Received December 1, 2018, accepted December 26, 2018, date of publication January 1, 2019, date of current version March 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2890565

An Efficient ABE Scheme With Verifiable Outsourced Encryption and Decryption

ZHIDAN LI, WENMIN LI, ZHENGPING JIN, HUA ZHANG, AND QIAOYAN WEN
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Wenmin Li (liwenmin02@outlook.com)

This work was supported by NSFC under Grant 61502044.

ABSTRACT Attribute-based encryption (ABE) is a promising cryptographic tool for data owner (DO) to realize fine-grained date sharing in the cloud computing. In the encryption of most existing ABE schemes, a substantial number of modular exponentiations are often required; the computational cost of it is growing linearly with the complexity of the access policy. Besides, in the most existing ABE with outsourced decryption, the computation cost of generating transformation key is growing linearly with the number of attributes associated with user private key; these computations are prohibitively high for mobile device users, which becomes a bottleneck limiting its application. To address the above issues, we propose a secure outsourcing algorithm for modular exponentiation in one single untrusted server model and a new method to generate the transformation key. Based on these techniques and Brent Waters's ciphertext-policy ABE scheme, we propose an ABE scheme with verifiable outsourced both encryption and decryption, which can securely outsource encryption and decryption to untrusted encryption service provider (ESP) and decryption service provider (DSP), respectively, leaving only a constant number of simple operations for the DO and eligible users to perform locally. In addition, both DO and the eligible users can check the correctness of results returned from the ESP and the DSP with a probability, respectively. Finally, we provide the experimental evaluation and security analysis of our scheme, which indicates that our construction is suitable for the mobile environment.

INDEX TERMS ABE, access control, secure outsourcing algorithm, modular exponentiation, verifiability, outsourced encryption, outsourced decryption.

I. INTRODUCTION

In the network, it frequently happens that sensitive data must be archived by storage server in a way that the specific users could be allowed to access it. Besides, with the development of cloud computing, data security is the main obstacle that impedes cloud computing from wide adoption. The reason is that user's data is stored in a public cloud, which is maintained and operated by untrusted cloud service provider. To address this problem, Sahai and Waters [1] proposed the access model such as ABE, which is a mechanism that enables data owner(DO) to encrypt data using access polices to realize fine-grained data sharing in the cloud computing. ABE schemes are divided into two categories: ciphertext-policy ABE(CP-ABE) [2], [3] and key-policy ABE(KP-ABE) [4]. In CP-ABE, the access policy is embedded into the ciphertext, while in the KP-ABE, the access policy assigned in private key. In CP-ABE, a user is able to decrypt a ciphertext if only if the set of attributes associated with the user's the private key satisfies the access policy associated with the ciphertext. In KP-ABE, a user is able to decrypt a ciphertext if only if the set of attributes associated with the ciphertext satisfies the access policy associated with the user's the private key.

Nevertheless, one of the main efficiency drawbacks of the most exiting ABE schemes is that the computational cost in the encryption and decryption phases is expensive. In the most existing ABE schemes, the encryption operations are mainly modular exponentiations, which grows with the complexity of the access policy. In the mobile cloud computing [5] and fog computing [6], [7], mobile device has limited computation ability to independently complete basic encryption to protect sensitive data residing in public cloud. Considering an application scenario that a DO wants to share his data to specific person without a desktop at hand, how to complete the encryption effectively. To address this issue, outscourced ABE, which provides a way to outsource intensive computing task during encryption to server provider without revealing data was introduced in [8] and [9].

Beyond the encryption outsourced, in the most existing ABE schemes, the decryption computation are mainly pairing operations, which also grows with the complexity of the access policy, Green *et al.* [10] proposed a remedy to this problem by introducing the notion of ABE with outsourced decryption, which largely eliminates the decryption overhead for users. However, in the most existing ABE schemes with outsourced decryption [10]–[12], the user has to deal with a number of computations when he wants to outsource the decryption to a decryption service provider(DSP). More precisely, the user has to generate a transformation key to DSP, but yet generation of the transformation key requires certain modular exponentiations computation, which grows linearly with the number of attributes associated with user private key. Moreover, in most of existing ABE schemes, if one attribute of user is revoked or added at any time, thus the private key requires key-update at user, then the transformation key has to be updated simultaneously, while the computation tasks cannot be performed effectively by resource-constrained mobile users. All of these heavy tasks on DO or user side would make it an efficiency bottleneck in the access control system.

A. OUR CONTRIBUTION

Aim at eliminating the most overhead computation on both the data owner and the user side, we propose an outsourced ABE scheme not only supporting outsourced decryption but also outsourced encryption. Firstly, we propose a secure outsourcing algorithm for modular exponentiation in one untrusted server model, which can allow user to securely outsource variable-exponent variable-base exponentiations(u^a or $u_1^a u_2^b$) to just one untrusted computation server, and user can efficiently check whether the computation is computed correctly or not. Then based on this type of algorithm, we present an ABE with verifiable outsourced encryption, it not only shifts the burdensome modular exponentiation computation task from DO to ESP, but also DO can check the correctness of the calculation done by ESP.

In addition, we consider the outsourced decryption in the proposed ABE with verifiable outsourced encryption, we propose a new approach which can reduce the transformation key generation cost to be constant. Above all, we present an $\underline{\text{ABE}}$ with $\underline{\text{v}}$erifiable $\underline{\text{o}}$utsourced both $\underline{\text{e}}$ncryption and $\underline{\text{d}}$ecryption(ABE-VOED) which can shift the burdensome encryption and decryption computation task of DO and user to ESP and DSP, respectively. Following our techniques, computation efficiency is achieved at both the DO and user side.

At last, we implement the ABE-VOED to evaluate the efficiency of our scheme and present the proof of our scheme. The software used by us is based on the CP-ABE implementation in the libfenc library [13]. The comparison of performance between other ABE schemes and ABE-VOED given in section VI indicates that our ABE scheme is more suitable for the mobile environment.

B. RELATED WORK

1) ABE WITH OUTSOURCED COMPUTATION

As mentioned above, due to the decryption computation cost grows linearly with the complexity of the access policy in general ABE schemes [14]–[17]. To address this problem, previous work focus on how to reduce the computation burden of the users, Attrapadung *et al.* [18] proposed a scheme which has short ciphertext so as to reduce the cost of the decryption, while Green *et al.* [10] first proposed shift the decryption task from user to a server, then user can decrypt the ciphertext by only do a little amount of multiplication calculations. In fact, it delegates all the pairing operations to a decryption service provider(DSP). Lai *et al.* [11] and Lin *et al.* [12] considered the verifiability of the cloud ŕs transformation, then they proposed a method to check the correctness of the transformation and presented concrete ABE scheme with verifiable outsourced decryption, respectively. Nevertheless, in the above ABE schemes with outsouorced decryption, the computation cost of transformation key is growing with the the number of the attributes associated with user's private key, this is a computational burden for mobile device user.

In general, most existing ABE schemes require a large number of modular exponentiations on the DO side in the encryption phase. In order to reduce the computing cost on the user side, especially mobile user. How to securely outsource this kind of expensive computations has drew considerable attention from theoretical researchers. The methods in [19]–[21] focus on how to securely outsource algebra computations. Another method is the fully homomorphic encryption(FHE), however, even if the privacy of the input and output can be preserved by it, the computational overhead is still huge and impractical. They are not suitable for reliving ABE computational overhead of exponentiation at DO side. Li *et al.* [8] and Zhou and Huang [9] proposed an outsourced ABE construction with delegated encryption, but they cannot verify the correctness of calculation results done by cloud service provider.

2) SECURELY OUTSOURCING EXPONENTIATIONS

The research of the securely outsourcing computations in the cryptographic community has been drawing widespread attention. Hehenberger and Lysyanskaya [22] are the first to propose the secure outsourcing algorithm for modular exponentiation based on the two previous approaches of precomputation [23], [24] and server-aided computation [19], [25]. The outsourcing algorithms of variable-exponent fixed-base and the fixed-exponent varies-base exponentiation in one untrusted server model are considered in [25] and [26], which presented an algorithm of outsourcing these two types of exponentiations in the two non-collusion untrusted model. Since the servers are not trusted by the outsourcers, then the outsource-secure algorithms of variable-exponent variable-base exponentiations in one-malicious version of two program model are proposed by [27], in which one server is trusted, and will not collude with the other dishonest one. Wang *et al.* [28] first proposed a secure outsourcing algorithm for variable-exponent variable-base multi-exponentiations in one untrusted computation server model, in which the probability of verifying correctness needs to be improved.

C. ORGANIZATION

Our paper is organized as follows. Preliminaries are provided in section II. In section III, we present a new secure outsourcing algorithm of modular exponentiations and its security analysis. The formal definition and the security model for our scheme is given in section IV. We present concrete ABE Scheme with Verifiable Outsourced both Encryption and Decryption(ABE-VOED) in section V. We also present the security analysis and the experimental evaluation in the section VI. Finally, a conclusion will be presented in section VII.

II. PRELIMINARIES

A. BILINEAR MAPS

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, g is a generator of the group $\mathbb{G}$, A bilinear mapping e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ satisfies following properties:

- Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$, $\forall g \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}_p^*$.
- Nondegeneracy:$e(g, g) \neq 1$.
- Computability: there is an efficient algorithm to compute $e(g^a, g^b)$, $\forall g \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}_p^*$

B. ACCESS STRUCTURES [29]

Let $\mathcal{P} = \{P_1, P_2 ... P_N\}$, A collection $\mathbf{A} \subseteq 2^{\mathcal{P}}$ is monotone if $\forall \Theta, \Theta'$, if $\Theta' \in \mathbf{A}$ and $\Theta' \subseteq \Theta$, then $\Theta \in \mathbf{A}$. An access structure (respectively, monotone structure) is a collection (respectively, monotone collection) $\mathbf{A}$ of non-empty subsets of $\{P_1, P_2 ... P_n\}$, *i.e.*, $\mathbf{A} \subseteq 2^{\mathcal{P}} \backslash \{\emptyset\}$. The sets in $\mathbf{A}$ are called the authored sets, and the sets not in $\mathbf{A}$ are called unauthorized sets.

C. DECISIONAL PARALLEL BILINEAR DIFFE-HELLMAN EXPONENT ASSUMPTION [3]

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

$$\begin{aligned} y = g, g^s, g^a, ..., g^{a^q}, g^{a^{q+2}}, ..., g^{a^{2q}} \\ \forall 1 \leq j \leq q g^{sb_j}, g^{a/b_j}, ..., g^{a^q/sb_j}, g^{a^{q+2}/sb_j}, ..., g^{a^2 q/sb_j}, \\ \forall_{1 \leq j,k \leq q, k \neq j} g^{a \cdot s \cdot b_k/b_j}, ..., g^{a^q \cdot s \cdot b_k/b_j} \end{aligned}$$

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

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

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

Definition 1: We say that the (decision) q-parallel *BDHE* assumption holds if no polynomial time algorithm has a non-negligible advantage in solving the decisional q-parallel *BDHE* problem.

D. LINEAR SECRET SHARING SCHEMES (LSSS)

A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear(over $\mathbb{Z}_p$) if:

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

As shown in [32], the *LSSS* has the property of *linear reconstruction*, it is to say if $S \in \mathbf{A}$, then there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$, s.t $\Sigma_{i \in I} \omega_i \lambda_i = s$, where $\{\lambda_i\}$ are valid shares of any secret s according to Π, $I = \{i : \rho(i) \in S\}$.

As described in [3], if an unauthorized set of S, the target vector (1,0,0...0) is not in the span of the rows of the set I, Moreover, it exist a vector ω satisfies that $\omega \cdot (1, 0, 0...0)$ and $\omega \cdot M_i$=0 for all $i \in I$.

E. DEFINITION OF OUTSOURCE-SECURITY

In this section, we review the formal definitions for secure outsourcing of a cryptographic algorithm [27], [28].

An algorithm Alg to be outsourced is divided into two parts, a trusted part T which is a carried by a outsourcer, and an untrusted part U which is invoked by T. T^U denotes that T implements Alg by invoking U. Supposing that T is given oracle access to an adversary U' (instead of U), the adversary U' can not learn anything interesting about the input and output of $T^{U'}$, Following the same notations in [27] and [28], we introduce the formal definitions for secure outsourcing of a cryptographic algorithm.

Definition 2 (Algorithm With Outsource-I/O): For an algorithm Alg takes five inputs and produces three outputs, then Alg obeys the outsource input/output specification, i.e. $Alg(x_1, x_2, x_3, x_4, x_5) \rightarrow (y_1, y_2, y_3)$. The first three inputs are generated by an honest party and classified by how much the adversary A=(E, U') knows about them, where E represents the adversary environment and generates adversarial inputs for Alg, while U' is an adversarial software in stead of U during the execution of T^U. The first input x_1 is called the honest, secret input, which is unknown to both E and U'; the second input x_2 is called honest, protected input, which may be known by E, but is protected from U'; the third input x_3 is called honest, unprotected input, which may be known by both E and U'. Besides, the last two inputs are adversarial-chosen inputs generated by the environment E, the fourth input x_4 is called adversarial, protected input, which is known to E, but protected from U', the last input x_5

is called adversarial, unprotected input, which may be known to both E and U'. Similarly, the outputs are classified by how much the adversary $A = (E, U')$ knows about them, the first output called secret is unknown to both E and U'; the second is protected, which may be known to E but not U'; the last one is unprotected which could be known by both E and U'.

It notes that the adversary A consists of two parties E and U'. Both can only make direct communications before the execution of T^U. In any other cases if necessary, they should be communicated via T. An algorithm Alg is outsource-secure if neither party of A=(E, U') can learn anything interesting during execution of T^U, the requirement is captured by the simulatability of (T, U). That is to say for any probabilistic polynomial time adversary A=(E, U'), the view of U' on the execution of T^U can be simulated in a computationally indistinguishable way given the unprotected inputs, and the view of E can also be simulated given the protected and the unprotected inputs.

Definition 3(Outsource-Security [27], [28]): Let Alg be an algorithm with outsource-I/O. A pair of algorithm (T, U) is said to be an outsource-secure implementation of Alg if:

1. Correctness: T^U is a correct implementation of Alg.
2. Security: For all probability polynomial time adversary $A = (E, U')$, there exist probabilistic expected polynomial-time simulators (SIM_E,$SIM_{U'}$) such that the following pairs of random variables are computationally indistinguishable.

- Pair One $EVIEW_{real} \sim EVIEW_{ideal}$
  -Real process as follows:

$$\begin{aligned}EVIEW_{real} = \{&(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu} \longleftarrow I(1^k, istate^{i-1});\\&(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i)\\&\longleftarrow E(1^k, EVIEW^{i-1}_{real}, x^i_{hp}, x^i_{hu});\\&(tstate^i, ustate^i, y^i_s, y^i_p, y^i_u)\\&\longleftarrow T^{U'(ustate^{i-1})}\\&(tstate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}):\\&(estate^i, y^i_p, y^i_u)\}\end{aligned}$$

  $EVIEW_{real} \sim EVIEW^i_{real}$ if $stop^i = TRUE$.
  The real process proceeds in rounds. In round i, the honest (secret, protected, and unprotected) inputs(x^i_{hs}, x^i_{hp}, x^i_{hu}) are picked using an honest, stateful process I to which the environment E does not have access. Then E, based on its *view* from the last round, chooses
  1) The value of its $estate_i$ variable as a way of remembering what it did next time it is invoked;
  2) Which previously generated honest inputs (x^i_{hs}, x^i_{hp}, x^i_{hu}) to give to $T^{U'}$ (note that E can specify the index j^i of these inputs, but not their values);
  3) The adversarial, protected input x^i_{ap};
  4) The adversarial unprotected input x^i_{au};
  5) The Boolean variable $stop^i$ that determines whether round i is the last round in this process

  Next the algorithm $T^{U'}$ is running on the inputs ($tstate^{i-1}$, $x^{j^i}_{hs}$, $x^{j^i}_{hp}$, x^i_{ap}, x^i_{au}), where $tstate^{i-1}$ is T's previously saved state, and produces a new state $tstate^i$ for T, as well as the secret y^i_s, protected y^i_p and unprotected y^i_u outputs. The oracle U' is given its previously saved state, $ustate^{i-1}$ as input, and the current state of U' is saved in the variable $ustate^i$. The view of the real process in round i consists of $estate^i$, and the values y^i_p and y^i_u($EVIEW^i_{real}$=($estate^i$,y^i_p,y^i_u)). The overall view of E in the real process is just its view in the last round (i.e.,i for which $stop^i$ = TRUE.)
  -ideal process as follows:

$$\begin{aligned}EVIEW_{ideal} = \{&(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu} \leftarrow I(1^k, istate^{i-1});\\&(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i)\\&\leftarrow E(1^k, EVIEW^{i-1}_{ideal}, x^i_{hp}, x^i_{hu});\\&(astate^i, y^i_s, y^i_p, y^i_u);\\&\leftarrow Alg(astate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au});\\&(sstate^i, ustate^i, Y^i_p, Y^i_u), rep^i;\\&\leftarrow SIM_E^{U'(ustate^{i-1})}\\&(sstate^{i-1}, \ldots, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}, y^i_p, y^i_u);\\&(z^i_p, z^i_u) = rep^i(Y^i_p, Y^i_u)\\&+(1 - rep^i)(y^i_p, y^i_u)\\&(estate^i, y^i_p, y^i_u)\}\end{aligned}$$

  $EVIEW_{ideal} \sim EVIEW^i_{ideal}$ if $stop^i = TRUE$.
  The ideal process also proceeds in rounds. The view of E is $EVIEW^i_{ideal}$=($estate^i$, z^i_p, z^i_u). In the ideal process, we have a stateful simulator SIM_E who shielded from the secret input x^i_{hs}, but given the non-secret outputs that Alg produces when run all the inputs for round i, decides to either output the values (y^i_p; y^i_u) generated by Alg, or replace them with some other values (Y^i_p; Y^i_u). Note that this is captured by having the indicator variable rep^i be a bit that determines whether y^i_p will be replaced with Y^i_p. In doing so, it is allowed to query oracle U'; moreover, U' saves its state as in the real experiment.
- Pair Two $UVIEW_{real} \sim UVIEW_{ideal}$:
  -The view that the untrusted U' obtains by participating in the real process described in Pair One. $UVIEW_{real}$ = $unstate^i$ if $stop^i$=$TRUE$.
  -The ideal process:

$$\begin{aligned}EVIEW_{ideal} = \{&(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu} \leftarrow I(1^k, istate^{i-1});\\&(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i)\\&\leftarrow E(1^k, estate^{i-1}, x^i_{hp}, x^i_{hu}, y^{i-1}_p, y^{i-1}_u);\\&(astate^i, y^i_s, y^i_p, y^i_u);\\&\leftarrow Alg(astate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au});\\&(sstate^i, ustate^i);\\&\leftarrow SIM_{U'}^{U'(ustate^{i-1})}\\&(sstate^{i-1}, x^{j^i}_{hu}, x^i_{au}):(ustate^i)\}\end{aligned}$$

  $EVIEW_{ideal} = EVIEW^i_{ideal}$ if $stop^i = TRUE$.

In the ideal process, we have a stateful simulator $SIM_{U'}$, which equipped with only the unprotected inputs (x^i_{hu}, x^i_{au}), queries U'. As before, U' may maintain state.

Definition 4 (α-Efficient, Secure Outsourcing): A pair of algorithms (T, U) are an α-efficient implementation of an algorithm *Alg* if

1) T^U is an outsource-secure implementation of Alg.
2) $\forall$ inputs x, the running time of T is no more than α-multiplicative factor of the running time of *Alg*.

Definition 5 (β-Checkable, Secure Outsourcing): A pair of algorithms (T, U) are an β-checkable implementation of an algorithm *Alg* if

1) T^U is an outsource-secure implementation of *Alg*.
2) $\forall$ inputs x, if U' deviates from the advertised functionality during the execution of $T^{U'}$, T can detect it with probability at least β.

Definition 6 (α, β-Outsourcing-Security): A pair of algorithms (T, U) is an α, β-outsourcing-security if it both α-efficient and β-checkable.

III. SECURE OUTSOURCING ALGORITHM OF MODULAR EXPONENTIATIONS

A. OUTSOURCING ALGORITHM

In this section, we propose a secure outsourcing algorithm for modular exponentiation in one single untrusted server inspired by [27] and [28], which is more suitable to outsource the encryption operations(e.g., u^a or $u_1^a u_2^b$) used in most of existing ABE schemes [13], [16] to an untrusted computational server, hereafter this algorithm will be abbreviated as *Exp*. In algorithm *Exp*, a user(T) can outsource the modular exponentiation computation to an untrust server(U), in this process, U cannot learn any useful information about the inputs and outputs. Similar to [27] and [28], $Exp_1(a, u) \to u^a$ or $Exp_2(a, b; u_1, u_2) \to u_1^a u_2^b$, which denotes that the algorithm Exp_1 or Exp_2 inputs (a, u) or (a, b, u_1, u_2) outputs u^a or $u_1^a u_2^b$. We present Exp_1 and Exp_2 as follows, the Exp_1(Algorithm1) is called outsource-secure algorithm of multi-based modular exponentiation, while the Exp_2(Algorithm2) is called outsource-secure algorithm of single-based modular exponentiation

As in the [27], [28], in order to speed up the computations, a subroutine named Rand will be invoked. The function of Rand is to generate a random, independent pair of the form (i, g^i) for each invocation, where $i \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}$. There are two methods to implement this functionality. One is the table-look up method, it means that generating a table T which contains random, independent pairs (i, g^i) in advance. The other is to apply the well-knowing preprocessing techniques, e.g., EBPV generator [30], which runs in time $O(\log^2 n)$ for a n-bit exponent g^n and is secure against adaptive adversaries.

Remark 1: The Algorithm1 has 2/5 probability to check the correctness of the calculation executed by U. U cannot distinguish the two queries(ie, $U(\zeta, g^{\alpha_1})$, $U(\eta, g^{\alpha_2})$) from all of the five queries that T makes, if the U fails during any execution of Exp_1, it will be detected with probability 2/5. In practice, a user will query many times computing service from ESP, if the ESP cheats frequently, there is a high probability to detected the dishonest of server; Obviously, the Algorithm2 has 1/2 probability to check the correctness of the calculation executed by U.

Algorithm 1 $Exp_1(a, b; u_1, u_2) \to u_1^a u_2^b$.

```
1: Let $\mathbb{G}$ be a cyclic group with the prime order $p$ and $g$ be a generator of $\mathbb{G}$. Given two arbitrary bases $u_1, u_2 \in_R \mathbb{G}$ and two arbitrary powers $a, b \in_R \mathbb{Z}_p^*$. The output is $u_1^a u_2^b$ mod $p$. The proposed Algorithm1 is given as follows:
2: $T$ first runs Rand to get three blinding pairs $(\gamma_1, g^{\gamma_1})$, $(\gamma_2, g^{\gamma_2})$ and $(\beta, g^{\beta})$.
3: The main trick is a more efficient solution to logically split $u_1$, $u_2$, $a$ and $b$ into random looking pieces that can be computed by $U$. Then the divisions are:
```

$$
\begin{aligned}
u_1^a u_2^b &= (g^{\gamma_1}\omega_1)^a (g^{\gamma_2}\omega_2)^b \\
&= g^{\gamma_1 a + \gamma_2 b}\omega_1^a \omega_2^b \\
&= g^{\gamma_1 a + \gamma_2 b}\omega_1^{c_1+dx}\omega_2^{c_2-dx} \\
&= g^{\gamma_1 a + \gamma_2 b}\omega_1^{c_1}\omega_2^{c_2}\omega_1^{dx}\omega_2^{-dx} \\
&= g^{\gamma_1 a + \gamma_2 b}\omega_1^{c_1}\omega_2^{c_2}(\omega_1\omega_2^{-1})^{dx} \\
&= g^{\beta} g^{\gamma_1 a + \gamma_2 b - \beta}\omega_1^{c_1}\omega_2^{c_2}(\omega_1\omega_2^{-1})^{dx} \mathit{mod}\ p
\end{aligned}
$$

```
   where $\omega_1 = u_1/g^{\gamma_1}$ mod $p$, $\omega_2 = u_2/g^{\gamma_2}$ mod $p$ $c_1 = a - dx$ mod $p$, $c_2 = b + dx$ mod $p$ where $d \in_R \mathbb{Z}_p^*$, $x$ is a random value that $x \geq 2^{\lambda}$, $\lambda$ is a security parameter, e.g., $\lambda$=64. Then $T$ calculates $(\omega_1\omega_2^{-1})^x$.
4: Next, $T$ runs Rand to obtain three pairs $(\alpha_1, g^{\alpha_1})$,$(\alpha_2, g^{\alpha_2})$,$(\alpha_3, g^{\alpha_3})$.
5: $T$ queries server $U$ in random order as:
```

$$
\begin{aligned}
&U(c_1, \omega_1) \to \omega_1^{c_1}; \\
&U(c_2, \omega_2) \to \omega_2^{c_2}; \\
&U(d, (\omega_1\omega_2^{-1})^x) \to (\omega_1\omega_2^{-1})^{dx}; \\
&U(\zeta, g^{\alpha_1}) \to g^{\alpha_1\zeta}; \\
&U(\eta, g^{\alpha_2}) \to g^{\alpha_2\eta};
\end{aligned}
$$

```
   where
```

$$
\begin{aligned}
\zeta &= (\gamma_1 a + \gamma_2 b - \beta)/\alpha_1 \ \mathit{mod}\ p \\
\eta &= (\alpha_3 - \zeta)/\alpha_2 \ \mathit{mod}\ p.
\end{aligned}
$$

```
6: In the end, $T$ checks the correct outputs of $U$, i.e.
```

$$U(\zeta, g^{\alpha_1}) \cdot U(\eta, g^{\alpha_2}) \overset{?}{=} g^{\alpha_3},$$

```
   if it does not hold, $T$ outputs error, otherwise, $T$ can compute:
```

$$
\begin{aligned}
u_1^a u_2^b &= g^{\beta} g^{\alpha_1\zeta}\omega_1^{c_1}\omega_2^{c_2}(\omega_1\omega_2^{-1})^{dx} \mathit{mod}\ p \\
&= g^{\beta} g^{\gamma_1 a + \gamma_2 b - \beta}\omega_1^{c_1}\omega_2^{c_2}(\omega_1\omega_2^{-1})^{dx} \mathit{mod}\ p
\end{aligned}
$$

```
Algorithm 2 Exp_2(a, u)→u^a
1: Let $G$ be a cyclic group with the prime order $p$ and $g$ be a generator. Given an arbitrary base $u \in_R \mathbb{G}$ and an arbitrary power $a \in_R \mathbb{Z}_p^*$. The output is $u^a \bmod p$. The proposed Algorithm2 is given as follows:
2: $T$ first runs Rand twice to get two blinding pairs $(\gamma, g^\gamma)$ and $(\beta, g^\beta)$.
3: The main trick is a more efficient solution to logically split $u$, $a$ into random looking pieces that can be computed by $U$. Then the divisions are:
       $u^a = (u)^{c+dx} = u^c u^{dx} = (g^\gamma \omega)^c u^{dx} = g^{\gamma c}\omega^c(u^x)^d = g^\beta g^{\gamma c-\beta}\omega^c(u^x)^d \bmod p$
   where $\omega = u/g^\gamma$, $c = a - dx \bmod p$, and $d \in_R \mathbb{Z}_p^*$, $x$ is a random value that $x \geq 2^\lambda$, $\lambda$ is a security parameter, e.g., $\lambda=64$. Then $T$ calculates $u^x$.
4: Next, $T$ runs Rand to obtain three pairs $((\alpha_1, g^{\alpha_1}), (\alpha_2, g^{\alpha_2}), (\alpha_3, g^{\alpha_3}))$.
5: T queries server $U$ in random order as:
       $U(c, \omega) \rightarrow \omega^c$;
       $U(d, u^x) \rightarrow u^{dx}$;
       $U(\zeta, g^{\alpha_1}) \rightarrow g^{\alpha_1\zeta}$;
       $U(\eta, g^{\alpha_2}) \rightarrow g^{\alpha_2\eta}$;
   where $\zeta=(\gamma c - \beta)/\alpha_1 \bmod p$, $\eta=(\alpha_3 - \zeta)/\alpha_2 \bmod p$.
6: In the end, $T$ checks the correct outputs of $U$, i.e.
       $U(\zeta, g^{\alpha_1}) \cdot U(\eta, g^{\alpha_2}) \overset{?}{=} g^{\alpha_3}$
   if it does not hold, $T$ outputs error, otherwise, $T$ can compute:
       $u^a = g^\beta g^{\alpha_1\zeta}\omega^c\omega^{dx} \bmod p = g^\beta g^{\gamma a-\beta}\omega^c\omega^{dx} \bmod p$
```

Remark 2: The Algorithm1 and Algorithm2 can be applied to scalar multiplication on elliptic curves, i.e. compute ag for $a \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}$, where $\mathbb{G}$ is a cyclic addition group on elliptic curves defined over a finite field $GF(p)$. All of the elements like g^a in the above Algorithms is written as ag, the details are omitted here.

B. SECURITY ANALYSIS

Theorem 1: In single untrusted program model, the Algorithm1 is a outsource-secure implementation of Exp, where the inputs $(a, b; u_1, u_2)$ may be honest, secret; or honest, protected; or adversarial, protected, and all the bases are distinct.

The proof of the Algorithm1 is similar to [27], [28]. The correctness is obviously if U performs honestly, here we mainly focus on the security. Let $\mathcal{A} = (E, U')$ be a PPT adversary that interacts with the algorithm T in the oursource-security model.

Firstly, we prove that $EVIEW_{real} \sim EVIEW_{ideal}$.(it means the adversary E learns nothing during the execution of T^U)

If the input $(a, b; u_1, u_2)$ is honest, protected; or adversarial, protected, the simulator SIM_E behaves the same way as in the real execution. If the input$(a, b; u_1, u_2)$ is honest, secret, then the SIM_E acts follows: on receiving the input in the ith round, SIM_E ignores it and random chooses five queries of the form(τ_i, g^{μ_i}) to U' in random orders, for $i = 1$ to 5 with the condition that $\tau_4\mu_4 + \mu_5\tau_5 = \tau$. Then SIM_E test the outputs $g^{\tau_4\mu_4}g^{\tau_5\mu_5} \overset{?}{=} g^\tau$, if an error is detected, SIM_E stores all the states and output $Y_p^i = \text{“}error''$, $Y_u^i = \emptyset$, $rep^i = 1$. If no error is detected, SIM_E outputs $Y_p^i = \emptyset$, $Y_u^i = \emptyset$, $rep^i = 0$. otherwise, SIM_E randomly chooses a group value $r \in G$ and outputs $Y_p^i = r$, $Y_u^i = \emptyset$, $rep^i = 1$, next, SIM_E saves the corresponding states. The input distributes to U' in the real and ideal executions are computationally indistinguishable. In the real execution, the five inputs are randomly chosen by T and thus computationally indistinguishable from random, in the ideal execution, the inputs are chosen uniformly at random. If U' behaves in an honest in the ith round, then $EVIEW_{real} \sim EVIEW_{ideal}$, this is because $T^{U'}$ executes Algorithm1 in the real experiment and in the ideal experiment simulator SIM_E doesn't change the output of Algorithm. If U' give dishonest results, it will be detected by T and SIM_E with probability $\frac{2}{5}$, resulting in an output of $error°$; otherwise, the U' will indeed be successful in manipulating the output of Algorithm1 with probability $\frac{3}{5}$. In the real experiment, the five outputs are multiplied together with a random value which is random to E. In the ideal experiment, SIM_E simulates with a random r. then $EVIEW_{real} \sim EVIEW_{ideal}$ even if U' behaves maliciously in the ith round. By the hybrid argument, we can easily conclude that $EVIEW_{real} \sim EVIEW_{ideal}$.

Next, we prove the Pair Two $UVIEW_{real} \sim UVIEW_{ideal}$.

The simulator $SIM_{U'}$ acts as follows: it ignores the inputs in the ith round and instead chooses random five queries of the form (τ_i, g^{μ_i}) to U'. Then $SIM_{U'}$ saves its own states and the sates of U'. We should note that E can distinguish between real and ideal experiments, but E can not communicate this information to U'(note that the output in the ideal is never spoiled). In the real experiment in ith round, T always re-randomizes its inputs to U'. In the ideal experiment, $SIM_{U'}$ always chooses queries for U' independently and randomly. Thus, we prove $UVIEW_{real} \sim UVIEW_{ideal}$ in each round i. By the hybrid argument, we can easily conclude that $UVIEW_{real} \sim UVIEW_{ideal}$.

Theorem 2: In single untrusted program model, the Algorithm $1(T, U)$ is a $(O(\frac{logx+14}{n}, \frac{2}{5}))$ outsource-secure implementation of Exp_1, where n denotes the number of bits of the exponent a, x is the random value that $x \geq 2^\lambda$, where λ is a security parameter, e.g., $\lambda = 64$.

TABLE 1. Comparison of computing $u_1^a u_2^b$.

	[27]	[28]	Algorithm1
M	5O(Rand)+10	17+1.5log$x$	1.5log$x$+6O(Rand)+9
Inversions	3	6	5
Invoke(U)	8	6	5
Checkability	1/2	1/3	2/5
Security Model	Two servers	Single server	Single server

TABLE 2. Comparison of computing u^a.

	[27]	[28]	Algorithm2
M	5O(Rand)+7	12+1.5log$x$	1.5log$x$+5O(Rand)+6
Inversions	3	4	3
Invoke(U)	6	4	4
Checkability	1/2	1/2	1/2
Security Model	Two servers	Single server	Single server

x is a random value that $x \geq 2^{\lambda}$, e.g., λ=64.

Our proof is similar to [27], [28], the Algorithm1 T makes 6 Rand plus (1.5log$x$+14) modular multiplications and 4 modular inverse in order to compute u^a mod p(other operations such modular additions are omitted, note that T computes the $(\omega_1\omega_2^{-1})^x$ as described in Algorithm1, it takes roughly 1.5log$x$ modular multiplications by the well-known square-and-multiply method to calculate it). Besides, it takes O(1) or O(log$^2 n$) modular multiplications using the table-look up or EBPV method [30], where n is the bit of the a. On the other hand, it takes roughly 1.5n modular multiplications to calculate u^a by the well-known square-and-multiply method, then the algorithm (T, U) is an $O(\frac{logx+14}{n})$−efficient implementation of Algorithm1.

Because U can not distinguish the two test queries from all five queries made by T. If U fails during any execution of Exp_1, it will be detected with a probability $\frac{2}{5}$.

Theorem 3: In single untrusted program model, the Algorithm2 is $(O(\frac{logx+14}{n}), \frac{1}{2})$outsource-secure.

We can easily prove the theorem by the same method as the Theorem1 and Theorem2.

C. COMPARISON

In this section, we compare our proposed Algorithm1 and Algorithm2 with the corresponding algorithms in [27], [28]. TABLE 1 presents the comparison of the computational efficiency on the outsourcer T side, the checkability, security model and other properties between the Algorithm1 and the corresponding algorithms that in [27], [28]. TABLE 2 presents the comparison of corresponding properties between the Algorithm2 and the corresponding algorithms in [27], [28]. Note that M denotes a modular multiplication, inversion denotes a modular inverse, and Invoke denotes an invocation of the subroutine Rand. We also omit other operations such as modular additions in those algorithms. Compared with [27], our algorithm is implemented in single untrusted server model which is the same as in [28], while [27] is implemented in two untrusted server model. Compared with [28], the interactions between the user and the computation server are less than that in [28]. Then our algorithm is more suitable to ABE system in the term of computation and communication cost on the user side. Note that the modular exponentiation is the most basic operation in discrete-logarithm based cryptographic protocols, and millions of such computations may be outsourced to the server every day. Thus, our proposed algorithm can save huge of computational resources for both the outsourcer T and the servers U.

IV. MODEL OF ABE-VOED AND THE SECURITY DEFINITION

In this section, we present the model of ABE-VOED based on the Algorithm1 and Algorithm2, and the security definition of ABE-VOED.

A. ABE-VOED SCHEME MODEL

Let $\mathbf{A}$ denotes an access policy and S represents a set of attributes, (I_{key}, I_{enc}) denotes that the inputs of the keygen algorithm and the encryption algorithm, respectively, if the scheme is CP-ABE scheme $(I_{key}, I_{enc}) := (S, \mathbf{A})$, else in a KP-ABE scheme$(I_{key}, I_{enc}) := (\mathbf{A}, S)$.

Our scheme is based on the Waters's scheme [3], our scheme contains a additional ESP when compared with existing ABE [3], [10]–[12], we can apply our technique to the most of the existing ABE schemes [3], [10]–[12]. Our framework as shown in FIGURE 1.

Definition 4 (Syntax of the ABE-VOED): The model includes five algorithms as follows:

- **Setup**$(\mathbf{U}) \rightarrow (PK, MSK)$ The setup algorithm takes as input a attributes set $\mathbf{U}$ involved in the system, It outputs public keys PK and master secret key MSK.
- **Keygen**$(PK, MSK, I_{key}) \rightarrow (SK)$. The key generation algorithm takes as input the PK, the MSK and user's attributes set $I_{key} \in \mathbf{U}$, then it outputs a private SK for user.
- **Encryption**$(PK, M, I_{enc}, \mathcal{M}) \rightarrow CT$. This encryption algorithm ran by the DO under the cooperation of ESP. it takes as input PK, a data $\mathcal{M}$, the intermediate ciphertext $CT_{intermediate}$ are generated with the Algorithm1 and Algorithm2 under the cooperation of ESP(outsourcing part of encryption task to ESP), and the I_{enc} is an access policy $\mathbf{A}$, then outputs a ciphertext CT.
- **GenTK** This algorithm executed by DO, it takes input the private key SK, it outputs two transformation key TK_1, TK_2 and corresponding retrieving key RK_1, RK_2.
- **Transform**(CT, TK) This algorithm ran by DSP$_1$ and DSP$_2$, respectively. DSP$_i$ takes as input a CT and a transformation key TK_i, i=1, 2, and outputs a partially decrypted ciphertext $CT_{[i]}$,i=1, 2.
- **Check Correctness** This algorithm takes input the $CT_{[i]}$ and RK_i, then it check the correctness of the computing by DSP$_i$, for i=1, 2. If the check result is yes, outputs a ciphertext CT^*; Otherwise, outputs $\perp$.
- **Decryption** The decryption algorithm takes as input a private SK, a ciphertext CT, a partially decrypted cipher-

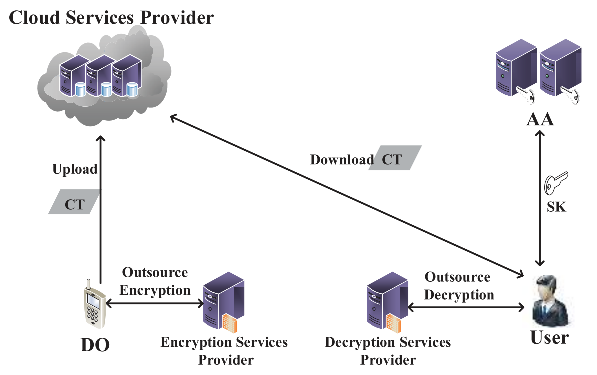

FIGURE 1. The system model of our scheme.

text CT^*, then outputs the data M, if the $f(I_{key}, I_{enc}) = 1$, else, it outputs $\perp$.

B. SCHEME SECURITY DEFINITION

In the system, we considered two types of adversaries as follows:

- Type-I adversary: A group of curious users who are able to access private keys for theirself, and aim to decrypt ciphertext intended for users not in the group.
- Type-II adversary: The curious users collude with encryption service provider who can potential obtain access private keys for all the corrupted users, the encryption parameters information and partial ciphertext, etc. Thus, we follow the replayable chosen-ciphertext attack(RCCA) in [31] and define RCCA security game for our scheme.

RCCA Security Game for Type-I

- **Setup**. The challenger runs Setup(**U**) and gives the public key PK to the adversary.
- **Phase 1**.The challenger initializes an integer j=0, initializes an empty table T^* and an empty set D to provide the oracles for adversary as follows:
  1) $\mathcal{O}_{SK}(I_{key})$ The challenger sets j=j+1, runs KeyGen and GenTK algorithm for I_{key} to obtain SK_{I_key} and $TK_{I_{key}}$, and stores the entry $(j, I_{key}, SK_{I_{key}}, TK_{I_{key}})$ in table T^*, and returns $SK_{I_{key}}$.
  2) $\mathcal{O}_{TK}(I_{key})$ The challenger checks whether the entry (I_{key},$SK_{I_{key}}$,$TK_{I_{key}}$) exists in table T^*. If so, then set D=$D\cup\{I_{key}\}$ and return $TK_{I_{key}}$; Otherwise runs the Gentk algorithm to for I_{key} to obtain $TK_{I_{key}}$, and returns $TK_{I_{key}}$.
  3) $\mathcal{O}_{Decryption}(j, CT)$ Upon receiving the ciphertext CT encrypted under I_{enc}. Challenger checks whether the $(j, I_{key}, SK_{I_{key}})$ in the table T^* and $f(I_{key}, I_{enc}) = 1$, if so, it runs the *Decrypt* algorithm to decrypt the CT and return the message $\mathcal{M}$; otherwise return $\perp$.
- **Challenge**. Adversary submits two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ as well as I'_{enc} with the constraint that $f(I'_{enc}, I_{key}) = 0$ for all $I_{key} \in D$. Challenger random chooses b$\in$ {0,1} and encrypts M_b under I'_{enc}. Finally, return the ciphertext CY'.
- **Phase 2** Phase 2 is as like phase 1 but with the restrictions as follows:
  1) Adversary can not query $\mathcal{O}_{SK}(I_{key})$ with $f(I'_{enc}, I_{key}) = 1$.
  2) Adversary can not query $\mathcal{O}_{Decryption}(CT)$ which outputs either $\mathcal{M}_0$ or $\mathcal{M}_1$.
- **Guess**. Adversary outputs a guess b' of b.

RCCA Security Game for Type-II

- **Setup**. The challenger runs Setup(λ, **U**) and gives the publics key PK to the adversary.
- **Phase 1**.The challenger initializes an integer $j = 0$, initializes an empty table T^* and an empty set D to provide the oracles for adversary as follows:
  1) $\mathcal{O}_{TK}(I_{key})$ It is identical to $\mathcal{O}_{TK}(I_{key})$ for Type-I adversary.
  2) $\mathcal{O}_{Encryption}(I_{enc})$ Upon receiving I_{enc}, j, $\mathcal{M}$, the challenger runs the Algorithm *Encrypt* to get CT and returns it to adversary.
  3) $\mathcal{O}_{Decryption}(j, CT)$ Upon receiving the ciphertext CT encrypted under I_{enc}. Challenger checks whether the $(j, I_{key}, SK_{I_{key}})$ in the table T^* and $f(I_{key}, I_{enc}) = 1$, if so, it runs the *Decrypt* to decrypt the CT and return the message $\mathcal{M}$; otherwise return $\perp$.
- **Challenge**. Adversary submits two messages $\mathcal{M}_0$ and $\mathcal{M}_1$ as well as I'_{enc}. Challenger random chooses b$\in$ {0,1} and encrypts $\mathcal{M}_b$ under I'_{enc}. Finally, return the ciphertext CT'.
- **Phase 2** Phase 2 is as like phase 1 but with the restriction that adversary can not query $O_{Decryption_{out}}(j, CT')$ which output either $\mathcal{M}_0$ or $\mathcal{M}_1$.
- **Guess**. Adversary outputs a guess b' of b.

Defination 7 (RCCA): A CP-ABE or KP-ABE scheme with outsourced encryption is secure against replayable chosen-ciphertext attack if no polynomial time adversaries have a non-negligible advantage in the RCCA security games defined above.

Note that, we can definite CPA-secure model if we remove the decryption queries in the Phase 1 and Phased 2 in the above two games; It is a selective secure model if we add an Init stage before setup where the adversary commits to the challenge access structure I'_{enc}. We will prove our scheme in the selective security model.

V. OUR CONSTRUCTION

A. CONSTRUCTION

The ABE-VOED is based on the scheme [3](note that we can build our scheme based on most of the existing CP-ABE or KP-ABE schemes), the Algorithm1 and Algorithm2. The Setup, Keygen and Decryption algorithms are same with [3], the main difference between [3] and our scheme are the encryption phase and the decryption phase. In the FIGURE 2

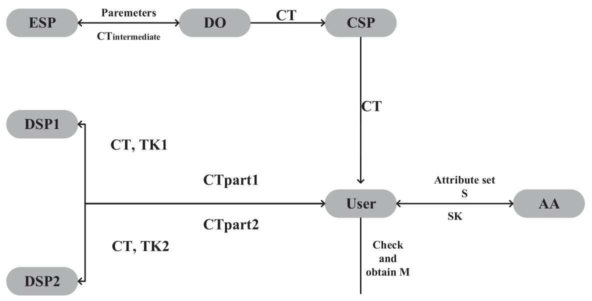

FIGURE 2. **The working process of our construction.**

we give the working process of our construction. The details of ABE-VOED are described as follows:

- **Setup(U)** The setup algorithm takes as input a universe **U** of attributes that is involved in the system, it then chooses a group $\mathbb{G}$ of prime order p, a generater g of $\mathbb{G}$ and a hash function H: $\{0,1\}^* \to \mathbb{Z}_p^*$ which will map any attributes described as binary string to a random group element in $\mathbb{Z}_p^*$. In addition, it also chooses random exponents α, $a\in\mathbb{Z}_p^*$. then it outputs the PK, MSK as follows:

$$PK = \{g, e(g,g)^\alpha, g^a, H\} \tag{1}$$

The authority sets the MSK as g^α.

- **Keygen**(PK, MSK, I_{key}) The key generation algorithm takes as input the public parameters PK, the MSK and the $I_{key}\in$**U**, where I_{key} is an attribute set S. Firstly, the algorithm chooses a random $t\in\mathbb{Z}_p^*$. It generates the private key SK as:

$$K = g^\alpha g^{at},\ L = g^t,\ \forall x \in S,\ K_x = g^{H(att(x))t}. \tag{2}$$

- **Encryption**($PK, \mathcal{M}, (M,\rho)$) This algorithm are executed by DO under the cooperation of ESP, recall that DO is the T, ESP is the U as described in Algorithm1 and Algorithm2, when a DO(T) wants to encrypt message $\mathcal{M}$ with an $LSSS$ access structure(M,ρ), the function ρ associates rows of M to attributes, M is an $\ell\times n$ matrix.

  DO chooses a random $s\in\mathbb{Z}_p^*$, and chooses a random vector $v=(s, y_2, y_3, , , y_n)\in\mathbb{Z}_p^n$, these values will be used to share the encryption exponent s. It calculates $\{\lambda_i = v\cdot M_i\}_{i=1}^{i=\ell}$, where M_i is the vector corresponding to the ith row of M. In addition, the DO chooses random $r_1,\ldots r_\ell\in\mathbb{Z}_p$, then the ciphertext CT =

$$\{C = \mathcal{M}e(g,g)^{\alpha s}, C' = g^s, \\ (C_i = g^{a\lambda_i}g^{-r_iH(att(i))}, D_i = g^{r_i})_{i=1}^{i=\ell}\}.$$

  The first two items (C, C') of CT are computed by DO(T), the rest of the items $\{C_i, D_i\}_{i=1}^{i=\ell}$(denoted as $CT_{intermediate}$) are computed with Algorithm1 and Algorithm2 under the cooperation of ESP(U), respectively. The details as follows:

  We note that for each C_i:
  $Exp_1(\lambda_i, H(att(i)\cdot -r_i; g^a, g) \to (g^a)^{\lambda_i} g^{H(att(i)\cdot -r_i} = C_i$
  as described in Algorithm1, DO(T) runs Rand to get (γ_1, g^{γ_1}),(γ_2, g^{γ_2}),(β, g^β),(α_1, g^{α_1}), (α_2, g^{α_2}), (α_3, g^{α_3}), then queries ESP(U) in random order as follows:

$$\begin{aligned} U(c_{i1}, \omega_1) &\to \omega_1^{c_{i1}};\\ U(c_{i2}, \omega_2) &\to \omega_2^{c_{i2}};\\ U(d_i, (\omega_1\omega_2^{-1})^x) &\to (\omega_1\omega_2^{-1})^{d_ix};\\ U(\zeta_i, g^{\alpha_1}) &\to (g^{\alpha_1})^{\zeta_i};\\ U(\eta_i, g^{\alpha_2}) &\to (g^{\alpha_2})^{\eta_i};\end{aligned}$$

  as described in Algorithm1.

$$\begin{aligned} \omega_1 &= g^a/g^{\gamma_1}, \quad \omega_2 = g/g^{\gamma_2}\\ c_{i1} &= \lambda_i - d_ix, \quad c_{i2} = -r_iH(att(i)) + d_ix\\ \zeta_i &= (\gamma_1\lambda_i - \gamma_2 r_i H(att(i)) - \beta)/\alpha_1\\ \eta_i &= (\alpha_3 - \zeta)/\alpha_2.\end{aligned}$$

  where d_i, r_i are random chosen from $\mathbb{Z}_p^*$, x is random value such that $x\geq 2^\lambda$, ie. λ=64, and calculates $(\omega_1\omega_2^{-1})^x$(note that for each computation of C_i, the $(\omega_1\omega_2^{-1})^x$ is invariable, DO(T) computes it only once in the process of computing $\{C_i\}_{i=1}^{i=\ell}$).
  Then ESP(U) sends this five outputs T_{out} to DO(T). where

$$T_{out} = (U(c_{i1}, \omega_1), U(c_{i2}, \omega_2), U(d_i, (\omega_1\omega_2^{-1})^x), \\ U(\zeta_i, g^{\alpha_1}), U(\eta_i, g^{\alpha_2})) \tag{3}$$

- **Checkability** Finally, the DO(T) can check the the correctness of the outputs from ESP(U), if

$$U(\zeta_i, g^{\alpha_1}) \cdot U(\eta_i, g^{\alpha_2}) \overset{?}{=} g^{\alpha_3}.$$

  then it indicates that ESP performs honestly, and DO(T) can compute

$$\begin{aligned} C_i &= g^\beta (g^{\alpha_1})^{\zeta_i} \omega_1^{c_{i1}} \omega_2^{c_{i2}} (\omega_1\omega_2^{-1})^{d_ix}\\ &= g^\beta g^{\gamma_1\lambda_i - \gamma_2 r_i H(att(i))} \omega^{c_{i1}+d_ix} \omega_2^{c_{i2}-d_ix}\\ &= g^\beta g^{\gamma_1\lambda_i - \gamma_2 r_i H(att(i))} (g^a g^{-\gamma_1})^{\lambda_i} (gg^{-\gamma_2})^{-r_iH(att(i))}\\ &= g^{a\lambda_i} g^{-r_iH(att(i))}\end{aligned}$$

  Otherwise, it indicates that ESP(U) has produced wrong responses, and thus DO(T) output "error".

  We can compute the $D_i = g^{r_i}$ with the Algorithm2: $Exp_2(r_i; g) \to g^{r_i}$, the details are omitted here. As described in Algorithm2, DO(T) will also compute once modular exponentiation(g^x) and some multiplication operations. At last, DO can get:

$$CT_{intermediate} = \{g^{a\lambda_i} g^{-r_iH(att(i))}, g^{r_i}\}_{i=1}^{i=\ell}.$$

  Finally, it outputs the whole ciphertext CT =

$$\{C = \mathcal{M}e(g,g)^{\alpha s}, C' = g^s, \\ (C_i = g^{a\lambda_i}g^{-r_iH(att(i))}, D_i = g^{r_i})_{i=1}^{i=\ell}\}$$

Note that: In this encryption phase, DO uses the Algorithm1 and Algorithm2 to outsource the computation task of $CT_{intermediate}$ to ESP, and only computes two modular exponentiations and $O(\ell)$ multiplications, and the cost of checking process is once multiplication operation. In total, DO only compute 4 modular exponentiations$(C, C', (\omega_1\omega_2^{-1})^x, g^x)$ and $O(\ell)$ multiplications in encryption phase.

- **GenTK**(SK) This algorithm takes input the private key $SK{=}(K, L, K_x)$, $x{\in}S$, it randomly chooses two values $z_1, z_2 \in \mathbb{Z}_p^*$, and chooses a K_j in K_x, note that the attribute $j{\in}S$ must be a necessary one to fully decrypt the CT. Then it generates two transformation key TK_1 and TK_2, and the corresponding retrieving key are RK_1 and RK_2, respectively. At last, the user sent the TK_1 and TK_2 to DSP$_1$ and DSP$_2$, respectively. Where

$$\begin{aligned} TK_1 &= (L, K_j^{z_1}, \{K_x\}_{x\in S, x\neq j}) \\ TK_2 &= (L, K_j^{z_1z_2}, \{K_x\}_{x\in S, x\neq j}) \\ RK_1 &= z_1 \\ RK_2 &= z_1z_2. \end{aligned}$$

- **Transform**$_{out}(CT, TK_1, TK_2, PK)$ DSP$_i$, for i=1,2 runs this algorithm with the input PK, CT and the corresponding key TK_i, for $i = 1, 2$, respectively. Then outputs a partially decrypted ciphertext $CT_{[i]}$, for $i{=}1, 2$. If the attributes set $S{=}I_{key}$ satisfies the access structure (M, ρ), let $I \subset \{1, 2...\ell\}$ be defined as $I{=}\{i : \rho(i) \in S\}$, let $\{\omega_i{\in}\mathbb{Z}_p\}_{i\in I}$ be a set of constants such that $\{\lambda_i\}_{i=1}^{i=\ell}$ are valid shares of any secret s according to M, then $\sum_{i\in I}\omega_i\lambda_i = s$. Upon receiving the CT and the TK_1,DSP$_1$ computes:

$$\begin{aligned} &CT_{[1]} \\ &= \frac{e(C', K)}{(e(C_j, L)e(D_i, K_j^{z_1}))^{\omega_j} \prod_{i\in I, i\neq j}(e(C_i, L)e(D_i, K_{\rho(i)}))^{\omega_i}} \\ &= \frac{e(g, g)^{\alpha s}e(g, g)^{ast}}{e(g, g^{H(j)})^{r_jt\omega_jz_1}\prod_{i\in I} e(g, g)^{at\lambda_i\omega_i}e(g, g^{H(j)})^{-r_jt\omega_j}} \\ &= \frac{e(g, g)^{\alpha s}e(g, g)^{ast}}{e(g, g^{H(j)})^{r_jt\omega_jz_1}e(g, g)^{\sum_{i\in I} at\lambda_i\omega_i}e(g, g^{H(j)})^{-r_jt\omega_j}} \\ &= \frac{e(g, g)^{\alpha s}}{e(g, g^{H(j)})^{r_jt\omega_jz_1}e(g, g^{H(j)})^{-r_jt\omega_j}} \\ &= \frac{e(g, g)^{\alpha s}}{e(g, g)^{H(j)r_jt\omega_jz_1}e(g, g)^{-r_jt\omega_jH(j)}} \end{aligned}$$

in the same way, DSP$_2$ computes:

$$\begin{aligned} &CT_{[2]} \\ &= \frac{e(C', K)}{e(C_j, L)e(D_i, K_j^{z_1z_2}))^{\omega_j} \prod_{i\in I, i\neq j}(e(C_i, L)e(D_i, K_{\rho(i)}))^{\omega_i}} \\ &= \frac{e(g, g)^{\alpha s}e(g, g)^{ast}}{e(g, g^{H(j)})^{-r_jt\omega_j}e(g, g)^{ast}e(g, g^{H(j)})^{r_jt\omega_jz_1z_2}} \\ &= \frac{e(g, g)^{\alpha s}}{e(g, g^{H(j)})^{-r_jt\omega_j}e(g, g^{H(j)})^{r_jt\omega_jz_1z_2}} \\ &= \frac{e(g, g)^{\alpha s}}{e(g, g)^{H(j)r_jt\omega_jz_1z_2}e(g, g)^{-r_jt\omega_jH(j)}} \end{aligned}$$

Then the user get the transformed ciphertext as $CT'{=}(CT_{[1]}, CT_{[2]})$

Check Correctness$(RK_{[1]}, RK_{[2]}, CT_{[1]}, CT_{[2]}, PK)$ This algorithm takes input the $CT_{[i]}$ and RK_i, then it check the correctness of the computing by DSP$_i$, for $i{=}1, 2$. The details as follows:

First, user inputs the pre-selected attribute j corresponding the key K_j, D_j and ω_j and the $RK_1{=}z_1$ and $RK_2{=}z_1z_2$, computing as follows:

$$\begin{aligned} A_{[1]} &= e(D_j, K_j^{z_1}) = e(g^{r_j}, g^{H(j)z_1})^{H(j)r_jtz_1} \\ B_{[1]} &= A_{[1]}^{\omega_j} = e(g, g)^{H(j)r_jt\omega_jz_1} \\ A_{[2]} &= A_{[1]}^{RK_2/RK_1} = e(g, g)^{H(j)r_jtz_1z_2} \\ B_{[2]} &= A_{[2]}^{\omega_j} = e(g, g)^{H(j)r_jt\omega_jz_1z_2} \\ C_0 &= A_{[2]}^{-\frac{\omega_j}{z_1z_2}} = e(g, g)^{-r_jt\omega_jH(j)} \\ &or \\ C_0 &= A_{[1]}^{-\frac{\omega_j}{z_1}} = e(g, g)^{-r_jt\omega_jH(j)} \end{aligned}$$

Then, input the $CT_{[1]}, CT_{[2]}$ and corresponding RK_1 and RK_2, check the equation:

$$CT_{[1]}B_{[1]}C_0 \overset{?}{=} CT_{[2]}B_{[2]}C_0$$

If the equation is not equal, it outputs $\perp$; otherwise, it is to say the outputs done by DSP$_1$ and DSP$_2$ are correct, then it outputs $CT^*{=}CT_{[1]}B_{[1]}C_0$ (or $CT_{[2]}B_{[2]}C_0){=}e(g, g)^{\alpha s}$.

In fact, the validation process is as follows:

$$\begin{aligned} CT_{[1]}B_{[1]}C_0 &= \frac{e(g, g)^{\alpha s}}{e(g, h_j)^{-r_jt\omega_j}e(g, h_j)^{r_jt\omega_jz_1}} \\ &\quad \cdot e(g, h_j)^{r_jt\omega_jz_1}e(g, h_j)^{-r_jt\omega_j} \\ &= e(g, g)^{\alpha s} \\ CT_{[2]}B_{[2]}C_0 &= \frac{e(g, g)^{\alpha s}}{e(g, h_j)^{-r_jt\omega_j}e(g, h_j)^{r_jt\omega_jz_1z_2}} \\ &\quad \cdot e(g, h_j)^{r_jt\omega_jz_1z_2}e(g, h_j)^{-r_jt\omega_j} \\ &= e(g, g)^{\alpha s} \end{aligned}$$

- **Dncryption**(PK, CT, CT^*)
If the Check Correctness algorithm outputs CT^*, the decryption algorithm takes as input the CT^* and the CT then, it computes $\mathcal{M} = C/CT^*$, it outputs the message $\mathcal{M}$; otherwise, it outputs $\perp$. Finally, completely decrypt the ciphertext as follows:

$$\begin{aligned} \frac{C}{C_{part}} &= \frac{\mathcal{M}e(g, g)^{\alpha s}}{e(g, g)^{\alpha s}} \\ &= \mathcal{M} \end{aligned}$$

B. SECURITY ANALYSIS

Our construction is based on the scheme [3], the difference between [3] and our scheme is that data encryption phase, the proof method of our scheme is same with [3].

Theorem 4: The construction of ABE-VOED without outsourced decryption scheme is indistinguishable secure against chosen-plaintext attack in the security definition model modified in the section III.B under DBDH assumption.

We will prove that our scheme is secure against both Type-I and Type-II adversaries. Note that the main difference between Type-I and Type-II is the oracle provided, specifically, Type-I adversary is provided with $\mathcal{O}_{SK}(\cdot)$, while Type-II is provided with $\mathcal{O}_{Enc}(\cdot)$ and $\mathcal{O}_{SK}(\cdot)$.

Type-I adversary, which is same as the proof of [abe m].

We will proof that if it exist an adversary $\mathcal{A}$ with non-negligible advantage ϵ in the security definition modified in the section III, we can build a simulator $\mathcal{B}$ that plays the decisional q-parallel BDHE problem.

- *Init*. The simulator is given a q-parallel BDHE challenge y, T. Then the adversary provide a challenge access structure (M^*, ρ^*), where M^* has n^* columns.
- *Setup*. The simulator chooses random $\alpha^* \in \mathbb{Z}_p$ and set $\alpha = \alpha^* + a^{q+1}$ by letting $e(g, g)^{\alpha} = e(g^a, g^{a^q})e(g, g)^{a^*}$. We describe the group elements $h_1, \ldots, h_U$ as follows:

$$h_x = g^{z_x} \prod_{i \in X} g^{aM^*_{i,1}/b_i} g^{a^2 M^*_{i,2}/b_i} \ldots g^{a^{n^*} M^*_{i,n^*}/b_i}$$

where $z_x \xleftarrow{R} \mathbb{Z}_p$ for $1 \le x \le U$, X is the set of indices i, such that $\rho^*(i) = x$. Note that, the $h_x \in G$ are distributed randomly elements duo to the random value g^{z_x} and if $X = \emptyset$, then $h_x = g^{z_x}$.
- *Phase*1 In this phase the adversary give a private key query for a set S which does not satisfy M^*, the simulator first chooses a random value $r \in \mathbb{Z}_p$ and finds a vector $\omega = (\omega_1, \ldots, \omega_{n^*})$ where $\omega = -1$ and $\omega \cdot M^*_i = 0$ for $(i \mid \rho^*(i) \in S)$, note that the vector must be exist by the definition of a LSSS, if not, the vector (1,...,0) would be in the span of S. Then the simulator implicitly defines $t = r + \omega_1 a^q + \omega_2 a^{q-1} + \cdots + \omega_{n^*} a^{q-n^*+1}$ by setting $L = g^t = g^r \prod_{i=1,\ldots,n^*} (g^{a^{q-n^*+1}})^{\omega_i}$. The simulator compute K as:

$$K = g^{\alpha} g^{at} = g^{a'} g^{ar} \prod_{i=2,\ldots,n^*} (g^{a^{q+2-i}})^{\omega_i}$$

Note that g^{at} contains a term of $g^{-a^{q+1}}$, which will cancel out with the unknown term in g^{α}, next we will calculate K_x $\forall \in S$, we let $K_x = L^{z_x}$ for $(x \in S|$there is no $\rho^*(i) = x)$. The difficult obstacle is to create K_x for $(x \in S|\rho^*(i) = x)$, note that $K_x = h_x^t$ may contain the form g^{a^{q+1}/b_i} that we can not simulate. However, all of these terms cancel for $M^*_i \omega = 0$, then the simulator creates K_x for $(x \in S|$there is no $\rho^*(i) = x)$ as follows:

$$K_x = h_x^t = L^{z_x} \prod_{i \in X} \prod_{j=1,\ldots,n^*} (g^{(a^j/b_i)r}) \prod_{\substack{k=1,\ldots,n^* \\ k \neq j}} (g^{a^{q+1+j-k/b_i}})^{\omega_k}$$

where $X = (i|\rho^*(i) = x)$.
- *Challenge*. In this phase, the adversary $\mathcal{A}$submits two challenge message M_0andM_1 to simulator. The simulator flips a random cion θ. Then it creates $C{=}M_{\theta}T \cdot e(g^s, g^{\alpha'})$ and $C' = g^s$. However, the difficult is to create the C_i since value contains terms that must cancel out. The simulator will choose random $y_2, \ldots, y_{n^*}$ and use the vector $\upsilon{=}(s, sa{+}y_2, sa^2{+}y_3, \ldots, sa^{n^*-1}{+}y_{n^*})$ to share the secret s.
Now, the simulator chooses random values $r'_1, \ldots, r'_{\ell}$ and creates the challenge ciphertext components as follows: $C_i{=}g^{a\lambda_i} h_{\rho^*(i)}^{-r_i}$, $D_i{=}g^{r_i}$ where $\lambda_i{=}M^*_i \cdot \upsilon$ and $r_i{=}{-}(r'_i + sb_i)$
- *Phase*2. This phase is same as phase 2 with the restrict that the adversary can not issue query I_{key} to $\mathcal{O}_{I_{key}}(\cdot)$ where $f(I_{key}, I'_{enc}) = 1$.
- *Guess*. The adversary will outputs a guess ν' of ν. If $\nu = \nu'$, then the simulator outputs 0 to guess that $T = e(g, g)^{a^{q+1}s}$, it is to say that CT is a normal ciphertext, so the adversary has a non-negligible to guess out $\nu = \nu'$; otherwise, it outputs 1 to indicate that T is a random group element in $\mathbb{G}_T$.
When T is random element in $\mathbb{G}_T$, then the message M_{ν} is random value to adversary, then $\Pr[\mathcal{B}(y, T{=}R) = 0] = \frac{1}{2}$, when T is a tuple the simulator $\mathcal{B}$ gives a perfect simulator, so $\Pr[\mathcal{B}(y, T{=}e(g, g)^{a^{q+1}s}){=}0]{=}\frac{1}{2}$, therefore, $\mathcal{B}$ can play the decisional q-parallel BDHE game with non-negligible advantage, this leads to a contradiction.

Type-II adversary.

Because the different between Type-II and Type-I is that the former can query both $\mathcal{O}_{SK}(\cdot)$ and $\mathcal{O}_{Enc}(\cdot)$, the theorem1 indicates that the adversary can not get anything information about the inputs $(a,b,u_1,u_2){=}(\lambda_i,{-}r_i,g^a,g^{H(att(i))})$, it is to say that the view of Type-II adversary is the same as the view of Type-I adversary, it means that if the Type-II adversary can win the game with a non-negligible advantage, then the Type-I adversary can also win the game1 with a non-negligible advantage, this leads to a contradiction.

Theorem 5: If the construction of ABE-VOED without outsourced decryption scheme is selectively CPA-secure, then the above construction of ABE-VOED is selectively CPA-secure.

Assume that there exists an adversary $\mathcal{A}$ can broke the construction of ABE-VOED with a non-negligible advantage in the selectively CPA-secure model, then we will build a algorithm $\mathcal{B}$ which can attack the basic construction ABE-VOED without outsourced decryption with a non-negligible advantage in the selectively CPA-secure model.

Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the selectively CPA-secure model of the ABE-VOED without decryption. $\mathcal{B}$ interact with $\mathcal{A}$ execute the following steps:

- Init The adversary $\mathcal{A}$ gives $\mathcal{B}$ the challenge access policy I_{enc}, then $\mathcal{B}$ sends it to $\mathcal{C}$ as it challenge access policy, after that $\mathcal{B}$ gives the PK of the basic ABE-VOED to $\mathcal{B}$.
- Setup $\mathcal{B}$ sends the PK to the adversary $\mathcal{A}$.
- Phase1 $\mathcal{B}$ initializes a table T^* and an empty set D. Then the adversary $\mathcal{A}$ adaptively executes query as follows:
  1) SK query for a set of attribute I_{key}: $\mathcal{B}$ calls for $\mathcal{C}$ to generate the $SK_{I_{key}}$ on I_{key}. then $\mathcal{B}$ set $D{=}D\bigcup I_{key}$ and returns to $\mathcal{A}$ the $SK_{I_{key}}$.
  2) TK query for a set of attribute I_{key}: $\mathcal{B}$ checks whether the tuple $(I_{key}, SK_{I_{key}}, TK_{I_{key}})$ in the table T^*, if so, it sends the TK to $\mathcal{A}$. Otherwise, $\mathcal{B}$ randomly chooses $z, z_1 \in \mathbb{Z}_p^*$. Then, $\mathcal{B}$ sets $K'{=}g^z g^{at}, L{=}g^t, k_j{=}g^{H(j)tz_1}, K_i{=}g^t\ \forall\ i{\in}I_{enc}, x \neq j$.

  Finally, $\mathcal{B}$ stores in table T^* the entry $(I_{enc}, *, TK = (K' = g^z g^{at}, L = g^t, k_j = g^{H(j)tz_1}, K_i = g^t), z, z_1)$ and returns to $\mathcal{A}$ the transformation key TK. Note that, $\mathcal{B}$ does not know the actual retrieving key $RK{=}(\alpha/z, z_1)$.
- Challenge The adversary $\mathcal{A}$ submits two equal length message $\mathcal{M}_0$, $\mathcal{M}_1$ and I^*_{enc}. $\mathcal{B}$ sends them to $\mathcal{C}$ to obtain the ciphertext CT^*. after that, $\mathcal{B}$ sends CT^* to the adversary $\mathcal{A}$ as it challenge ciphertext.

  -Query phase2 $\mathcal{A}$ adaptively makes SK queries as in the Query phase1 with the restriction that the set of attributes I_{key} which can not satisfy the access policy I_{enc}.
- Guess The adversary $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ also outputs β'. Then if $\mathcal{A}$ can attack the construction of ABE-VOED with outsourced decryption in the selectively CPA-secure model with a non-negligible advantage, then we build β can attack the construction of ABE-VOED without outsourced decryption in the selectively CPA-secure model with non-negligible advantage.

VI. PERFORMANCE ANALYSIS

A. EFFICIENCY ANALYSIS

In this section, we analyze the efficiency of our scheme, and compare the property and performance of our scheme with the original CP-ABE scheme [3] and the sate-of-the-art [8], [10]–[12] in TABLE 3 and TABLE 4, respectively. First, only the work [8] is similar to our work which delegate the encryption task to server, but it cannot be able to check the correctness of the outputs done by ESP. The schemes [11], [12] are ABE schemes with outsourced decryption, and the scheme [3] is original CP-ABE scheme. In TABLE 3, only our proposed scheme satisfies checkable both outsource encryption and decryption.

In TABLE 4, we present the comparison of the efficiency between [3], [8], [10]–[12] and our scheme, we use Exp to denote a modular exponentiation, $|\ell|$ and $|\omega|$ denote the attribute numbers in the policy and the number of user attributes, respectively. Compared with [3], [10], and [11], as described in Section our scheme is superior in efficiency on the computation cost of DO in encryption phase, DO need to compute 4Exp and O($|\ell|$) multiplications in encryption phase, which is almost constant and not growing with the attributes numbers in the policy of the ciphertext. While the encryption cost in [3] and [10] is growing linearly with the number of attributes in the access policy, because the ciphertext CT in 3 and [10]–[12] is the type of (C, C', C_i, D_i), for $i = 1, ...N'$, where N' is the number of attributes in the access policy, and all the items of CT are computed by DO in [3], [10], and [11]. Compared with [8], the encryption cost on the DO side in our scheme has a little weak advantage than [8], the DO also computes 4Exp in [8]. As described in Algorithm1, DO computes $(\omega_1\omega_2^{-1})^x$ and u^x, where x is a random value such that $x{\geq}2^\lambda$ λ is a security parameter, e.g., $\lambda{=}64$, then the compute time is small than modular exponentiations(g^s and $H(att(i)^s$, where $s{\in}\mathbb{Z}_p^*$, p is 160bit prime) that in [8]. Besides, as the mentioned above, Li *et al.* [8] is secure under the assumption that at least two servers(ESP) are trustworthy, while our assumption is that the ESP can be untrusted, we can check the correctness of the outputs done by ESP. Actually, we deliver the exponentiation computation to ESP and check the correct of computation done by ESP. Regard to outsource decryption, the blinding key cost on user side is reduced to 2Exp in our scheme, while the transformation key cost on user side is growing with the number of attributes associated with user's private key in [11] and [12], because the transformation key $TK{=}SK^z$ are generated by blinding all items of private key $SK{=}(K, L, K_i)$ with a random value z.

B. EXPERIMENTAL PERFORMANCE

In this section, we provide experimental evaluation of our scheme and other schemes on two hardware experimental platforms. In order to explain the necessary of outsourcing encryption, we present the experimental evaluation of our scheme and other schemes without outsourcing encryption [3], [8]. Besides, to explain the advantage of the transformation key cost in our scheme, we give the comparison of generating transformation key cost between our scheme and other schemes [11], [12]. Our hardware experimental platform consists of two types, one is Intel, another is 'mobile platform environment simulated on desktop', abbreviated MPES, whose computation ability is similar to a smart phone or mobile device. More specifically, our experiments are conducted on an Intel Core i3-4170 CPU@3.70GHz×2 processor with 3.2GB RAM running 64-bit ubuntu 16.04LTS and python 3.5.2 under the virtual machine, and on an Intel Core i3-4170 CPU@1.80GHz×2 processor with 1.9GB RAM running 64-bit ubuntu 16.04LTS and python 3.5.2 under the virtual machine which simulates the mobile platform environment.

In order to evaluate the performance of our scheme and other ABE schemes [3], [8], [10]–[12], we implement our scheme in software based on the library [13] and using a

TABLE 3. Feature Comparisons.

Scheme	Policy	Delegate Encryption	Check1	Outsource Decryption	Check2
Waters's ABE	Lsss	–[1]	–	–	–
Green et al's ABE	Lsss	–	–	Yes	–
Lai et al's ABE	Tree	–	–	Yes	Yes
Chen et al's ABE	Tree	Yes	–	–	–
Ours	Lsss	Yes	Yes	Yes	Yes

[1] – denotes this property is not considered in this scheme.

TABLE 4. Efficiency Comparisons.

Scheme	Encryption Cost(DO)	Encryption Cost(ESP)	Blinding Key Cost(DO)
Waters al's ABE	$(3\lvert\ell\rvert + 2)$Exp	–	–
Lai et al's ABE	$(3\lvert\ell\rvert + 2)$Exp	–	$\lvert\omega\rvert$Exp
Chen et al's ABE	4Exp	2n$\lvert\ell\rvert$Exp	–
Ours	4Exp+O($\lvert\ell\rvert$)Multiplication	9$\lvert\ell\rvert$Exp	2Exp

– denotes this property is not considered in this scheme. Exp denotes a modular exponentiation.
$\lvert\ell\rvert$ denotes the attribute number in the policy. $\lvert\omega\rvert$ denotes the number of user attributes.

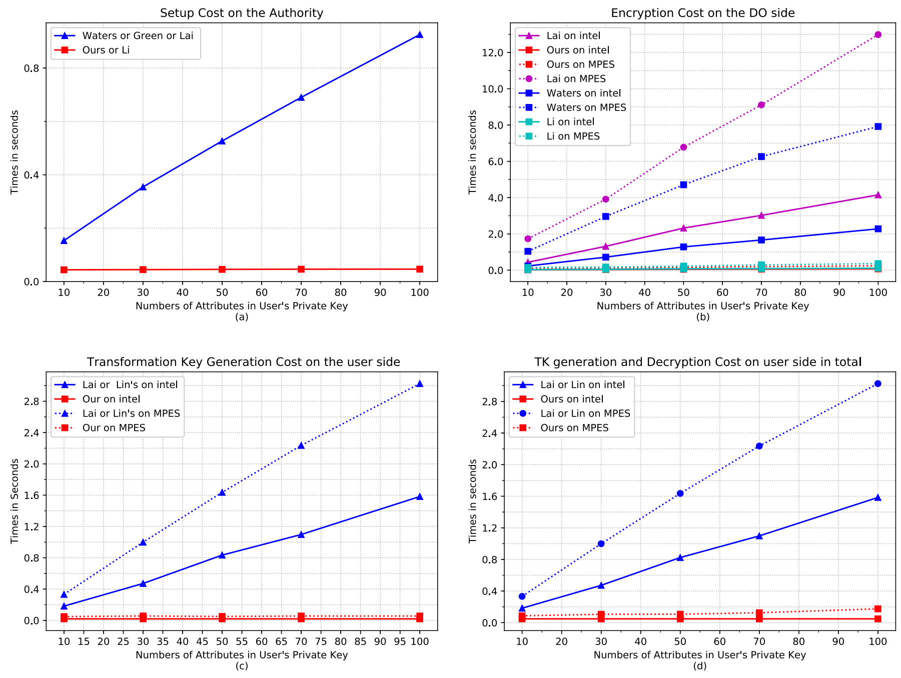

FIGURE 3. Evaluation of our scheme and other schemes.

224-bit MNT elliptic curve from the Stanford Pairing-Based Crypto library [33]. All of the implementation based on the MNT curve implies the use of asymmetric, a small change need to be made on our scheme of symmetric setting in the implementation. Generally speaking, in a CP-ABE scheme, the decryption and the encryption time impacted by the complexity of the ciphertext policy, then, without loss of generality, we generate the ciphertext policies in the form of $(A_1$ or $A_2)$ AND $(A_3$ or $A_4)$...AND $(A_{N-1}$ or $A_N)$, and generate 5 distinct policies in this form with N=10,30,50,70,100. We repeat each experiment 50 times on the desktop computer and 50 times on the MPES and take the average values as the experimental results. In order to realize reduce the computational burden of DO, user delegates the decryption computation to a DSP. In fact, decryption computation cost of user and DSP in total is more than the original ABE scheme, the reason for this is that some additional computation is paid for preserving privacy.

As shown in the FIGURE 3(a), which illustrates that the efficiency comparison between our scheme and [3], [8], [10], and [11] in the setup phase, our scheme cost in the setup reduces to constant as in [10], because the authority will generate the public keys associated with attributes in [3], [10], and [11], e.g., $h_i ... h_U$, while authority public a hash function H in our scheme; FIGURE 3(b) shows the encryption phase efficiency(on the Intel and MPES) comparison between our construction and other ABE schemes. Compared with [8], as the analysis shows in Section uppercase IV.A, our scheme has a little advantage in encryption cost than that in [8], but our scheme can verify the computation do by ESP, while [8] has no this property. Compared with [3] and [10]–[12], no matter implemented on Intel or MPES, our construction encryption cost on the DO side grows close to zero as the number of attributes in the policy increases, when the attributes number N=100, the encryption cost on the DO side is 70ms on the desktop and 240ms on the MPES, respectively. While the corresponding cost on the DO side is growing linearly with the complexity of the ciphertext policy in [3] and [10]–[12], the encryption cost on the DO side is almost 2 seconds and 4 seconds on the desktop in [3] and [11], respectively, and is about 9 seconds and 13 seconds on the MPES in [3] and [11], respectively. Then, due to the most of the operations done by ESP, our scheme is more suitable for mobile users. Thus our construction can be equally applied to mobile devices in practice.

Regarding to decryption phase, FIGURE 3(c)(d) present the comparison of generating the transformation key cost and final decryption cost between our construction and other schemes, no matter on Intel or MPES, the transformation key generation and final decryption cost in our construction is lower than in [11] and [12]. The time implemented on mobile platform is more than 3 seconds with 100 attributes, while the time in our construction is reduced to a constant(ms level), which is not growing with the number of attributes associated with user's private key.

Generally, compared with [3], [8], and [10]–[12], our construction achieves some advantages in encryption and decryption computational cost, respectively, which indicates that our scheme is more suitable for the mobile device in practice.

VII. CONCLUSION

In this paper, we think about a new requirement of ABE with outsourced encryption–verifiability. we propose a secure outsourcing algorithm for exponentiation modulo a prime in one untrusted program model, based on this algorithm, we propose an ABE scheme with verifiable outsourced both encryption and decryption. Besides, in this paper, we also reduce the transformation key cost on the user side into a constant. In the end, we present the security and the efficiency analysis of our scheme. Extensive analysis and the experimental results indicate that our scheme is secure and highly efficient for mobile devices in terms of computation overhead.

REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2005, pp. 457–473.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[3] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.* Berlin, Germany: Springer, 2011, pp. 53–70.

[4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[5] D. Zeng, S. Guo, and J. Hu, "Reliable bulk-data dissemination in delay tolerant networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2180–2189, Aug. 2014.

[6] Z. Guan *et al.*, "APPA: An anonymous and privacy preserving data aggregation scheme for fog-enhanced IoT," *J. Netw. Comput. Appl.*, vol. 125, pp. 82–92, Jan. 2018.

[7] J. Wu, M. Dong, K. Ota, J. Li, and Z. Guan, "FCSS: Fog computing based content-aware filtering for security services in information centric social networks," *IEEE Trans. Emerg. Topics Comput.*, to be published. doi: 10.1109/TETC.2017.2747158.

[8] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," in *Proc. Int. Conf. Inf. Commun. Secur.* Berlin, Germany: Springer, 2012, pp. 191–201.

[9] Z. Zhou and D. Huang, "Efficient and secure data storage operations for mobile cloud computing," in *Proc. Int. Conf. Netw. Service Manage., Int. Fed. Inf. Process.*, 2012, pp. 37–45.

[10] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proc. 20th USENIX Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2011, pp. 1–16.

[11] 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.

[12] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 10, pp. 2119–2130, Oct. 2017.

[13] M. Green, A. Akinyele, and M. Rushanan. (2014). *Libfenc: The Functional Encryption Library*. [Online]. Available: http://code.google.com/p/libfenc

[14] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 456–465.

[15] C.-I. Fan, V. S.-M. Huang, and H.-M. Ruan, "Arbitrary-state attribute-based encryption with dynamic membership," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 1951–1961, Aug. 2014.

[16] X. Mao, J. Lai, Q. Mei, K. Chen, and J. Weng, "Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 533–546, May 2016.

[17] K. T. Nguyen, N. Oualha, and M. Laurent, "Securely outsourcing the ciphertext-policy attribute-based encryption," *Web Inf. Syst. World Wide Web-Internet*, vol. 21, no. 1, pp. 169–183, 2018.

[18] N. Attrapadung *et al.*, "Attribute-based encryption schemes with constant-size ciphertexts," *Theor. Comput. Sci.*, vol. 422, pp. 15–38, Mar. 2012.

[19] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur.*, 2010, pp. 48–59.

[20] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Int. J. Inf. Secur.*, vol. 4, no. 4, pp. 277–287, 2005.

[21] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Proc. 6th Annu. Conf. Privacy, Secur. Trust (PST)*, 2008, pp. 240–245.

[22] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. Int. Conf. Theory Cryptograph.* Berlin, Germany: Springer, 2005, pp. 264–282.

[23] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up discrete log and factoring based schemes via precomputations," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.* Berlin, Germany: Springer, 1998, pp. 221–235.

[24] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital signatures," in *Proc. Conf. Theory Appl. Cryptol.* New York, NY, USA: Springer, 1989, pp. 263–275.

[25] M. Van Dijk, D. Clarke, B. Gassend, G. E. Suh, and S. Devadas, "Speeding up exponentiation using an untrusted computational resource," *Des., Codes Cryptogr.*, vol. 39, no. 2, pp. 253–273, 2006.

[26] X. Ma, J. Li, and F. Zhang, "Outsourcing computation of modular exponentiations in cloud computing," *Cluster Comput.*, vol. 16, no. 4, pp. 787–796, 2013.

[27] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[28] Y. Wang *et al.*, "Securely outsourcing exponentiations with single untrusted program for cloud storage," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2014, pp. 326–343.
[29] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Fac. Comput. Sci., Technion-Israel Inst. Technol., Haifa, Israel, 1996.
[30] P. Q. Nguyen, I. E. Shparlinski, and J. Stern, "Distribution of modular sums and the security of the server aided exponentiation," in *Proc. Workshop Comput. Number Theory Cryptol.*, 1999, pp. 1–16.
[31] R. Canetti, H. Krawczyk, and J. B. Nielsen, "Relaxing chosen-ciphertext security," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2003.
[32] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Fac. Comput. Sci., Technion-Israel Inst. Technol., Haifa, Israel, 1996.
[33] B. Lynn. (2013). *The Stanford Pairing Based Crypto Library*. [Online]. Available: http://crypto.stanford.edu/pbc

ZHIDAN LI received the bachelor's degree from Zhengzhou University, in 2013. He is currently with the Institute of Network and Technology, BUPT. His research interests include the networks security and cryptography protocols.

WENMIN LI received the B.S. and M.S. degrees in mathematics and applied mathematics from Shaanxi Normal University, Xi'an, Shaanxi, China, in 2004 and 2007, respectively, and the Ph.D. degree in cryptology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012. Her research interests include cryptography and information security.

ZHENGPING JIN received the B.S. degree in mathematics and applied mathematics and the M.S. degree in applied mathematics from Anhui Normal University, in 2004 and 2007, respectively, and the Ph.D. degree in cryptography from the Beijing University of Posts and Telecommunications, in 2010, where he is currently a Lecturer. His research interests include cryptography, information security, the internet security, and applied mathematics.

HUA ZHANG received the B.S. degree in telecommunications engineering from Xidian University, in 1998, the M.S. degree in cryptology from Xidian University, in 2005, and the Ph.D. degree in cryptology from the Beijing University of Posts and Telecommunications, in 2008, where she is currently a Lecturer. Her research interests include cryptography, information security, and networks security.

QIAOYAN WEN received the B.S. and M.S. degrees in mathematics from Shaanxi Normal University, Xi'an, China, in 1981 and 1984, respectively, and the Ph.D. degree in cryptography from Xidian University, Xi'an, in 1997. She is currently a Professor with the Beijing University of Posts and Telecommunications. Her current research interests include coding theory, cryptography, information security, the internet security, and applied mathematics.