated with the private key satisfies the access structure associated with the ciphertext. Then the plaintext can be recovered from the transformed ciphertext by the algorithm $\mathbf{Decrypt}_{out}$. Based on the model of CP-ABE with outsourced decryption proposed in [3], we formally define a verification algorithm and provide the model of VO-ABE as follows.

Definition 11 (VO-ABE): A VO-ABE with an attribute universe U for an access structure space $\mathcal{P}$ is defined by the following polynomial-time algorithms:

$\mathbf{Setup}(1^\lambda, U) \rightarrow (PP, MSK)$*: The setup algorithm takes as input a security parameter λ and an attribute universe description U, then outputs the public parameters PP and a master secret key MSK.*

$\mathbf{KeyGen}(PP, MSK, I_{key}) \rightarrow SK_I$*: The key generation algorithm takes as input the public parameters PP, the master secret key MSK, and an access structure $I_{key} \in \mathcal{P}$ for KP-ABE (an attribute set $I_{key} \subseteq U$ for CP-ABE), then outputs a private key SK_I.*

$\mathbf{Encrypt}(PP, M, I_{enc}) \rightarrow CT$*: The encryption algorithm takes as input the public parameters PP, a message M and an attribute set $I_{enc} \subseteq U$ for KP-ABE (an access structure $I_{enc} \in \mathcal{P}$ for CP-ABE), then outputs a ciphertext CT.*

$\mathbf{Decrypt}(PP, SK_I, CT) \rightarrow M$*: The decryption algorithm takes as input the public parameters PP, a private key SK_I and a ciphertext CT, then returns a message M.*

$\mathbf{GenTK}(PP, SK_I) \rightarrow (TSK, TK_I)$*: The transform key generation algorithm takes as input the public parameters PP and a private key SK_I, then outputs a transform key TK_I and a secret value TSK used for retrieving the encrypted message.*

$\mathbf{Transform}_{out}(PP, CT, TK_I) \rightarrow CT'$*: The transform algorithm takes as input the public parameters PP, a ciphertext CT and a transform key TK_I, then outputs a transformed ciphertext CT'.*

$\mathbf{Verify}(PP, CT', TSK) \rightarrow (\mathsf{b}, st)$*: The verification algorithm takes as input the public parameters PP, a transformed ciphertext CT' and a secret value TSK, then outputs a bit b and a status value st.*

$\mathbf{Decrypt}'(PP, CT, CT', TSK) \rightarrow M$*: The decryption algorithm takes as input the public parameters PP, a ciphertext CT, a transformed ciphertext CT' and a secret value TSK, then runs* $\mathbf{Verify}(PP, CT', TSK)$ *to obtain b and st. If $\mathsf{b} = 1$, it outputs M. If $\mathsf{b} = 0$, it returns an error symbol $\perp$.*

Correctness. For all $(PP, MSK) \leftarrow \mathbf{Setup}(1^\lambda, U)$, $SK_I \leftarrow \mathbf{KeyGen}(PP, MSK, I_{key})$, $CT \leftarrow \mathbf{Encrypt}(PP, M, I_{enc})$, $(TSK, TK_I) \leftarrow \mathbf{GenTK}(PP, SK_I)$, $CT' \leftarrow \mathbf{Transform}_{out}(PP, CT, TK_I)$. Consider two cases:

Case 1: $f(I_{key}, I_{enc}) = 1$. $\mathbf{Decrypt}(PP, SK_I, CT) \rightarrow M$, and $\mathbf{Decrypt}'(PP, CT, CT', TSK) \rightarrow M$ if $\mathbf{Verify}(PP, CT', TSK)$ returns 1.

Case 2: $f(I_{key}, I_{enc}) \neq 1$. $\mathbf{Decrypt}(PP, SK_I, CT)$ and $\mathbf{Decrypt}'(PP, CT, CT', TSK)$ output $\perp$.

Our model of VO-ABE can be considered in both KP and CP settings. A user is able to generate the transform key by himself if it is necessary to outsource decryption. The verification algorithm is defined for the final decryption executed by the user. After receiving the transformed ciphertext, the user verifies the correctness of the transform done by the proxy and obtains an error symbol if the verification fails. If the transformed ciphertext passes the verification, the user takes

the status from the verification algorithm and continues to run the decryption algorithm, then can successfully recover the plaintext.

In the following, we describe security games for VO-ABE and present the formal definitions. Apart from the key generation oracle $\mathcal{O}^{\text{KeyGen}}(\cdot)$ defined before, we need to provide the definitions of a transform key generation and decryption oracles for the security model of VO-ABE.

Oracles Descriptions. All the oracles are given access to the public parameters and a master secret key (PP, MSK), however, we omit them in descriptions for simplicity.

1) The decryption oracle $\mathcal{O}^{\text{Decrypt}}(\cdot,\cdot)$ takes as input I_{key}, CT and returns the output generated from: $\textbf{KeyGen}(PP, MSK, I_{key}) \rightarrow SK_I$; $\textbf{Decrypt}(PP, SK_I, CT) \rightarrow M/\perp$.

2) The transform key generation oracle $\mathcal{O}^{\text{GenTK}}(\cdot)$ takes as input I_{key} and returns TK_I generated from: $\textbf{KeyGen}(PP, MSK, I_{key}) \rightarrow SK_I$; $\textbf{GenTK}(PP, SK_I) \rightarrow (TSK, TK_I)$.

3) The decryption oracle $\mathcal{O}^{\text{Decrypt}'}(\cdot,\cdot,\cdot)$ takes as input I_{key}, CT, CT' and returns the output generated from: $\textbf{KeyGen}(PP, MSK, I_{key}) \rightarrow SK_I$, $\textbf{GenTK}(PP, SK_I) \rightarrow (TSK, TK_I)$, $\textbf{Decrypt}'(PP, CT, CT', TSK) \rightarrow M/\perp$.

$Exp^{\text{IND}}_{\mathcal{A},\Pi_{\text{VOABE}}}(1^\lambda, U)$:

- **Setup.** The challenger runs the setup algorithm to obtain (PP, MSK) and returns PP to the adversary $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ is given access to $\mathcal{O}^{\text{KeyGen}}(\cdot), \mathcal{O}^{\text{GenTK}}(\cdot)$.
- **Challenge.** $\mathcal{A}$ submits two (equal length) messages M_0, M_1 and I^*_{enc} to the challenger, where $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} issued to $\mathcal{O}^{\text{KeyGen}}(\cdot)$ in Phase 1. The challenger picks $b \xleftarrow{R} \{0, 1\}$ and sends to $\mathcal{A}$ the challenge ciphertext CT^* generated from: $\textbf{Encrypt}(PP, M_b, I^*_{enc}) \rightarrow CT^*$.
- **Phase 2.** $\mathcal{A}$ is given access to the oracles as in Phase 1 with the restriction that $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} issued to $\mathcal{O}^{\text{KeyGen}}(\cdot)$.
- **Guess.** $\mathcal{A}$ outputs a guess b' of b. The experiment returns 1 if and only if $b' = b$.

Definition 12 (Data Privacy): A VO-ABE scheme is IND-CPA secure if for any PPT adversary $\mathcal{A}$, the advantage in this security game:

$$\left|\Pr\left[Exp^{\text{IND}}_{\mathcal{A},\Pi_{\text{VOABE}}}(1^\lambda, U) = 1\right] - \frac{1}{2}\right| \leq \text{negl}(\lambda)$$

Selective Security*: A VO-ABE scheme is selectively secure if we add an* **Init** *stage before* **Setup** *where the adversary commits to the challenge I^*_{enc}.*

$Exp^{\text{SOUND}}_{\mathcal{A},\Pi_{\text{VOABE}}}(1^\lambda, U)$:

- **Setup.** The challenger runs the setup algorithm to obtain (PP, MSK) and sends PP to the adversary $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ is given access to the oracles $\mathcal{O}^{\text{KeyGen}}(\cdot)$, $\mathcal{O}^{\text{GenTK}}(\cdot)$, $\mathcal{O}^{\text{Decrypt}}(\cdot,\cdot)$, $\mathcal{O}^{\text{Decrypt}'}(\cdot,\cdot,\cdot)$.
- **Challenge.** $\mathcal{A}$ submits a message M^* and I^*_{enc} to the challenger. The challenger computes **Encrypt** $(PP, M^*, I^*_{enc}) \rightarrow CT^*$ and returns to $\mathcal{A}$ the challenge ciphertext CT^*.
- **Phase 2.** $\mathcal{A}$ is given access to the oracles as in Phase 1.
- **Forge.** $\mathcal{A}$ outputs I^*_{key} and a transformed ciphertext $CT^{*\prime}$. The challenger obtains TSK for I^*_{key} by running $\textbf{KeyGen}(PP, MSK, I^*_{key}) \rightarrow SK_I$ and $\textbf{GenTK}(PP, SK_I) \rightarrow (TSK, TK_I)$. $\mathcal{A}$ wins the game if **Verify** $(PP, CT^{*\prime}, TSK)$ returns 1 and $\textbf{Decrypt}'(PP, CT^*, CT^{*\prime}, TSK) \notin \{M^*, \perp\}$.

Remark: Without loss of generality, we consider the above game under the following assumptions:

1) The adversary $\mathcal{A}$ does not submit I_{key} to $\mathcal{O}^{\text{GenTK}}(\cdot)$ if it has issued the same I_{key} to $\mathcal{O}^{\text{KeyGen}}(\cdot)$, since $\mathcal{A}$ can generate the transform key by itself.

2) $f(I^*_{key}, I^*_{enc}) = 1$. For the case of $f(I^*_{key}, I^*_{enc}) \neq 1$, $\textbf{Decrypt}'(PP, CT^*, CT^{*\prime}, TSK)$ outputs $\perp$ and $\mathcal{A}$ fails the game.

3) $\mathcal{A}$ has submitted I^*_{key} to the oracle $\mathcal{O}^{\text{GenTK}}(\cdot)$ before it outputs I^*_{key} and $CT^{*\prime}$, i.e., $\mathcal{A}$ obtains the transform key TK_I for I^*_{key} and the challenger owns the corresponding TSK. This assumption is reasonable since the challenger can obtain TSK as in the response of the transform key generation oracle if $\mathcal{A}$ has not issued the transform key query for I^*_{key}.

Definition 13 (Verification Soundness): A VO-ABE scheme meets the verification soundness if the probability for any PPT adversary to win the above game is negligible.

IV. ABE Scheme With Verifiable Decryption

Our ABE scheme with verifiable decryption has three building blocks: an AB-KEM, a symmetric-key encryption scheme and a commitment scheme. Our ciphertext consists of the ciphertext part of the AB-KEM and the ciphertext generated from combining a hybrid encryption and a commitment of the plaintext by bundling the same randomness, which is used for verification.

Let $\Pi_{\text{KEM}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be an AB-KEM and let $\Pi_{\text{SKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric-key encryption scheme. Let KDF denote a key derivation function with the output length ℓ, where ℓ is selected according to the length of the private key in the symmetric-key encryption scheme. Given a commitment scheme $(\mathsf{Commit}, \mathsf{Decommit})$, we construct an ABE scheme with verifiable decryption as follows.

- **Setup**$(1^\lambda, U)$: The algorithm runs $\mathsf{Setup}(1^\lambda, U) \rightarrow (PK, MSK)$, and chooses the key derivation function KDF with the output length ℓ, then publishes the public parameters $PP = (PK, \mathsf{KDF}, \ell)$ and MSK.
- **KeyGen**(PP, MSK, I_{key}): The key generation algorithm runs $\mathsf{KeyGen}(PK, MSK, I_{key}) \rightarrow SK$, and outputs $SK_I = (I_{key}, SK)$.
- **Encrypt**(PP, M, I_{enc}): Parse $PP = (PK, \mathsf{KDF}, \ell)$. The encryption algorithm runs $\mathsf{Encrypt}(PK, I_{enc}) \rightarrow (\mathsf{DK}, C)$. It chooses a random coin r, sets $\widehat{C} = \mathsf{Commit}_r(M)$, and runs $\mathsf{Enc}(\mathsf{KDF}(\mathsf{DK}, \ell), M \parallel r) \rightarrow \overline{C}$, then outputs the ciphertext $CT = (I_{enc}, C, \overline{C}, \widehat{C})$.
- **Decrypt**(PP, SK_I, CT): Parse $PP = (PK, \mathsf{KDF}, \ell)$, $CT = (I_{enc}, C, \overline{C}, \widehat{C})$ and $SK_I = (I_{key}, SK)$. The decryption algorithm runs $\mathsf{Decrypt}(SK, C) \rightarrow \mathsf{DK}$ and $\mathsf{Dec}(\mathsf{KDF}(\mathsf{DK}, \ell), \overline{C}) \rightarrow M \parallel r$, then outputs M if $\mathsf{Decommit}_r(\widehat{C}) = M$, and an error symbol $\perp$ otherwise.

Correctness. For all $(PP, MSK) \leftarrow \textbf{Setup}(1^\lambda, U)$, $SK_I \leftarrow \textbf{KeyGen}(PP, MSK, I_{key})$, $CT \leftarrow \textbf{Encrypt}(PP, M, I_{enc})$, where $SK_I = (I_{key}, SK)$, $CT = (I_{enc}, C, \overline{C}, \widehat{C})$ and $(\mathsf{DK}, C) \leftarrow \mathsf{Encrypt}(PK, I_{enc})$, $\widehat{C} = \mathsf{Commit}_r(M)$, $\overline{C} \leftarrow \mathsf{Enc}(\mathsf{KDF}(\mathsf{DK}, \ell), M \parallel r)$. Consider two cases:

Case 1: $f(I_{key}, I_{enc}) = 1$. $\mathsf{Decrypt}(SK, C) \rightarrow \mathsf{DK}$, $\mathsf{Dec}(\mathsf{KDF}(\mathsf{DK}, \ell), \overline{C}) \rightarrow M \parallel r$. If $\mathsf{Decommit}_r(\widehat{C}) = M$, $\textbf{Decrypt}(PP, SK_I, CT) \rightarrow M$; otherwise, $\textbf{Decrypt}(PP, SK_I, CT)$ outputs the error symbol $\perp$.

Case 2: $f(I_{key}, I_{enc}) \neq 1$. $\textbf{Decrypt}(PP, SK_I, CT)$ outputs the error symbol $\perp$.

For convenience, we denote the above ABE scheme as BasicABE. We will construct our VO-ABE scheme based on BasicABE. Now we describe how security of the AB-KEM, the symmetric-key encryption scheme, the key derivation function and the commitment scheme implies the security of BasicABE. One of our main results is provided as follows.

Theorem 1: Suppose that the AB-KEM Π_{KEM} *is IND-CPA secure, the key derivation function* KDF *is secure, the symmetric-key encryption scheme* Π_{SKE} *is semantically secure and the commitment scheme* (Commit, Decommit) *is computationally hiding. Then the constructed scheme* BasicABE *is IND-CPA secure.*

Proof: To prove the theorem, we consider four games:

- Game_0: The real IND-CPA security game of ABE, where the challenge ciphertext $CT^* = (I^*_{enc}, C^*, \overline{C}^*, \widehat{C}^*)$ is generated from the procedure that the challenger executes $\mathsf{Encrypt}(PK, I^*_{enc}) \rightarrow (\mathsf{DK}, C^*)$, then picks $b \xleftarrow{R} \{0, 1\}$ and a random r, finally computes $\widehat{C}^* = \mathsf{Commit}_r(M_b)$ and runs $\mathsf{Enc}(\mathsf{KDF}(\mathsf{DK}, \ell), M_b \parallel r) \rightarrow \overline{C}^*$.
- Game_1: Same as Game_0 except that the challenger runs $\mathsf{Enc}(\mathsf{NK}, M_b \parallel r) \rightarrow \overline{C}^*$ where NK is generated randomly from the key generation algorithm Gen.
- Game_2: Same as Game_1 except that the challenger runs $\mathsf{Enc}(\mathsf{NK}, 0^{|M_b \| r|}) \rightarrow \overline{C}^*$ for the challenge ciphertext.
- Game_3: Same as Game_2 except that the challenger computes $\widehat{C}^* = \mathsf{Commit}_r(0^{|M_b|})$ for the challenge ciphertext.

We can show the computational indistinguishability between the pairs Game_0 and Game_1, Game_1 and Game_2, Game_2 and Game_3, respectively, which implies that Game_3 is computationally indistinguishable from Game_0. Since the challenge ciphertext in Game_3 contains no information about the messages submitted by the adversary, we conclude that the advantage of the adversary in Game_3 is negligible. Thus the advantage of the adversary in the real game is negligible. We prove this theorem by the following lemmas.

Lemma 1: Suppose that the AB-KEM $\Pi_{\mathrm{KEM}} =$ (Setup, KeyGen, Encrypt, Decrypt) *is IND-CPA secure, and the key derivation function* KDF *is secure. Then* Game_0 *and* Game_1 *are computationally indistinguishable.*

Proof: Consider the following game:

Game': Same as Game_0 except that the challenger runs $\mathsf{Enc}(\mathsf{KDF}(\mathsf{RK}, \ell), M_b \parallel r) \rightarrow \overline{C}^*$ where RK represents a random session key of the AB-KEM.

We first show that IND-CPA security of the AB-KEM implies that Game' is computationally indistinguishable from Game_0.

Suppose there exists a PPT adversary $\mathcal{A}$ that can distinguish Game_0 and Game' with non-negligible probability. We build an algorithm $\mathcal{B}$ to break IND-CPA security of the AB-KEM. $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

- **Setup.** The simulator $\mathcal{B}$ receives the public key PK from the challenger, then selects the key derivation function KDF with the output length ℓ, and sends the public parameters $PP = (PK, \mathsf{KDF}, \ell)$ to $\mathcal{A}$.
- **Phase 1.** $\mathcal{B}$ forwards any private key query from $\mathcal{A}$ to its own key generation oracle and returns to $\mathcal{A}$ the answer.
- **Challenge.** $\mathcal{A}$ submits (equal length) M_0, M_1 and I^*_{enc} where $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} queried in Phase 1. $\mathcal{B}$ sends I^*_{enc} to the challenger. The challenger runs $\mathsf{Encrypt}(PK, I^*_{enc}) \rightarrow (\mathsf{DK}, C^*)$, then picks $\beta \xleftarrow{R} \{0, 1\}$, and returns (KK_β, C^*) to $\mathcal{B}$. If $\beta = 0$, $\mathsf{KK}_0 = \mathsf{DK}$; otherwise, $\mathsf{KK}_1 = \mathsf{RK}$ where RK is a random session key. $\mathcal{B}$ selects $b \xleftarrow{R} \{0, 1\}$ and a random coin r, then computes $\widehat{C}^* = \mathsf{Commit}_r(M_b)$, and runs $\mathsf{Enc}(\mathsf{KDF}(\mathsf{KK}_\beta, \ell), M_b \parallel r) \rightarrow \overline{C}^*$. Finally $\mathcal{B}$ sends $CT^* = (I^*_{enc}, C^*, \overline{C}^*, \widehat{C}^*)$ to $\mathcal{A}$ as the challenge ciphertext.
- **Phase 2.** $\mathcal{A}$ adaptively issues private key queries with the restriction that $f(I_{key}, I^*_{enc}) \neq 1$ for any queried I_{key}, and $\mathcal{B}$ proceeds as in Phase 1.
- **Guess.** $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$. If $b' = b$, $\mathcal{B}$ outputs 0; otherwise, $\mathcal{B}$ outputs 1.

We can see that if $\beta = 0$, Game_0 has been properly simulated; otherwise, Game' has been properly simulated. Hence, if the adversary $\mathcal{A}$ can distinguish Game_0 and Game' with non-negligible probability, we can construct an algorithm $\mathcal{B}$ to attack the IND-CPA secure AB-KEM with non-negligible advantage. Then the proof of the computational indistinguishability between Game_0 and Game' is completed.

Since the security of the KDF implies that $\mathsf{KDF}(\mathsf{RK}, \ell)$ is computationally indistinguishable from a randomly generated key of the symmetric-key encryption scheme, we can show that Game' and Game_1 are computationally indistinguishable by similar analysis as above. Therefore, we conclude that Game_0 and Game_1 are computationally indistinguishable. □

Lemma 2: Suppose the symmetric-key encryption scheme $\Pi_{\mathrm{SKE}} =$ (Gen, Enc, Dec) *is semantically secure. Then* Game_1 *and* Game_2 *are computationally indistinguishable.*

Proof: Suppose there exists a PPT adversary $\mathcal{A}$ that can distinguish Game_1 and Game_2 with non-negligible probability. We build an algorithm $\mathcal{B}$ to break the semantical security of the symmetric-key encryption scheme. $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

- **Setup.** After receiving the security parameter from the challenger, the simulator $\mathcal{B}$ calls $\mathsf{Setup}(1^\lambda, U)$ to obtain (PK, MSK) and sends PK to $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ issues private key queries. Since $\mathcal{B}$ knows the master secret key MSK, it can answer any queries.
- **Challenge.** $\mathcal{A}$ submits (equal length) M_0, M_1 and I^*_{enc} where $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} queried in Phase 1. $\mathcal{B}$ runs $\mathsf{Encrypt}(PK, I^*_{enc}) \rightarrow (\mathsf{DK}, C^*)$. After that, $\mathcal{B}$ selects $b \xleftarrow{R} \{0, 1\}$ and a random coin r, then computes $\widehat{C}^* = \mathsf{Commit}_r(M_b)$, and sends $\overline{M}_0 = M_b \parallel r$,

$\overline{M}_1 = 0^{|M_b \| r|}$ to the challenger. The challenger runs $\mathsf{Gen}(1^\lambda) \rightarrow \mathsf{NK}$, and picks $\beta \xleftarrow{R} \{0,1\}$, then encrypts the message $\overline{M}_\beta$ by running $\mathsf{Enc}(\mathsf{NK}, \overline{M}_\beta) \rightarrow \overline{C}^*$ and returns to $\mathcal{B}$ the ciphertext $\overline{C}^*$. $\mathcal{B}$ sends $CT^* = (I^*_{enc}, C^*, \overline{C}^*, \widehat{C}^*)$ to $\mathcal{A}$ as the challenge ciphertext.

- **Phase 2.** $\mathcal{A}$ adaptively issues private key queries with the restriction that $f(I_{key}, I^*_{enc}) \neq 1$ for any queried I_{key}, and $\mathcal{B}$ responds as in Phase 1.
- **Guess.** $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. If $b' = b$, $\mathcal{B}$ outputs 0; otherwise, $\mathcal{B}$ outputs 1.

We can see that if $\beta = 0$, Game_1 has been properly simulated; otherwise, Game_2 has been properly simulated. Hence, if the adversary $\mathcal{A}$ can distinguish Game_1 and Game_2 with non-negligible probability, we can construct an algorithm $\mathcal{B}$ to attack the semantically secure symmetric-key encryption scheme with non-negligible advantage. □

Lemma 3: Suppose that the commitment scheme $(\mathsf{Commit}, \mathsf{Decommit})$ *is computationally hiding. Then* Game_2 *and* Game_3 *are computationally indistinguishable.*

Proof: Suppose there is a PPT adversary $\mathcal{A}$ that can distinguish Game_2 and Game_3 with non-negligible probability. We construct an algorithm $\mathcal{B}$ to break the computational hiding of the commitment scheme. Let $\mathcal{S}$ be the sender corresponding to $\mathcal{B}$ as the receiver in the commitment scheme. $\mathcal{S}$ and $\mathcal{B}$ are given common parameters. The simulator $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

- **Setup.** $\mathcal{B}$ calls $\mathsf{Setup}(1^\lambda, U)$ to obtain (PK, MSK) and sends PK to $\mathcal{A}$.
- **Phase 1.** $\mathcal{B}$ answers private key queries issued by $\mathcal{A}$.
- **Challenge.** $\mathcal{A}$ submits (equal length) M_0, M_1 and I^*_{enc} where $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} issued in Phase 1. $\mathcal{B}$ picks $b \xleftarrow{R} \{0,1\}$, and sets $\overline{M}_0 = M_b$, $\overline{M}_1 = 0^{|M_b|}$, then sends $(\overline{M}_0, \overline{M}_1)$ to $\mathcal{S}$. After that, $\mathcal{S}$ picks $\beta \xleftarrow{R} \{0,1\}$ and selects a random r, then returns the commitment $\widehat{C}^* = \mathsf{Commit}_r(\overline{M}_\beta)$ to $\mathcal{B}$. The simulator $\mathcal{B}$ runs $\mathsf{Encrypt}(PK, I^*_{enc}) \rightarrow (\mathsf{DK}, C^*)$ and encrypts the message $0^{|M_b \| r|}$ by running $\mathsf{Enc}(\mathsf{NK}, 0^{|M_b \| r|}) \rightarrow \overline{C}^*$, where $\mathsf{NK} \leftarrow \mathsf{Gen}(1^\lambda)$, then sends $CT^* = (I^*_{enc}, C^*, \overline{C}^*, \widehat{C}^*)$ to $\mathcal{A}$ as the challenge ciphertext.
- **Phase 2.** $\mathcal{A}$ adaptively issues private key queries satisfying that $f(I_{key}, I^*_{enc}) \neq 1$ for any queried I_{key}, and $\mathcal{B}$ responds properly.
- **Guess.** $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$. If $b' = b$, $\mathcal{B}$ outputs 0; otherwise, $\mathcal{B}$ outputs 1.

If $\beta = 0$, $\mathcal{B}$ has simulated Game_2 properly; otherwise, $\mathcal{B}$ has properly simulated Game_3. Thus, if the adversary $\mathcal{A}$ has a non-negligible probability to distinguish Game_2 and Game_3, the algorithm $\mathcal{B}$ can attack the computational hiding of the commitment scheme with non-negligible advantage. □

Combining all the above discussions, we complete the proof of Theorem 1. □

V. Generic Construction of VO-ABE

The $\mathsf{BasicABE}$ proposed in Section IV is constructed based on an AB-KEM, a symmetric-key encryption scheme and a commitment scheme. Let $T = (T_1, T_2)$ be an endomorphism transform of which the inverse algorithm is denoted as T', and suppose the decryption algorithm of the AB-KEM satisfies that $\mathsf{Decrypt}(T_2(SK), C) = T_2(\mathsf{Decrypt}(SK, C))$. We construct a VO-ABE scheme based on $\mathsf{BasicABE}$. The algorithms (**Setup**, **KeyGen**, **Encrypt**, **Decrypt**) are the same as in $\mathsf{BasicABE}$. The remaining algorithms are described as follows.

- **GenTK**(PP, SK_I): Parse $SK_I = (I_{key}, SK)$. The transform key generation algorithm runs $T(SK) \rightarrow (\delta, T_2(SK))$, where δ is selected randomly, then sets $TK_I = (I_{key}, T_2(SK)), TSK = \delta$, and returns (TSK, TK_I).
- **Transform**$_{out}(PP, CT, TK_I)$: The transform algorithm parses $CT = (I_{enc}, C, \overline{C}, \widehat{C})$ and $TK_I = (I_{key}, T_2(SK))$, runs $\mathsf{Decrypt}(T_2(SK), C) \rightarrow C'$, and returns $CT' = (C', \overline{C}' = \overline{C}, \widehat{C}' = \widehat{C})$ as the transformed ciphertext.
- **Verify**(PP, CT', TSK): Parse $PP = (PK, \mathsf{KDF}, \ell)$. The verification algorithm runs $T'(TSK, C') \rightarrow \mathsf{DK}$ and $\mathsf{Dec}(\mathsf{KDF}(\mathsf{DK}, \ell), \overline{C}') \rightarrow M \| r$. If $\mathsf{Commit}_r(M) = \widehat{C}'$ holds, it outputs $\mathsf{b} = 1$ and $st = (M, r)$; otherwise, it returns $\mathsf{b} = 0$.
- **Decrypt**$'(PP, CT, CT', TSK)$: The decryption algorithm parses $CT = (I_{enc}, C, \overline{C}, \widehat{C})$ and $CT' = (C', \overline{C}', \widehat{C}')$. If $\overline{C} \neq \overline{C}'$ or $\widehat{C} \neq \widehat{C}'$, the algorithm outputs $\perp$. Otherwise it runs **Verify**(PP, CT', TSK) to obtain b and st. If $\mathsf{b} = 1$, it outputs M. If $\mathsf{b} = 0$, it returns $\perp$.

Correctness. For all $(PP, MSK) \leftarrow$**Setup** $(1^\lambda, U), SK_I \leftarrow$ **KeyGen**(PP, MSK, I_{key}), $CT \leftarrow$ **Encrypt**(PP, M, I_{enc}). Here $PP = (PK, \mathsf{KDF}, \ell), SK_I = (I_{key}, SK)$, $CT = (I_{enc}, C, \overline{C}, \widehat{C})$ where $(\mathsf{DK}, C) \leftarrow \mathsf{Encrypt}(PK, I_{enc})$, $\widehat{C} = \mathsf{Commit}_r(M), \overline{C} \leftarrow \mathsf{Enc}(\mathsf{KDF}(\mathsf{DK}, \ell), M \| r)$. For all $(TSK, TK_I) \leftarrow$ **GenTK**(PP, SK_I) with $TK_I = (I_{key}, T_2(SK))$, $TSK = \delta$, and $CT' \leftarrow$ **Transform**$_{out}(PP, CT, TK_I)$, where $CT' = (C', \overline{C}, \widehat{C})$ and $C' = \mathsf{Decrypt}(T_2(SK), C)$. Consider two cases:

Case 1: $f(I_{key}, I_{enc}) = 1$.

$$C' = \mathsf{Decrypt}(T_2(SK), C) = T_2(\mathsf{Decrypt}(SK, C)) = T_2(\mathsf{DK}).$$

Here the second equality holds for the assumed property of the decryption algorithm of the AB-KEM. Since T' is the inverse algorithm of T, it holds that

$$T'(TSK, C') = T'(\delta, C') = T'(\delta, T_2(\mathsf{DK})) = \mathsf{DK}.$$

Then M and r are generated from $\mathsf{Dec}(\mathsf{KDF}(\mathsf{DK}, \ell), \overline{C}) \rightarrow M \| r$. If $\mathsf{Commit}_r(M) = \widehat{C}$, **Verify**$(PP, CT', TSK)$ returns 1 and **Decrypt**$'(PP, CT, CT', TSK)$ outputs M.

Case 2: $f(I_{key}, I_{enc}) \neq 1$. **Decrypt**$(PP, SK_I, CT)$ and **Decrypt**$'$ (PP, CT, CT', TSK) output $\perp$.

Our construction requires that the decryption algorithm of the AB-KEM has some homomorphic property, i.e., for an endomorphism transform $T = (T_1, T_2)$, $\mathsf{Decrypt}(T_2(SK), C) = T_2(\mathsf{Decrypt}(SK, C))$. If we consider an endomorphism transform defined on a multiplicative group as described in Section II, this homomorphic property is exactly multiplicative homomorphism. Specifically, if the transform for a private key SK is defined as $T_2(SK) = SK^z$ by a random value z, it holds that $\mathsf{Decrypt}((SK)^z, C) = (\mathsf{Decrypt}(SK, C))^z$. Actually, most

existing pairing-based AB-KEMs (KP or CP) satisfy the property of multiplicative homomorphism. Thus, our technique can be applied to most existing AB-KEMs in both KP and CP settings.

We have the following security result of the above generic construction of VO-ABE.

Theorem 2: Assume that BasicABE *is IND-CPA secure and the commitment scheme* (Commit, Decommit) *is computationally binding. Then the above constructed VO-ABE scheme is IND-CPA secure and meets the verification soundness.*

Proof: We first prove that IND-CPA security of BasicABE implies IND-CPA security of the VO-ABE scheme.

Suppose there exists a PPT adversary $\mathcal{A}$ that has a non-negligible advantage to attack IND-CPA security of the VO-ABE scheme. We can build an algorithm $\mathcal{B}$ to attack IND-CPA secure BasicABE with non-negligible advantage. The simulator $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine in the following steps.

- **Setup.** $\mathcal{B}$ forwards PP to $\mathcal{A}$ from the challenger.
- **Phase 1.** $\mathcal{A}$ adaptively issues private key and transform key queries. $\mathcal{B}$ answers private key queries by using its own key generation oracle. For any transform key query on I_{key}, $\mathcal{B}$ calls $\mathsf{Setup}(1^\lambda, U) \rightarrow (PK', MSK')$, $\mathsf{KeyGen}(PK', MSK', I_{key}) \rightarrow SK'$, and runs $T(SK') \rightarrow (\delta', T_2(SK'))$, then returns to $\mathcal{A}$ the simulated transform key $TK_I = (I_{key}, T_2(SK'))$.
- **Challenge.** $\mathcal{A}$ submits (equal length) M_0, M_1 and I^*_{enc} where $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} in private key queries above. $\mathcal{B}$ forwards them to the challenger and obtains CT^*. Then $\mathcal{B}$ sends CT^* to $\mathcal{A}$ as the challenge ciphertext.
- **Phase 2.** $\mathcal{A}$ continues to adaptively issue private key and transform key queries with the restriction that $f(I_{key}, I^*_{enc}) \neq 1$ for any I_{key} in private key queries. $\mathcal{B}$ answers the private key queries as in Phase 1. For any transform key query on I_{key}, if $f(I_{key}, I^*_{enc}) \neq 1$, $\mathcal{B}$ obtains the private key SK from its key generation oracle and runs $T(SK) \rightarrow (\delta, T_2(SK))$, then returns $TK_I = (I_{key}, T_2(SK))$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ responds as in Phase 1.
- **Guess.** $\mathcal{A}$ outputs its guess b' of b. Then $\mathcal{B}$ outputs b'.

Since the public output $T_2(\cdot)$ of the endomorphism transform T is computationally indistinguishable between different inputs, $\mathcal{A}$ can distinguish the simulated transform key from the real one with at most negligible probability. Thus, except with negligible probability, the algorithm $\mathcal{B}$ has perfectly simulated the IND-CPA security game of the constructed VO-ABE scheme for the adversary $\mathcal{A}$. If $\mathcal{A}$ can win the IND-CPA security game of the VO-ABE scheme with non-negligible advantage, $\mathcal{B}$ can attack the IND-CPA secure BasicABE with non-negligible advantage.

Below we show that IND-CPA security of BasicABE and the computational binding of the commitment scheme (Commit, Decommit) imply the verification soundness of the VO-ABE scheme.

Assume there is a PPT adversary $\mathcal{A}$ that can win the verifiable game of the VO-ABE scheme with non-negligible probability. We can construct an algorithm $\mathcal{B}$ to attack the computational binding of the commitment scheme with non-negligible probability. Let $\mathcal{R}$ be the receiver and $\mathcal{B}$ is the sender in the commitment scheme. The simulator $\mathcal{B}$ runs $\mathcal{A}$ in the following steps.

- **Setup.** $\mathcal{B}$ calls the setup algorithm of the constructed VO-ABE scheme to obtain (PP, MSK) and sends PP to $\mathcal{A}$.
- **Phase 1.** $\mathcal{A}$ is given access to $\mathcal{O}^{\text{KeyGen}}(\cdot)$,$\mathcal{O}^{\text{GenTK}}(\cdot)$, $\mathcal{O}^{\text{Decrypt}}(\cdot,\cdot)$, $\mathcal{O}^{\text{Decrypt}'}(\cdot,\cdot,\cdot)$. Since $\mathcal{B}$ knows the master secret key MSK, it is able to provide all these oracles.
- **Challenge.** $\mathcal{A}$ submits a message M^* and I^*_{enc}. $\mathcal{B}$ encrypts the message M^* with I^*_{enc} by the encryption algorithm of the VO-ABE scheme to obtain $CT^* = (I^*_{enc}, C^*, \overline{C}^*, \widehat{C}^*)$, where $\widehat{C}^* = \mathsf{Commit}_r(M^*)$. Then $\mathcal{B}$ returns CT^* to $\mathcal{A}$ and sends $\widehat{C}^*$ to $\mathcal{R}$.
- **Phase 2.** $\mathcal{A}$ is given access to the oracles as in Phase 1.
- **Forge.** $\mathcal{A}$ outputs I^*_{key} and a transformed ciphertext $CT^{*\prime} = (C^{*\prime}, \overline{C}', \widehat{C}')$ where $\overline{C}' = \overline{C}^*$, $\widehat{C}' = \widehat{C}^*$. Note that $\mathcal{A}$ has issued I^*_{key} to $\mathcal{O}^{\text{GenTK}}(\cdot)$ and obtains the transform key TK_I. $\mathcal{B}$ owns the corresponding TSK. If $\mathbf{Verify}(PP, CT^{*\prime}, TSK)$ returns 1 and (M', r'), $\mathcal{B}$ outputs (M', r'); otherwise, $\mathcal{B}$ outputs a random tuple.

The IND-CPA security of BasicABE guarantees that for any PPT adversary $\mathcal{A}$, the random r used to commit to M^* is computationally indistinguishable from a uniformly distributed random variable. Let $\Pr[\mathcal{A}\ \textit{forges}]$ denote the probability for $\mathcal{A}$ to win the game of verification soundness. Let $\Pr[\mathcal{B}\ \textit{succeeds}]$ be the probability that $\mathcal{B}$ outputs (M', r') satisfying $M' \neq M^*$ and

$$\mathsf{Commit}_{r'}(M') = \widehat{C}^* = \mathsf{Commit}_r(M^*).$$

Denote Q as the space of all the commitment values. Then $|\mathsf{Q}| \in \Theta(2^\lambda)$. We have

$$\begin{aligned}\Pr[\mathcal{B}\ \textit{succeeds}] &= \Pr[\mathcal{A}\ \textit{forges}] + (1 - \Pr[\mathcal{A}\ \textit{forges}]) \cdot \frac{1}{|\mathsf{Q}|} \\ &\geq \Pr[\mathcal{A}\ \textit{forges}] \left(1 - \frac{1}{|\mathsf{Q}|}\right).\end{aligned}$$

Suppose that $\mathcal{A}$ wins the game with non-negligible probability, i.e., $\Pr[\mathcal{A}\ \textit{forges}]$ is non-negligible. Then $\Pr[\mathcal{B}\ \textit{succeeds}]$ is non-negligible, which contradicts to computational binding of the commitment scheme. □

Remark 1: Note that the security of the resulting VO-ABE scheme depends on the security of the building blocks, so if the underlying AB-KEM is selectively/adaptively secure, the resulting VO-ABE scheme is also selectively/adaptively secure. Namely, security property will be actually inherited by the resulting VO-ABE scheme. We remark that one should choose the proper endomorphism transform accordingly, which can be implemented on groups of either prime or composite orders.

VI. Our Instantiation

In this section, we instantiate our generic construction with an AB-KEM [6], the one-time pad and the Pedersen commitment scheme [27]. For simplicity and for a fair comparison with the LDGW-scheme [3], we only present a concrete VO-ABE scheme based on the selectively secure AB-KEM [6] and just discuss CP-ABE here. The same technique is available

for KP-ABE since our construction does not require the underlying ABE to be KP or CP, as explained in Section V. We note that one can adapt our technique to an adaptively secure AB-KEM [7] and achieves an adaptively secure VO-ABE scheme.

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p, where $p \in \Theta(2^\lambda)$. We use a specific endomorphism transform $T : \mathbb{X} \to \mathbb{Z}_p^* \times \mathbb{X}$ defined on a cyclic group $\mathbb{X}$ of prime order p, where a detailed description is provided in Section II. Our concrete VO-ABE scheme is described as follows.

- **Setup**$(1^\lambda, U)$: Choose $g, h, w \xleftarrow{R} \mathbb{G}$ and $\alpha, a \xleftarrow{R} \mathbb{Z}_p^*$. Select random values $\{s_i \in \mathbb{Z}_p^* : \forall i \in U\}$ and a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$. Adopt the key derivation function KDF with the output length ℓ and finally set $PP = (g,\ h,\ w,\ g^a,\ e(g,g)^\alpha,\ \{T_i = g^{s_i} : \forall i \in U\}, H, \mathsf{KDF}, \ell)$, $MSK = \alpha$. Return $(PP,\ MSK)$.
- **KeyGen**$(PP,\ MSK,\ S)$: Pick $t \xleftarrow{R} \mathbb{Z}_p^*$. The private key $SK = (K = g^\alpha g^{at},\ K_0 = g^t,\ K_i = T_i^t,\ \forall i \in S)$. Return $SK_I = (S, SK)$.
- **Encrypt**$(PP,\ M \in \{0,1\}^*, \mathbb{A})$: Let $\mathbb{A} = (\mathbf{A}, \rho)$ be a Linear Secret Sharing Scheme (LSSS) access structure, where $\mathbf{A}$ is an $m \times n$ matrix and $\rho : [m] \to U$ is a map from each row A_i of $\mathbf{A}$ to an attribute $\rho(i)$. Choose a random vector $\vec{v} = (s, v_2, \cdots, v_n) \in \mathbb{Z}_p^{*n}$. Select $r_i \xleftarrow{R} \mathbb{Z}_p^*$, $i \in [m]$. The ciphertext of the AB-KEM is

  $$C = (E = g^s,\ C_i = g^{aA_i \cdot \vec{v}} T_{\rho(i)}^{-r_i},\ D_i = g^{r_i},\ \forall i \in [m])$$

  and the session key is $\mathsf{DK} = e(g,g)^{\alpha s}$. Choose a random $r \in \{0,1\}^*$, and set $\widehat{C} = h^{H(M)} w^{H(r)}$, $\overline{C} = (M \parallel r) \oplus \mathsf{KDF}(\mathsf{DK}, \ell)$. The ciphertext is $CT = (\mathbb{A},\ C,\ \overline{C},\ \widehat{C})$.
- **Decrypt**$(PP,\ SK_I,\ CT)$: If S does not satisfy $\mathbb{A}$, output $\perp$; otherwise, there exist $\omega_i \in \mathbb{Z}_p^*$ such that $\sum_{\rho(i) \in S} \omega_i A_i = (1, 0, \cdots, 0)$. Compute

  $$\mathsf{DK} = \frac{e(E, K)}{\Pi_{\rho(i) \in S} (e(C_i, K_0) \cdot e(K_{\rho(i)}, D_i))^{\omega_i}} = e(g,g)^{\alpha s}$$

  and $\overline{C} \oplus \mathsf{KDF}(\mathsf{DK}, \ell) = M \parallel r$. If $h^{H(M)} w^{H(r)} = \widehat{C}$, output the message M; otherwise, output $\perp$.
- **GenTK**$(PP,\ SK_I)$: Run the endomorphism transform T on SK and obtain $T(SK) = (z,\ T_2(SK))$, where $z \xleftarrow{R} \mathbb{Z}_p^*$ and the public output $T_2(SK) = SK^z = (K^z,\ K_0^z,\ K_i^z,\ \forall i \in S)$. Set the transform key $TK_I = (S,\ K' = K^z,\ K_0' = K_0^z,\ K_i' = K_i^z,\ \forall i \in S)$ and $TSK = z$. Return (TSK, TK_I).
- **Transform**$_{out}(PP,\ CT,\ TK_I)$: If S does not satisfy $\mathbb{A}$, output $\perp$; otherwise, there exist $\omega_i \in \mathbb{Z}_p^*$ such that $\sum_{\rho(i) \in S} \omega_i A_i = (1, 0, \cdots, 0)$. Compute

  $$C' = \frac{e(E, K')}{\Pi_{\rho(i) \in S} (e(C_i, K_0') \cdot e(K'_{\rho(i)}, D_i))^{\omega_i}} = e(g,g)^{\alpha s z},$$

  and output the transformed ciphertext as $CT' = (C', \overline{C}' = \overline{C}, \widehat{C}' = \widehat{C})$.
- **Verify**$(PP,\ CT',\ TSK)$: By using the inverse transform T', we obtain

  $$T'(TSK, C') = C'^{(1/TSK)} = C'^{(1/z)} = e(g,g)^{\alpha s} = \mathsf{DK}.$$

  Compute $\overline{C}' \oplus \mathsf{KDF}(\mathsf{DK}, \ell) = M \parallel r$. If $h^{H(M)} w^{H(r)} = \widehat{C}'$ holds, output $\mathsf{b} = 1$ and $st = (M, r)$; otherwise, output $\mathsf{b} = 0$.
- **Decrypt**$'(PP, CT, CT', TSK)$: Parse $CT = (\mathbb{A}, C, \overline{C}, \widehat{C})$ and $CT' = (C', \overline{C}', \widehat{C}')$. If $\overline{C} \neq \overline{C}'$ or $\widehat{C} \neq \widehat{C}'$, output $\perp$; otherwise, run **Verify**(PP, CT', TSK) to obtain b and st. If $\mathsf{b} = 1$, return M. If $\mathsf{b} = 0$, return $\perp$.

TABLE I
EXPONENTIAL AND PAIRING OPERATIONS

Scheme	Algorithm	$\mathbb{G}$	$\mathbb{G}_T$	pairings
Ref.[3]	Encrypt	$2(3m+1)+2$	2	0
	Transform$_{out}$	0	$2m'$	$2(2m'+1)$
	Decrypt$_{out}$	2	2	0
Our scheme	Encrypt	$3m+3$	1	0
	Transform$_{out}$	0	m'	$2m'+1$
	Decrypt$'$	2	1	0

m, m' $(m' \leq m)$ denote the number of rows of the secret-sharing matrix and the rows used during decryption, respectively.

Suppose the decisional q-parallel BDHE assumption [6] holds. The AB-KEM used in the above construction is selectively IND-CPA secure. Under the Discrete-Logarithm assumption, the Pedersen commitment scheme is perfectly hiding and computationally binding. According to Theorem 1 and Theorem 2, we have

Theorem 3: Suppose that the decisional q-parallel BDHE assumption holds and the key derivation function KDF is secure. Then the above constructed ABE scheme with outsourced decryption is selectively IND-CPA secure and meets the verification soundness.

VII. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our instantiation of CP-ABE scheme with verifiable outsourced decryption. For comparing the efficiency of our scheme to the LDGW-scheme [3], we tabulate the exponential and pairing operations in Table I.

As shown in Table I, our scheme only requires nearly half of the modular exponentiations used in the LDGW-scheme [3] during the encryption phase and needs one less modular exponentiation in $\mathbb{G}_T$ than [3] during the final decryption phase. The computation cost for the third-party service to transform a standard ABE ciphertext in our scheme is half of that in [3].

For further comparison of the efficiency, we implement the schemes in Charm [28] with a 224-bit MNT elliptic curve from the Stanford Pairing-Based Crypto library [29]. Our implementation uses a laptop with a 2.60 GHz Intel Core i5-3320M CPU and 4GB RAM running 64-bit Ubuntu 14.04. The programming language is Python 3.4. The key derivation function used here is KDF1 defined in ISO-18033-2 [30]. We compare the complexity of our proposed ABE scheme with the LDGW-scheme [3] in the size of a ciphertext, the encryption time, the transform time and the time of decrypting a transformed ciphertext. In order to illustrate how the complexity of ciphertext policy affects on the performance, we generate 100 different policies in the form of (a_1 and a_1 and $\cdots$ and a_N) as in [3], where each a_i

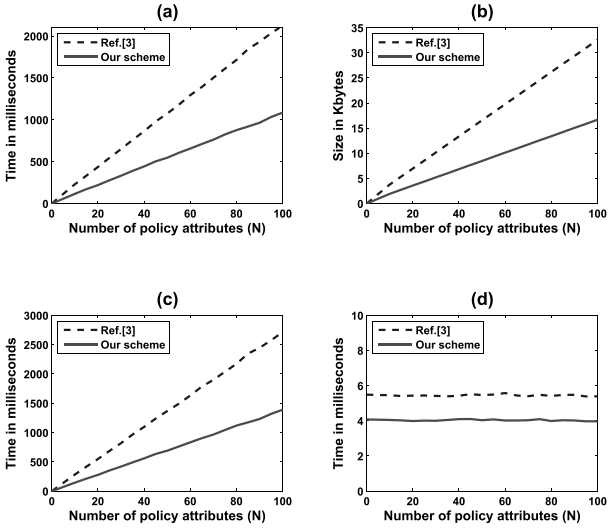

Fig. 1. Performance Comparisons. (a) ABE Encryption Time. (b) ABE Ciphertext Size. (c) Transform Time. (d) Final Decryption Time.

is an attribute and $1 \leq N \leq 100$. We repeat the experiment 100 times and take the average values as the final results.

Fig. 1 (a) and (b) show that under the same policy, the encryption time and the size of a standard ABE ciphertext of our scheme are far less than [3]. Specifically, a message encrypted under a policy with 100 attributes needs around 1 second and the size of the generated ciphertext is around 16.68 KB, which are both nearly half of the counterparts in [3]. Fig. 1 (c) illustrates that the third-party service for our scheme just needs half of the time used in [3] to transform a standard ABE ciphertext. For Fig. 1 (d), the final decryption time is slightly more than half of that in [3]. All our experimental results are coincident with the theoretic analysis presented above.

Therefore, we conclude that our instantiation of ABE with verifiable outsourced decryption is more efficient than the existing scheme [3].

ACKNOWLEDGMENT

The authors thank Junfeng Fan for helpful discussions and the anonymous reviewers from many valuable comments.

REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.
[2] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Secur. Symp.*, 2011, p. 34.
[3] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.
[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, May 2007, pp. 321–334.
[5] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 456–465.
[6] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Public Key Cryptography*, 2011, pp. 53–70.
[7] A. 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.
[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
[9] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 195–203.
[10] R. L. Rivest, "All-or-nothing encryption and the package transform," in *Proc. 4th Int. Workshop Fast Softw. Encryption*, 1997, pp. 210–218.
[11] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai, "Exposure-resilient functions and all-or-nothing transforms," in *Proc. EUROCRYPT*, 2000, pp. 453–469.
[12] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Ràfols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theoretical Comput. Sci.*, vol. 422, pp. 15–38, Mar. 2012.
[13] S. Hohenberger and B. Waters, "Attribute-based encryption with fast decryption," in *Proc. Public-Key Cryptography*, 2013, pp. 162–179.
[14] B. Chevallier-Mames, J.-S. Coron, N. McCullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Proc. 9th Int. Conf. Smart Card Res. Adv. Appl.*, 2010, pp. 24–35.
[15] B. G. Kang, M. S. Lee, and J. H. Park, "Efficient delegation of pairing computation," Cryptol. ePrint Arch., Rep. 2005/259, 2005. [Online]. Available: http://eprint.iacr.org/
[16] P. P. Tsang, S. S. Chow, and S. W. Smith, "Batch pairing delegation," in *Proc. 2nd Int. Workshop Adv. Inf. Comput. Secur.*, 2007, pp. 74–90.
[17] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. EUROCRYPT*, 1998, pp. 127–144.
[18] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, Feb. 2006.
[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. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Proc. CRYPTO*, 2010, pp. 483–501.
[21] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou, "Fine-grained access control system based on outsourced attribute-based encryption," in *Proc. 18th Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2013, pp. 592–609.
[22] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2013.
[23] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Proc. Public-Key Cryptography*, 2014, pp. 293–310.
[24] J. Katz and Y. Lindell, *Introduction to Modern Cryptography* (Chapman & Hall/CRC Cryptography and Network Security Series). London, U.K.: Chapman & Hall, 2007.
[25] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme," in *Proc. CRYPTO*, 2010, pp. 631–648.
[26] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proc. 2nd Theory Cryptography Conf.*, 2005, pp. 325–341.
[27] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. CRYPTO*, 1991, pp. 129–140.
[28] J. A. Akinyele *et al.*, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, 2013.
[29] B. Lynn, *The Stanford Pairing Based Crypto Library*. [Online]. Available: http://crypto.stanford.edu/pbc
[30] V. Shoup, *Information Technology—Security Techniques—Encryption Algorithms—Part 2: Asymmetric Ciphers*, document ISO/IEC 18033-2, 2004.

Suqing Lin received the B.S. degree from Wenzhou Normal College, and the M.S. degree from Sichuan Normal University, China. She is currently pursuing the Ph.D. degree in information security with the Institute of Information Engineering, Chinese Academy of Sciences. After receiving the M.S. degree, she was an Assistant and a Lecturer with the City College of Wenzhou University. Her research interests include cryptography and information security.

Rui Zhang received the B.E. degree from Tsinghua University, and the M.S./Ph.D. degrees from The University of Tokyo. He was a JSPS Research Fellow. He joined National Institute of Advanced Industrial Science and Technology, Japan, as a Research Scientist. He is currently with the Institute of Information Engineering, Chinese Academy of Sciences, as a Research Professor. His research interests include applied cryptography, network security, and information theory.

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

Mingsheng Wang received the Ph.D. degree from Beijing Normal University, China, in 1994. He is currently a Research Professor with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing. His current research interests lie in computation algebra, cryptography, and information security.