Received April 13, 2018, accepted May 27, 2018, date of publication July 17, 2018, date of current version August 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2843778

Ciphertext-Policy Attribute-Based Signcryption With Verifiable Outsourced Designcryption for Sharing Personal Health Records

FUHU DENG[1], YALI WANG[1], LI PENG[1], HU XIONG[1,2], JI GENG[1], AND ZHIGUANG QIN[1], (Member, IEEE)

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[2]Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Hu Xiong (xionghu.uestc@gmail.com)

This work was supported in part by the National Science Foundation of China under Grant 61370026 and Grant U1401257, in part by the Science and Technology Project of Guangdong Province under Grant 2016A010101002, in part by the 13th Five-Year Plan of National Cryptography Development Fund for Cryptographic Theory of China under Grant MMJJ20170204, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2016J091, in part by the Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, in part by the Neijiang Science and Technology Incubating Project under Grant 170676, in part by the Sichuan Science and Technology Project under Grant 2018GZ0236 and Grant 2017FZ0004, and in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2015KYQD136.

ABSTRACT Personal health record (PHR) is a patient-centric model of health information exchange, which greatly facilitates the storage, access, and share of personal health information. In order to share the valuable resources and reduce the operational cost, the PHR service providers would like to store the PHR applications and health information data in the cloud. The private health information may be exposed to unauthorized organizations or individuals since the patient lost the physical control of their health information. Ciphertext-policy attribute-based signcryption is a promising solution to design a cloud-assisted PHR secure sharing system. It provides fine-grained access control, confidentiality, authenticity, and sender privacy of PHR data. However, a large number of pairing and modular exponentiation computations bring heavy computational overhead during designcryption process. In order to reconcile the conflict of high computational overhead and low efficiency in the designcryption process, an outsourcing scheme is proposed in this paper. In our scheme, the heavy computations are outsourced to ciphertext transformed server, only leaving a small computational overhead for the PHR user. At the same time, the extra communication overhead in our scheme is actually tolerable. Furthermore, theoretical analysis and the desired securing properties including confidentiality, unforgeability, and verifiability have been proved formally in the random oracle model. Experimental evaluation indicates that the proposed scheme is practical and feasible.

INDEX TERMS Personal health record system, attribute-based signcryption, cloud computing, outsourcing computation.

I. INTRODUCTION

With the rapid development of cloud computing, a large number of companies and individuals utilize the public cloud to store and share data. By outsourcing data in the cloud, the users no longer need to maintain the local storage. Instead, users can store the data in a pay-per-use manner and save the cost of hardware and software deployment. Taking Personal Health Record (PHR) system for example, many PHR services are outsourced to the cloud server to enjoy the benefits of cloud computing. The users can access their PHR data from cloud rather than from the PHR service providers. Undoubtedly, the cloud-assisted PHR system attracts a lot of attention from government and industry. However, it brings a series of questions about security and privacy of the sensitive personal health information of the patients. An unauthorized user may access or modify the PHR data stored in the cloud server. On the other hand, the PHR data collected from patients might be polluted if the malicious adversary delivers the false data to the PHR service provider. Therefore, the most crucial question is how to ensure the PHR data is only available to the users who are authorized by the PHR owner. And how to ensure the data collected from

patients is authentic without disclosing the identity of the patients.

Recently, Attribute-Based Encryption (ABE) [1] has gotten widespread attention in the research community. It realizes fine-grained access control and converts one-to-one communication mode into one-to-many communication modes. Moreover, it can effectively ensure the confidentiality of data. There are two types of ABE, named as key-policy ABE (KP-ABE) [2], [3] and ciphertext-policy ABE (CP-ABE) [4], [5], respectively. In the KP-ABE scheme, attribute set is used to annotate the ciphertexts and access policies are associated with users' private keys. In the CP-ABE scheme, each ciphertext is associated with an access policy, and each user's private key is associated with attribute set. Only when the attribute set satisfies the access policy, the corresponding ciphertext can be decrypted. Similarly, Attribute-Based Signature (ABS) [6] also attracts extensive attention for its ability to sign a message without revealing the identity of the signer. In ABS, a signer who possesses a set of attributes from the authority can sign a message with a predicate that satisfies his attributes. The signature reveals no more than the fact that a single user with some sets of attributes satisfying the predicate has attested to the message.

In order to achieve the confidentiality and authenticity simultaneously, an efficient and flexible method named as Attribute-Based Signcryption (ABSC) [7]. Indeed, ABSC has been applied to secure cloud-assisted PHR system due to its unique characteristics. Although ABSC is a promising security solution to PHR applications, almost all ABSC schemes require a large number of bilinear pairing operations which bring high computational cost on the user side. In addition, the plaintext is signcrypted and uploaded to the cloud server in the PHR system. Then, the signcryption operation only needs to be executed once. However, the frequency of designcryption operation is far greater than signcryption operation. Therefore, how to improve the efficiency of the designcryption in PHR system is a key issue. Recently, Rao [8] introduced a CP-ABSC scheme for cloud-based PHR sharing system which provided fine-grained access control, confidentiality, authenticity, signcryptor privacy and public verifiability, simultaneously. However, due to the large number of bilinear pairings involved in the designcryption process, the computational cost of PHR users has increased significantly. In order to decrease the computational overhead on PHR users, a designcryption scheme with outsourcing mechanism is proposed.

Additionally, the design rationale used in our scheme to achieve verifiable outsourcing designcryption is totally different from the existing ABE schemes with verifiable outsourcing decryption [9]–[14] and server-aided signature verification [15], [16]. In Lai *et al.* [9]'s scheme, the verifiability is realized by appending a redundant ciphertext of a random message and a special tag to each ciphertext. In the final decryption step, the original untransformed ciphertext is required to be an auxiliary input to verify the correctness of the result returned from the cloud server. And in [14], the verifiability is realized by the Pedersen commitment [17] associated with the original ciphertext. Different from the existing works [9]–[14], the verifiable outsourcing of the designcryption in our scheme is achieved by elegantly incorporating the idea of server-aided signature verification [15], [16] and the key blinding technique [18]. As far as we know, this is the first time to achieve attribute-based signcryption scheme with verifiable outsourcing designcryption. Moreover, the design philosophy behind our verifiable outsourcing of designcryption can be viewed as the sophisticated combination of ABE schemes with verifiable outsourcing decryption [9]–[14] and server-aided signature verification [15], [16], which undoubtedly, is novel and different from the existing work.

A. CONTRIBUTIONS

In this paper, we present a new Ciphertext-Policy Attribute-Based Signcryption with Outsourced Designcryption (CP-OABSC) scheme in the cloud-based PHR system. As far as we know, this is the first time to equip the secure outsourcing to the ABSC scheme. The design philosophy behind our verifiable outsourcing of designcryptionan is novel. The major computation in the designcryption process is outsourced to the untrusted cloud server. Only constant computation is required to be run on the PHR user side. Moreover, the result returned by the untrusted cloud server can be verified by the associated user. And the extra communication overhead in our scheme is actually tolerable. The high-level description of our protocol is illustrated in Fig. 1.

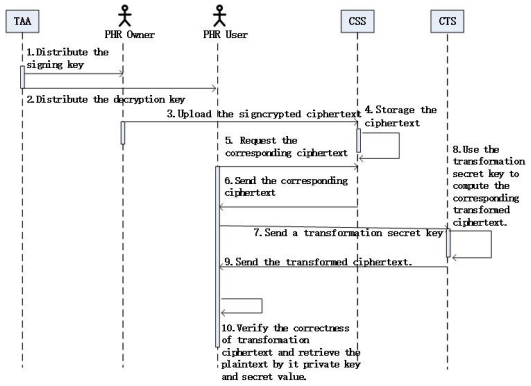


FIGURE 1. High-level description of our protocol.

The main contributions are as follows:

1) Firstly, we formalize the framework of CP-OABSC. After that, the verifiability for the CP-OABSC has also been modeled formally. The design philosophy behind our verifiable outsourcing of designcryption can be viewed as the sophisticated combination of ABE schemes with verifiable outsourcing decryption and server-aided signature verification.
2) In order to reduce the expansion rate of ciphertext, we utilize a mixed signcryption technology, in which an attribute-based encryption method is used

to encapsulate the symmetric key and a symmetric encryption algorithm is used to encrypt the PHR data.

3) We also prove the correctness and security of the proposed scheme and its complexity and efficiency are also analyzed. We further compare our scheme with other ABSC schemes in terms of signing key size, decryption key size and ciphertext size, the computational cost of signcryption and designcryption.

B. ORGANIZATION

The rest of the paper is organized as follows : In section 2, we will briefly summarize the related work about secure outsourcing of attribute-based encryption, attribute-based signature and attribute-based signcryption. In section 3, we will describe the preliminaries, and present the system model and security requirements of CP-OABSC scheme. We propose the concrete scheme in section 4. Subsequently, the security proof and performance analysis of the scheme are presented in section 5 and section 6. Finally, we draw a conclusion in section 7.

II. RELATED WORK

A. SECURE OUTSOURCING OF ATTRIBUTE-BASED ENCRYPTION

To reduce computational cost on the user side, Green *et al*. [18] proposed a decryption algorithm in which complex computation is outsourced to an untrusted third-party and left a small computational overhead for users to recover the plaintext. They utilized a semi-trusted cloud server to transform any ABE ciphertext into an ElGamal-style ciphertext. In addition, the semi-trusted cloud server knows nothing about the plaintext and the user's private key. In order to outsource the decryption computation to a cloud server, a user needs to use his private key to generate a blind key (BK) and a retrieving key (RK). The user transmits the BK to the cloud server, then the cloud server returns an ElGamal-style ciphertext to the sender. Finally, the user utilizes the RK to recover the plaintext. However, in their scheme, a dishonest cloud server may return a fake result by replacing the original ciphertext and its tag with another ciphertext and corresponding tag. To ensure the correctness of the transformation ciphertext returned from the semi-trusted cloud server, Lai *et al*. [9] suggested a concrete construction to verify the correctness of the transformed ciphertext. In their construction, a component which composed of a real message and a random message is introduced. However, the original untransformed ciphertext is also required to be input in the final decryption stage. Compared with [18], this method causes nearly double in both ciphertext size and decryption operation cost.

To increase the efficiency of Lai's scheme in [9], a key encapsulated mechanism (KEM) was introduced in [10] and [11], simultaneously. Their methods decrease the communication cost and computation cost nearly by half compared with Lai's scheme [9]. Similarly, the scheme in [12] encrypts a message and a random number together. The random number is used to realize verifiability. Li *et al*. [13] considered the huge computation overhead at both the attribute authority center side and user side. They presented a new outsourced ABE scheme which not only supporting outsourced decryption but also enabling delegating key generation. In their scheme, the actual attributes and a default attribute are embedded into a user's private key. The computation related to the actual attributes is outsourced to the cloud server and the computation associated with the default attribute is performed by the attribute authority center. Then, the terminal user's private key can be obtained by merging these two parts together. Moreover, the method of outsourced decryption is same as [18]. Ma *et al*. [14] proposed two ciphertext-policy attribute-based key encapsulation mechanisms (CP-AB-KEM). They are the first one to take into account the verifiable of outsourced encryption and outsourced decryption. Furthermore, their mechanisms provide exculpability, which means that the users can't accuse the Decryption Service Providers (DSP) to return incorrect results. Similarly, an outsourced ABE scheme with anti-fraud function was proposed by Xu *et al*. [19]. Their scheme prevents the cloud server from deceiving the users. The encrypted data can't be transformed without the permit even they meet the access conditions actually. Taking into account the overall efficiency of the system, Zhang *et al*. [20] and Wang *et al*. [21] presented the schemes which not only achieved secure outsourced key-issuing, encryption and decryption, but also increased the communication cost at the client side.

Recently, with the development of outsourced ABE, most of researchers have drawn attention to deploy the outsourced ABE technology to reduce the computational cost and communication overhead on user side and authority side. For example, Li *et al*. [22] applied the outsourced key generation and outsourced decryption to an ABE system which holds the keyword search function. In [23], Li *et al*. achieved a highly efficient user revocation by outsourcing both of the computation of encryption and decryption to cloud servers. In [24], Li *et al*. combined the verifiable outsourced decryption technique with the ABE scheme which possesses the property of constant ciphertext length. In [25], Wang *et al*. achieved user's anonymity and multi-authority efficiently through the introduction of outsourcing decryption technology.

B. ATTRIBUTE-BASED SIGNATURE

Fuzzy identity-based signature (IBS) was presented and formalized in [26] and [27] which allows a user to sign with part of his attributes. In addition, the verifier can check whether the signature is signed or not. In order to obtain the same purpose as IBS, the concept of ABS was presented in [28]. However, neither of these types of signatures take the anonymity of the signer into consideration. Considering the anonymity of the signer, Maji *et al.* [6] proposed an ABS scheme which based on groups with bilinear pairings and knew the privacy of the signer. Their construction supports predicate which described by monotone span programs. However, it only proved to be secure in the generic group model.

For the purpose of constructing an efficient ABS scheme with provable security under a standard hardness assumption, Shahandashti and Safavi-Naini [29], Li *et al.* [30] successfully proposed a reliable and secure ABS scheme under the computational Diffie-Hellman assumption. However, these two schemes only support the restricted forms of signature predicates. The signature length of the above two schemes grows linearly with the size of attributes in the predicate. In addition, Escala *et al*. [31] proposed a revocable ABS for threshold predicate, which shares a similar efficiency with Li *et al*. [30]'s work in signing. Recently, Herranz *et al*. [32] proposed two threshold predicate ABS schemes with a constant size of signature. However, the first scheme requires a large number of extensive computations, and the second one involves $O(d^2)$ exponentiations in signing, where d is the upper bound of the threshold value.

Considering more expressive predicate, beyond the previous work in [6], Maji *et al*. [33] proposed a general framework for constructing ABS scheme. They showed three instantiations of monotonic predicates which can be dealt under standard secure assumptions. Based on dual pairing vector spaces, Okomato and Takashima [34] proposed the first fully secure ABS scheme to support the general non-monotone predicate. Then, they built an ABS scheme [35] with similar features in the multi-authority settings. Herranz *et al*. [32] observed that the signature size of all previous ABS schemes grows linearly with the number of attributes. So, they presented two threshold ABS schemes with a constant size of the signature. In particular, the second scheme enjoys the unique feature of supporting large universes of attributes. Although the existing ABS requires a large number of modular exponentiations in signing and the complexity usually grows linearly with the size of the predicate formula in threshold ABS, it is very useful in the private access control, anonymous credentials and trust negotiations etc. applications.

C. ATTRIBUTE-BASED SIGNCRYPTION

The first ABSC was proposed by Gagné *et al*. [7] with formal security definitions of *message confidentiality* and *ciphertext unforgeability* for signcryption in attribute-based setting. In order for users to provide different rights for signature and decryption, signing attributes are separated to decryption attributes. Later, Emura *et al*. [36] designed an ABSC scheme with dynamic property that allows updating the signing access structures without reissuing users' secret keys. The signature part makes use of access trees whereas AND-gate policies are used in encryption and decryption process. Wang and Huang [37] proposed another ABSC scheme by adopting access trees for both signature and encryption parts. The security of [37] is given in the generic group model and random oracle model. By considering the drawbacks of random oracle model, the schemes in [7] and [36] are proved to be secure in the standard model. Hu *et al*. [38] suggested a fuzzy ABSC in order to introduce authenticated access control in body area network, whereas no formal security proof for ciphertext unforgeability is provided in existing security models. Ciphertext size in all of these schemes are increasing linearly with the sum of required signing and encryption attributes. Moreover, the number of expensive bilinear pairing computations are also linear to the number of required attributes. Recently, attribute-based ring signcryption [39] and traceable attribute-based signcryption [40] are also proposed respectively. Rao and Dutta [41] proposed an ABSC scheme with the constant size ciphertext using the technique of key-policy ABS. Liu *et al*. [42] proposed a CP-ABSC for PHR system based on CP-ABE [43] and ABS [33]. However, the CP-ABSC [42] didn't achieve the property of public ciphertext verifiability.

III. PRELIMINARIES

A. BILINEAR GROUPS

Let $\mathcal{G}$ be an algorithm that inputs a security parameter λ and outputs a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, e)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative groups of the same prime order q. A bilinear map $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ has the following properties:

1) Bilinear, that is, for $\forall g \in \mathbb{G}_1$, and $x, y \in \mathbb{Z}_q^*$, the equation $e(g^x, g^y) = e(g, g)^{xy}$ is true;
2) Non-degenerate, that is, the inequality $e(g, g) \neq 1$ is satisfied;
3) Computability, that is, $\forall (g_1, g_2) \in \mathbb{G}_1^2$, there is an efficient algorithm to compute $e(g_1, g_2)$.

B. COMPLEXITY ASSUMPTIONS

1) COMPUTATIONAL DIFFIE-HELLMAN EXPONENT PROBLEM

Taking a bilinear pairing group $D = (q, \mathbb{G}_1, \mathbb{G}_2, e)$ and a distribution of the form

$$\mathcal{D} = \left\{ (g, g^x, g^{x^2}, \ldots, g^{x^n}, g^{x^{n+2}}, \ldots, g^{x^{2n}}) \in \mathbb{G}_1^{2n} \middle| \begin{array}{l} g \xleftarrow{R} \mathbb{G}_1 \\ x \xleftarrow{R} \{2, 3, \ldots, q-1\} \end{array} \right\} \quad (1)$$

Given $\mathcal{D}_{x,g} = (g, g^x, g^{x^2}, \ldots, g^{x^{n+2}}, \ldots, g^{x^{2n}}) \xleftarrow{R} \mathcal{D}$, then it outputs $g^{x^{n+1}}$. The advantage of adversary $\mathcal{A}$ in solving the computational n-Diffie-Hellman Exponent (n-cDHE for short) problem is defined to be $Adv_{\mathcal{A}}^{n-cDHE} = Pr[g^{x^{n+1}} \leftarrow \mathcal{A}(D, \mathcal{D}_{x,g})] \leq \epsilon$.

Definition 1: The n-cDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is said to be (T, ϵ)-hard if the advantage $Adv_{\mathcal{A}}^{n-cDHE} \leq \epsilon$ for all probabilistic polynomial time (PPT) adversary $\mathcal{A}$ running in time at most T.

2) DECISIONAL DIFFIE-HELLMAN EXPONENT PROBLEM

Taking a bilinear group $D = (q, \mathbb{G}_1, \mathbb{G}_2, e)$ and a distribution of the form

$$\mathcal{D} = \left\{ (g, g^\xi, g^x, g^{x^2}, \ldots, g^{x^n}, g^{x^{n+2}}, \ldots, g^{x^{2n}}) \in \mathbb{G}_1^{2x+1} \middle| \begin{array}{l} g \xleftarrow{R} \mathbb{G}_1 \\ \xi, x \xleftarrow{R} \{2, 3, \ldots, q-1\} \end{array} \right\} \quad (2)$$

Given $\mathcal{D}_{\xi,x,g} = (g, g^{\xi}, g^{x}, g^{x^2}, \ldots, g^{x^{n+2}}, \ldots, g^{x^{2n}}) \xleftarrow{R} \mathcal{D}$, then it outputs $X \xleftarrow{R} \mathbb{G}_2$. The decisional n-Diffie-Hellman Exponent (n-cDHE for short) determines whether $X = e(g^{\xi}, g^{x^{n+1}})$ or X is a random element of $\mathbb{G}_2$. The advantage of $\mathcal{A}$ in solving the n-cDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as $Adv_{\mathcal{A}}^{n-dBDHE} = |Pr[1 \leftarrow \mathcal{A}(D, \mathcal{D}_{\xi,x,g}, X)|X = e(g^{\xi}, g^{x^{n+1}})] - Pr[1 \leftarrow \mathcal{A}(D, \mathcal{D}_{\xi,x,g}, X)|X \text{ is random}]|$ is at most ϵ.

Definition 2: The n-dBDHE problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is said to be (T, ϵ)-hard if $Adv_{\mathcal{A}}^{n-dBDHE} \leq \epsilon$ for all PPT adversary $\mathcal{A}$ running in time at most T.

C. MONOTONE SPAN PROGRAMS

The Monotone Span Program (MSP) from [44] is described as follows. Let $\{\upsilon_1, \upsilon_2, \ldots, \upsilon_m\}$ be a set of variables. An MSP over $\mathbb{Z}_q$ is a labeled matrix $\Phi = (\mathbf{M}, \varphi)$, where $\mathbf{M}$ is an $l \times n$ matrix over $\mathbb{Z}_q$ and the labeling function φ maps every row of $\mathbf{M}$ to a literal of $\{\upsilon_1, \upsilon_2, \ldots, \upsilon_m\} \in \{0, 1\}^m$, that is, $\varphi : [l] \rightarrow \{\upsilon_1, \upsilon_2, \ldots, \upsilon_m\}$.

Let the vector $\vec{x} = (x_1, x_2, \ldots, x_m) \in \{0, 1\}^m$ and $X_\mu = \{i \in [l] : [\varphi(i) = \upsilon_j] \wedge [x_j = \mu]\}$ where $\mu = 1$ or 0. Note that $X_1 \cup X_0 = [l]$. Besides, $\vec{M}^{(i)}$ is the ith row of $\mathbf{M}$. We denote $\Phi(\vec{x}) = 1$, if Φ accepts the input $\vec{x}$. Similarly, $\Phi(\vec{x}) = 0$ means Φ rejects $\vec{x}$. Then,

$$\Phi(\vec{x}) = 1 \Longleftrightarrow [\exists\{a_i : i \in X_1\} \subset \mathbb{Z}_q \text{ such that } \sum_{i \in X_1} a_i \cdot \vec{M}^{(i)} = \vec{1}_n] \tag{3}$$

where $\vec{1}_n = (1, 0, \ldots, 0)$ is a vector of length n. If we set $a_i = 0$ for all $i \in X_0$, then Eq.(3) can be rewritten as

$$\Phi(\vec{x}) = 1 \Longleftrightarrow [\exists(a_1, a_2, \ldots, a_l) \in \mathbb{Z}_q^l \text{ such that } \sum_{i \in [l]} a_i \cdot \vec{M}^{(i)} = \vec{1}_n \text{ and } a_i = 0 \quad \forall\, i \in X_0]. \tag{4}$$

In addition, a MSP Φ computes a monotone boolean function $\chi : \{0, 1\}^m \rightarrow \{0, 1\}$ if $\Phi(\vec{x}) = 1$ for all $\vec{x} \in \{\vec{x} : \chi(\vec{x}) = 1\}$. That is, $\Phi(\vec{x}) = 1 \Longleftrightarrow \chi(\vec{x}) = 1$.

Lemma 1: If $\Phi(\vec{x}) = 0$, then there exists a vector $\vec{\omega} = (\omega_1, \omega_2, \ldots, \omega_n) \in \mathbb{Z}_q^n$ with $\omega_1 = -1$. Such that $\vec{\omega} \cdot \vec{M}^{(i)} = 0$ for all $i \in X_1$.

D. PREDICATES

Definition 3: Here we use A to be the universe of attributes. A predicate (over A) is a monotone boolean function whose inputs are related to the attributes of A. An attribute set $U \in A$ is said to satisfy a predicate χ (or χ accepts L) if $\chi(U) = 1$. Here an input is set to be 1 (i.e., true) if its corresponding attribute is a member of U. Otherwise, the input is set to be 0 (i.e., false) if its corresponding attribute is not a member of U. If U doesn't satisfy χ , we denote it by $\chi(U) = 0$.

Since the predicate χ is monotone, $\chi(L) = 1$ indicates $\chi(V) = 1$ for every attribute set $V \supset L$.

Assuming χ is a predicate, L_χ denotes the set of attributes utilized in χ. Then, the corresponding MSP for χ is a labeled matrix $\Phi = (\mathbf{M}_{l \times n}, \varphi)$, where $\varphi : [l] \rightarrow L_\chi$ is a labeling of the rows of $\mathbf{M}$ by attributes from L_χ.

We define $y_1 = \{i \in [l] : [\varphi(i) = u] \wedge [u \in U]\}$ and $y_0 = \{i \in [l] : [\varphi(i) = u] \wedge [u \notin U]\}$. Then $y_1 = \{i \in [l], \varphi(i) \in U\}$ and $y_0 = \{i \in [l], \varphi(i) \notin U\}$. On the other hand, $y_1 \cup y_0 = [l]$, where $U \subset A$ represents an attribute set.

A predicate χ (with its $\Phi = (\mathbf{M}_{l \times n}, \varphi)$) accepts an input attribute set U by the following criterion as stated in *Eq*. (4).

$$\begin{aligned}\chi(U) &= 1 \Longleftrightarrow \Phi(U) = 1 \\ &\Longleftrightarrow [\exists(a_1, a_2, \ldots, a_\ell) \in \mathbb{Z}_q^l \\ &\qquad \text{such that } \sum_{i \in [l]} a_i \cdot \vec{M}^{(i)} = \vec{1}_n \text{ and } a_i = 0 \quad \forall i \in y_0].\end{aligned}$$

Hence,

$$\begin{aligned}\chi(U) = 1 \Longleftrightarrow\ & [\exists(a_1, a_2, \ldots, a_l) \in \mathbb{Z}_q^l \\ & \text{such that } \sum_{i \in [l]} a_i \cdot \vec{M}^{(i)} = \vec{1}_n \\ & \text{and } a_i = 0\ \forall i \text{ where } \varphi(i) \notin U].\end{aligned} \tag{5}$$

The following result (which is an analogue statement of Lemma 1) is very useful to present the security proof of the CP-OABSC scheme which will be proposed in section 3.6.

Lemma 2: Let χ, U be a predicate and attribute set, respectively. If $\chi(U) = 0$, then there exists a vector $\vec{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{Z}_q^n$ with $u_1 = -1$ such that $\vec{u} \cdot \vec{M}^{(i)} = 0$ for all i where $\varphi(i) \in U$.

E. LINEAR ALGEBRA

Lemma 3: Let $\mathbf{M}$ be a matrix of size $l \times n$ over a field $\mathbb{F}$ and $\vec{M}^{(i)}$ be the ith row of the matrix $\mathbf{M}$. Then, the dimension of the vector space $\mathbb{V} = \{(d_1, d_2, \ldots, d_l) \in \mathbb{F}^l : \sum_{i \in l} d_i \cdot \vec{M}^{(i)} = \vec{0}_n\}$ is l - rank($\mathbf{M}$), where $\vec{0}_n = (0, 0, \ldots, 0)$ is a zero vector of length n and rank($\mathbf{M}$) is rank of the matrix $\mathbf{M}$.

It can be seen that rank $(\mathbf{M}) \leq l$.

1) If rank($\mathbf{M}$)$\leq$ l, then $\mathbb{V}$ contains polynomial number of vectors $(d_1, d_2, \ldots, d_l)$ that satisfy $\sum_{i \in l} d_i \cdot \vec{M}^{(i)} = \vec{0}_n$. Under this condition, the corresponding predicate consists of both AND and OR gates.
2) If rank($\mathbf{M}$) $= l$, then the vector space $\mathbb{V} = \{\vec{0}_n\}$. That is, the corresponding predicate is an AND gate.

In our scheme, we utilize the predicates that are boolean functions composed of both AND and OR gates as in [45].

F. SYSTEM MODEL OF CP-OABSC

The architecture of our scheme is illustrated in Fig. 2. In our scheme, the PHR owners upload the signcrypted ciphertext to the Cloud Storage Server (CSS) which is assumed to be fully trusted. When PHR users want to access the PHR data stored on the CSS, they must use their own attribute set to verify whether or not it is satisfy the access policy. Moreover, the users need to send a transformation secret key

to the semi-trusted Ciphertext Transformation Server (CTS). Finally, the CTS returns a transformed ciphertext. At the same time, the PHR user can verify the correctness of transformation ciphertext, and retrieve the plaintext by its private key and secret value. The proposed CP-OABSC scheme consists of the following algorithms:

1) **Setup** (1^{λ}): The **Setup** algorithm is run by the Trusted Attribute Authority (TAA), which takes security parameter λ, attribute universe A as inputs. Then, it outputs the public parameters PK and a master secret key MSK.
2) **DecKeyGen**(PK, MSK, θ_d): The TAA takes MSK, PK and an attributes set θ_d as inputs. Then, it outputs the decryption key SK_{θ_d} for PHR user.
3) **DecKey**$_{blind}(SK_{\theta_d})$: The PHR user takes a decryption key SK_{θ_d} as inputs. Then, it outputs the transformation secret key TSK_{θ_d} and the retrieving secret key RSK_{θ_d}.
4) **SignKeyGen** (PK, MSK, θ_s): The TAA takes MSK, PK and an attributes set θ_s as inputs. Then, it outputs the signing key SK_{θ_s} for PHR owner.
5) **SignKey**$_{blind}(SK_{\theta_s})$: The PHR user takes a signing key SK_{θ_s} as inputs. Then, it outputs the transformation secret key TSK_{θ_s} and the retrieving secret key RSK_{θ_s}.
6) **Signcrypt**$(PK, M_{phr}, SK_{\theta_s}, \chi_s, \chi_e)$: The **Signcrypt** algorithm is run by a PHR owner, which takes the public parameters PK, a PHR file M_{phr}, signer's attribute set θ_s, signing key SK_{θ_s}, signing predicate χ_s and encryption predicate χ_e as inputs. Only in the case of θ_s satisfies χ_s where $\chi_s(\theta_s) = 1$, the PHR owner can signcrypt the PHR data M_{phr}. Finally, it will generate a ciphertext SCT_{χ_e} such that only the PHR user who possesses a set of attributes θ_d which satisfies χ_e will be able to designcrypt the corresponding ciphertext.
7) **Designcrypt**$_{user}$ $(PK, SCT_{\chi_e}, \chi_s, SK_{\theta_d})$: The PHR user takes PK, an attribute set θ_d, a ciphertext SCT_{χ_e} and the decryption key SK_{θ_d} corresponding to θ_d as inputs. Then, it outputs the plaintext M_{phr} or a reject symbol $\perp$.
8) **SignVerify**$_{out}$ $(PK, TSK_{\theta_s}, SCT')$: The CTS takes PK, a transformation secret key TSK_{θ_s} and a partial ciphertext SCT' as inputs. Then, it outputs a verification result VR for PHR user.
9) **Decrypt**$_{out}$ $(PK, TSK_{\theta_d}, E_1, E_3)$: The CTS takes PK, a partial ciphertext E_1 and E_3, a transformation secret key TSK_{θ_d} as inputs. Then, it outputs a transformed ciphertext TCT for PHR user.
10) **DecSignVerify**$_{user}(RSK_{\theta_d}, RSK_{\theta_s}, TCT, VR)$: The PHR user takes the retrieving secret key RSK_{θ_d} and RSK_{θ_s}, a verification result VR and a transformed ciphertext TCT as inputs. Then, it outputs a plaintext M_{phr} or a reject symbol $\perp$.

G. SECURITY MODEL OF CP-OABSC

1) CONFIDENTIALITY

Similar to [8], we use a security game to describe the confidentiality of message. There $\mathcal{C}$ is a challenger and $\mathcal{A}$ is an adversary respectively.

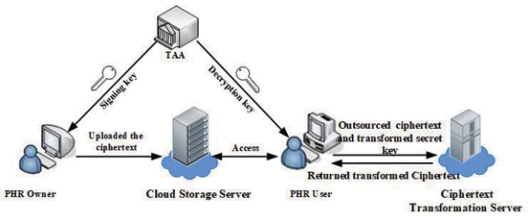


FIGURE 2. Architecture of CP-OABSC scheme.

1) **Setup**: $\mathcal{C}$ runs **Setup** algorithm to get the public parameters PK and a master secret key MSK. Then, it sends PK to $\mathcal{A}$ and keeps MSK to itself.
2) **Query Phase 1**: $\mathcal{C}$ creates an empty table R and an empty set L. Then, $\mathcal{A}$ can adaptively issue the following queries:
   a) **DecKeyGen Query**: For each attribute set θ_d, the challenger $\mathcal{C}$ runs **DecKeyGen** $(PK, MSK, \theta_d) \rightarrow SK_{\theta_d}$ (where $\chi_e^*(\theta_d) = 0$) and sets $R = R \cup \{\theta_d\}$. Then, it sends the decryption key SK_{θ_d} to $\mathcal{A}$.
   b) **DecKey**$_{blind}$ **Query**: On input an attribute set θ_d, $\mathcal{C}$ runs **DecKeyGen**$(PK, MSK, \theta_d) \rightarrow SK_{\theta_d}$, **DecKey**$_{blind}(SK_{\theta_d}) \rightarrow (TSK_{\theta_d}, RSK_{\theta_d})$ and stores the entry $(\theta_d, SK_{\theta_d}, TSK_{\theta_d}, RSK_{\theta_d})$ in table R. Then, it sends TSK_{θ_d} to $\mathcal{A}$.
3) **Challenge Phase**: $\mathcal{A}$ submits two equal length messages m_0, m_1 and a decryption predicate χ_e^*. Note that none of attribute sets in R satisfy χ_e^*. Then, $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$. Finally, $\mathcal{C}$ signcrypts m_b under $\chi_e^*(\theta_d) = 1$ with the signcryption algorithm and sends it to $\mathcal{A}$.
4) **Query Phase 2**: After receiving $SCT_{\chi_e}^*$, $\mathcal{A}$ can continue adaptively to issue queries in the same way as **Query Phase 1** except the **Designcrypt Query**, for any attribute set θ_d and θ_s such that $\chi_e^*(\theta_d) = 1$ $(\chi_s^*(\theta_s) = 1)$.
5) **Guess**: The adversary $\mathcal{A}$ outputs a guess bit $b \in \{0,1\}$ and wins the game if and only if $b' = b$. The advantage of $\mathcal{A}$ in this game is defined as $Adv(\mathcal{A}) = |Pr[b' = b] - \frac{1}{2}|$.

Definition 4: A CP-OABSC scheme is CPA-secure if no polynomial time adversaries who possess a non-negligible advantage win the above security game.

2) VERIFIABILITY

Verifiability of CP-OABSC scheme is described by a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The game consists of the following steps:

1) **Setup**: $\mathcal{C}$ runs **Setup** algorithm to get the PK and MSK. Then, $\mathcal{C}$ sends PK to $\mathcal{A}$.
2) **Challenge**: $\mathcal{A}$ gives a set of attributes θ_d^*, $\mathcal{C}$ runs **DecKeyGen**(PK, MSK, θ_d^*) algorithm to get $SK_{\theta_d}^*$ and sends it to $\mathcal{A}$.
3) **Output**: $\mathcal{A}$ outputs a decryption predicate χ_e and an entry $(SCT_{\chi_e}^*, TSK_{\theta_d}^*, RSK_{\theta_d}^*, SCT_{\chi_e}^{*\prime})$, where

$\chi_e^*(\theta_d^*) = 1$, $SCT_{\chi_e}^*$ is an original ciphertext, $TSK_{\theta_d}^*$ and $RSK_{\theta_d}^*$ are the corresponding transformation secret key and retrieving secret key. Besides, $SCT_{\chi_e}^{*\prime}$ is the transformed ciphertext of $SCT_{\chi_e}^*$. The adversary $\mathcal{A}$ wins the game if $\mathbf{Decrypt}_{out}(PK, TSK_{\theta_d}^*, SCT_{\chi_e}^{*\prime}, RSK_{\theta_d}^*) \notin \{M_{phr}^*, \perp\}$.

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

Definition 5: A CP-OABSC scheme is verifiable if no polynomial time adversaries who possess non-negligible advantage win the above security game.

3) UNFORGEABILITY

The formal definition of unforgeability is based on the following game involving a challenger $\mathcal{C}$ and an adversary $\mathcal{F}$.

1) **Setup**: $\mathcal{C}$ selects a security parameter $\lambda \in \mathbb{N}$ and runs **Setup** algorithm. It obtains the master secret key MSK and sends the public parameters PK to $\mathcal{F}$.
2) **Queries**: Besides a table R and an empty set L, $\mathcal{F}$ adaptively issues the following queries:
   a) **DecKeyGen** and **DecKey**$_{blind}$ **Queries**: These queries are identical to those in the CPA-secure game.
   b) **SignKeyGen Query**: For each θ_s, $\mathcal{C}$ runs **SignKeyGen**$(PK, MSK, \theta_s) \rightarrow SK_{\theta_s}$ (where $\chi_s^*(\theta_s) = 0$) and sets $R = R \cup \{\theta_d\} \cup \{\theta_s\}$. Then, it sends the signing key SK_{θ_s} to $\mathcal{F}$.
   c) **SignKey**$_{blind}$ **Query**: On input an attribute set θ_s, $\mathcal{C}$ runs **SignKeyGen**$(PK, MSK, \theta_s) \rightarrow SK_{\theta_s}$, **DecKey**$_{blind}$ $(SK_{\theta_s}) \rightarrow (TSK_{\theta_s}, RSK_{\theta_s})$ and stores the entry $(*, SK_{\theta_s}, TSK_{\theta_s}, RSK_{\theta_s})$ in table R. Then, it returns TSK_{θ_s} to $\mathcal{F}$.
   d) **SignVerify**$_{out}$: When $\mathcal{F}$ queries the CTS to return the verification result VR associated with attribute set θ_s, $\mathcal{C}$ can answer this query by running the **SignVerify**$_{out}$ algorithm.
3) **Forgey**: Finally, $\mathcal{F}$ outputs a verification result VR^* with the signing predicate χ_s^*.

We say that $\mathcal{F}$ wins the game if i) VR^* is a correct verification result on $TSK_{\theta_s}^*$ with the signing predicate χ_s^*; ii) for any queried attribute set $\theta_s \in A$, $\chi_s^*(\theta_s) \neq 1$; iii) the pair $(VR^*, TSK_{\theta_s}^*)$ has not been submitted to the **SignVerify**$_{out}$ oracle. Hence, the advantage $Adv_{OABSC,F}^{EUF}(\lambda)$ of $\mathcal{F}$ is defined as the probability that it wins the game above.

Definition 6: A CP-OABSC scheme is considered to be secure against existential unforgeability, if no PPT $\mathcal{F}$ can win the security game with a non-negligible advantage.

IV. CONSTRUCTION

In this section, we first modify the original model of Rao's CP-ABSC scheme [8] to allow the PHR user to outsource some designcryption process to a semi-trusted cloud server. Our scheme applies "signature then encryption" and combines with outsourcing technology to the PHR system. In this scheme, both the signing and encryption predicates are represented by Monotone Span Programs (MSPs). Let $\chi = (\mathbf{M}, \varphi)$ be an access structure, it is assumed that the row labeling function φ is injective [43] which is a building block of our CP-OABSC scheme. In addition, we adopt a one-time symmetric-key encryption scheme with key space $\phi = \{0,1\}^{\iota}$ and message space $S = \{0,1\}^*$ that can be defined as $\prod_{SE} = (\text{SE-Enc}, \text{SE-Dec})$. Here, we take the tuple $D = (q, \mathbb{G}_1, \mathbb{G}_2, e)$ as a bilinear group. The remaining algorithms are described as follows.

Setup (1^{λ}): The TAA executes the **Setup** algorithm and selects a security parameter $\lambda \in \mathbb{N}$ as input. It adopts the key derivation function (KDF) and uses ι to denote the length of the output of the KDF. Let A be the universe of attributes and g be a random generator of $\mathbb{G}_1$. The algorithm chooses three hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$, $H_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$, $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and a random exponent $a \in \mathbb{Z}_q^*$, then sets $Y = e(g,g)^a$. For each attribute $\omega \in A$, it picks $\hbar_\omega, \nu, u', v', \eta_1, \eta_2, \mu_0, \mu_1, \mu_2, \ldots \mu_\iota \in_R \mathbb{G}_1$. Finally, it outputs the public parameters as

$$PK = (D, Y, \nu, u', v', \eta_1, \eta_2, \mu_0, \{\mu_i\}_{i\in[\iota]}, \{\hbar_\omega\}_{\omega\in A}, H_1, H_2, H_3, \Pi_{SE}, KDF, S, A)$$

It outputs the master secret key MSK as

$$MSK = g^a.$$

DecKeyGen(PK, MSK, θ_d): If a PHR user wants to join the PHR system, he/she needs to apply for the decryption key from the TAA associated with his/her attribute set $\theta_d \in A$. Then, the TAA runs the **DecKeyGen** algorithm and takes as input PK, MSK and the decryption attributes set θ_d. Then, it chooses a random number $\beta \in \mathbb{Z}_q^*$. Finally, it computes as follows

$$K_d = g^a \nu^\beta, K_d' = g^\beta, K_{d,\omega} = \hbar_\omega^\beta, \forall \omega \in \theta_d$$

So, the decryption key for θ_d is

$$SK_{\theta_d} = (\theta_d, K_d, K_d', \{K_{d,\omega}\}_{\omega\in\theta_d}).$$

DecKey$_{blind}(SK_{\theta_d})$: The **DecKey**$_{blind}$ algorithm takes as input a PHR user's decryption key SK_{θ_d} and chooses a random number $t \in \mathbb{Z}_q^*$. Then, it computes as follows

$$TK_d = g^{at} \nu^{\beta t}, TK_d' = g^{\beta t},$$
$$TK_{d,\omega} = \hbar_\omega^{\beta t}, \forall \omega \in \theta_d.$$

Finally, it outputs the transformation secret key as

$$TSK_{\theta_d} = (TK_d, TK_d', \{TK_{d,\omega}\}_{\omega\in\theta_d}).$$

and outputs the retrieving secret key as $RSK_{\theta_d} = t$.

SignKeyGen (PK, MSK, θ_s): If a PHR owner wants to join the PHR system, he/she needs to apply for the signing key from the TAA associated with his/her attribute set $\theta_s \in A$. Then, the TAA runs the **SignKeyGen** algorithm and takes as input PK, MSK and the signing attributes set θ_s, and chooses a random number $\gamma \in \mathbb{Z}_q^*$. Finally, it computes as follows

$$K_s = g^a \nu^\gamma, K_s' = g^\gamma, K_{s,\omega} = \hbar_\omega^\gamma, \quad \forall \omega \in \theta_s$$

So, the signing key for θ_s is

$$SK_{\theta_s} = (\theta_s, K_s, K_s', \{K_{s,\omega}\}_{\omega\in\theta_s}).$$

SignKey$_{blind}$ (SK_{θ_s}): The **SignKey**$_{blind}$ algorithm takes a PHR user's signing key SK_{θ_s} as input and chooses a random number $s\in \mathbb{Z}_q^*$. Then, it computes as follows

$$TK_s = g^{as}v^{\gamma s}, TK_s' = g^{\gamma s},$$
$$TK_{s,\omega} = \hbar_\omega^{\gamma s}, \forall\omega\in\theta_s.$$

Finally, it outputs the transformation secret key as

$$TSK_{\theta_s} = (\theta_s, TK_s, TK_s', \{TK_{s,\omega}\}_{\omega\in\theta_s}).$$

and outputs the retrieving secret key as $RSK_{\theta_s} = s$.

Signcrypt $(PK, M_{phr}, SK_{\theta_s}, \chi_s, \chi_e)$: If a PHR owner wants to upload a PHR data $M_{phr} \in \{0,1\}^*$ to the cloud server. He/She first formulates an encryption predicate χ_e and a signing predicate χ_s which satisfy $\chi_s(\theta_s) = 1$. The selection of signing predicate with this property preserves the PHR owner's privacy.

The **Signcrypt** algorithm takes as input the public parameters PK, a PHR file M_{phr}, the signing key for the signing attribute set θ_s, a signing predicate χ_s with the property that $\chi_s(\theta_s) = 1$, and the encryption predicate χ_e. Here, $\chi_s = (\mathbf{M}_s, \varphi_s)$ and $\chi_e = (\mathbf{M}_e, \varphi_e)$, where $\mathbf{M}_s(resp.\mathbf{M}_e)$ is an $l_s \times n_s$ (resp. $l_e \times n_e$) matrix with row labeling function $\varphi_s : [l_s] \to A$ (resp. $\varphi_e : [l_e] \to A$). Let $M_s^{(i)}$ (resp. $M_e^{(i)}$) be the ith row of the matrix $\mathbf{M}_s$ (resp. $\mathbf{M}_e$). The plaintext M_{phr} encrypted with the symmetric-key encryption algorithm (such as AES).

Since $\chi_s(\theta_s) = 1$, the algorithm computes a vector $\vec{c} = (c_1, c_2, \ldots, c_{l_s}) \in \mathbb{Z}_q^{l_s}$ such that $\vec{c}\cdot\mathbf{M}_s = \vec{1}_{n_s}$, that is, $\sum_{i\in[l_s]} c_i\cdot\vec{M}_s^i = \vec{1}_{n_s}$, and $c_i = 0$ for all i where $\varphi_s(i)\notin\theta_s$.

The algorithm randomly chooses a vector $(d_1, d_2, \ldots, d_{l_s}) \in \mathbb{Z}_q^{l_s}$ such that $\sum_{i\in[l_s]} d_i\cdot\vec{M}_s^{(i)} = \vec{0}_{n_s}$. Moreover, it picks $\varepsilon, \delta \in_R \mathbb{Z}_q^*$ and sets $\vec{\alpha} = (\varepsilon, \delta_2, \ldots, \delta_{n_e})$, here $\delta_2, \ldots, \delta_{n_e} \in_R \mathbb{Z}_q^*$. Next, the $\mathbf{SE}-\mathbf{Enc}$ algorithm utilizes the KDF to generate a new session key SEK with $key = Y^\varepsilon\|S_1\|tt$, where tt is the current time, and computes

$$\mathsf{KDF}(Y^\varepsilon\|S_1\|tt, \iota) = SEK\|d, S_0 = u'^{H(SEK)}v'^{H(d)}.$$

Then, it chooses a random number $a' \in_R \mathbb{Z}_q^*$ and re-randomizes the signing key SK_{θ_s} as follows

$$K_s^R = K_s\cdot v^{a'}, K_s^{R'} = K_s'\cdot g^{a'}$$
$$K_{s,\omega}^R = K_{s,\omega}\cdot\hbar_\omega^{a'}, \forall\omega\in\theta_s$$

Finally, it computes and outputs the signcrypted message as follows

$$E_1 = g^\varepsilon,\ S_1 = g^{\varepsilon\cdot\delta}$$
$$E_2 = \mathbf{SE\text{-}Enc}(SEK\|d, M_{phr})$$
$$E_3 = \{E_3^{(i)} = v^{\vec{\alpha}\cdot\vec{M}_e^{(i)}}\hbar_{\varphi_e(i)}^\varepsilon\}_{i\in[l_e]}$$
$$\varrho = H_2(E_1),\ E_4 = (\eta_1\eta_2^\varrho)^\varepsilon$$
$$S_2 = \{S_2^{(i)} = g^{d_i}(K_s^{R'})^{c_i}\}_{i\in[l_s]}$$
$$(f_1, \ldots, f_l) \in \{0,1\}^l = H_1(S_2, tt, \chi_s, \chi_e)$$
$$\psi = H_3(S_1, E_2, E_3, E_4, \chi_s, \chi_e)$$
$$S_3 = K_s^R(\prod_{i\in[l_s]}(K_{s,\varphi_s(i)}^R)^{c_i}\cdot\hbar_{\varphi_s(i)}^{d_i})(\mu_0\prod_{i\in[l]}\mu_i^{f_i})^\varepsilon E_4^{\psi\cdot\delta}$$

The ciphertext is published as

$$SCT_{\chi_e} = (\chi_e, E_1, E_2, E_3, E_4, S_0, S_1, S_2, S_3, tt).$$

Now, the PHR owner uploads the signcrypted file $\mathcal{C} = (SCT_{\chi_e}, \chi_s)$ to the cloud server CSS.

Designcrypt$_{user}(PK, SCT_{\chi_e}, \chi_s, SK_{\theta_d})$: When a PHR user wants to access a PHR file, he/she sends the access request to the cloud server CSS. Then, CSS sends back the corresponding PHR data of the form $\mathcal{C} = (SCT_{\chi_e}, \chi_s)$ to PHR user using SSH protocol.

Once receiving the ciphertext SCT_{χ_e}, the **Designcrypt**$_{user}$ algorithm will be executed by a PHR user who possesses a set of authorized attributes. To verify the signature and recover the plaintext, the PHR user first checks the current time tt'. Assume that $\bar{tt}$ is a predefined time limit for message decryption. If $|tt' - tt| \le \bar{tt}$ and $\chi_e(\theta_d) = 1$, the PHR user can verify the ciphertext and decrypt the message. Otherwise, it returns $\perp$.

The **Designcrypt**$_{user}$ algorithm takes as input $PK, SCT_{\chi_e}, \chi_s, SK_{\theta_d}$. It randomly selects $x_2, \ldots, x_{n_s}$ from $\mathbb{Z}_q^*$ and sets $\vec{\varpi} = (1, x_2, \ldots, x_{n_s})\cdot\vec{M}_s^{(i)}, \forall i\in[l_s]$. After that, it computes as follows

$$(f_1, f_2, \ldots, f_l) \in \{0,1\}^l = H_1(S_2, tt, \chi_s, \chi_e)$$
$$\varrho = H_2(E_1)$$
$$\psi = H_3(S_1, E_2, E_3, E_4, \chi_s, \chi_e)$$

Then, the PHR user can proceed in the following way to recover the plaintext M_{phr}.

(1) After receiving the signature S_3, the verification is presented by checking whether the following equation holds:

$$Y \stackrel{?}{=} \frac{e(S_3, g)}{(\prod_{i\in[l_s]} e(v^{\vec{\varpi}_i}\cdot\hbar_{\varphi_s(i)}, S_2^{(i)}))\cdot e(\mu_0\prod_{i\in[l]}\mu_i^{f_i}, E_1)\cdot e((\eta_1\eta_2^\varrho)^\psi, S_1)}$$

Only if it holds, the signature can be proved as valid. The main calculation process is as follows: $S_2^{(i)} = g^{d_i}(K_s^{R'})^{c_i} = g^{d_i+T\cdot c_i}$(where $T = \gamma + a'$)

$$S_3 = K_s^R(\prod_{i\in[l_s]}(K_{s,\varphi_s(i)}^R)^{c_i}\cdot\hbar_{\varphi_s(i)}^{d_i})(\mu_0\prod_{i\in[l]}\mu_i^{f_i})^\varepsilon E_4^{\psi\cdot\delta}$$
$$= g^a v^T(\prod_{i\in[l_s]}\hbar_{\varphi_s(i)}^{T\cdot c_i+d_i})(\mu_0\prod_{i\in[l]}\mu_i^{f_i})^\varepsilon\cdot E_4^{\psi\cdot\delta}$$

(where $T = \gamma + a'$)

$$\sum_{i\in[l_s]}(T\cdot c_i + d_i)\cdot\vec{\varpi}_i$$
$$= \sum_{i\in[l_s]}(T\cdot c_i + d_i)\cdot((1, x_2, \ldots, x_{n_s})\cdot\vec{M}_s^{(i)})$$
$$= (T, Tx_2, \ldots, Tx_{n_s})\cdot\sum_{i\in[l_s]} c_i\cdot\vec{M}_s^{(i)} + (1, x_2, \ldots, x_{n_s})$$

$$
\begin{aligned}
&\cdot \sum_{i\in[l_s]} d_i \cdot \vec{M}_s^{(i)} \\
&= (T, Tx_2, \ldots, Tx_{n_s}) \cdot (1, 0, \ldots, 0) + (1, x_2, \ldots, x_{n_s}) \\
&\quad \cdot (0, 0, \ldots, 0) \\
&= T
\end{aligned}
$$

$$
\frac{e(S_3, g)}{(\prod_{i\in[l_s]} e(v^{\vec{\varpi}_i} \cdot \hbar_{\varphi_{s(i)}}, S_2^{(i)})) \cdot e(\mu_0 \prod_{i\in[l]} \mu_i^{f_i}, E_1) e((\eta_1\eta_2^{\varrho})^{\psi}, S_1)}
$$

$$
\begin{aligned}
&= \frac{e(g, g)^a \cdot e(v, g)^T \cdot e(\prod_{i\in[l_s]} \hbar_{\varphi_{s(i)}}^{T\cdot c_i + d_i}, g)}{e(v, g)^{\sum_{i\in[l_s]} (T\cdot c_i + d_i)\cdot \vec{\varpi}_i} \cdot e(\prod_{i\in[l_s]} \hbar_{\varphi_{s(i)}}^{T\cdot c_i + d_i}, g)} \\
&\quad \times \frac{e(\mu_0 \prod_{i\in[l]} \mu_i^{f_i}, g^{\varepsilon}) \cdot e((\eta_1\eta_2^{\varrho})^{\varepsilon}, g^{\psi\cdot\delta})}{e(\mu_0 \prod_{i\in[l]} \mu_i^{f_i}, g^{\varepsilon}) \cdot e((\eta_1\eta_2^{\varrho})^{\varepsilon}, g^{\psi\cdot\delta})} \\
&= Y
\end{aligned}
$$

(2) Since $\chi_e(\theta_d) = 1$, compute a vector $\vec{k}' = (k_1', k_2', \ldots, k_{l_e}') \in \mathbb{Z}_q^{l_e}$ such that $\vec{k}' \cdot \mathbf{M}_e = \vec{1}_{n_e}$, that is, $\sum_{i\in[l_e]} \vec{k_i}' \cdot \vec{M}_e^{(i)} = \vec{1}_{n_e}$, and $k_i' = 0$ for all i where $\varphi_{e(i)} \notin \theta_d$.

(3) Recover Y^{ε} with the following formula $\frac{e(K_d \cdot \prod_{i\in[l_e]} K_{d,\varphi_{e(i)}}^{k_i'}, E_1)}{e(K_d', \prod_{i\in[l_e]} (E_3^{(i)})^{k_i'})}$. The main calculation process is as follows: $\sum_{i\in[l_e]} k_i' \cdot (\vec{\alpha} \cdot \vec{M}_e^{(i)}) = \vec{\alpha} \cdot \sum_{i\in[l_e]} k_i' \cdot \vec{M}_e^{(i)} = \vec{\alpha} \cdot \vec{1}_{n_e} = (\varepsilon, \delta_2, \ldots, \delta_{n_e}) \cdot (1, 0, \ldots, 0) = \varepsilon$

$$
\begin{aligned}
&\frac{e(K_d \cdot \prod_{i\in[l_e]} K_{d,\varphi_{e(i)}}^{k_i'}, E_1)}{e(K_d', \prod_{i\in[l_e]} (E_3^{(i)})^{k_i'})} \\
&= \frac{e(g^a v^{\beta} \cdot \prod_{i\in[l_e]} \hbar_{\varphi_{e(i)}}^{\beta\cdot k_i'}, g^{\varepsilon})}{e(g^{\beta}, \prod_{i\in[l_e]} v^{k_i'\cdot(\vec{\alpha}\cdot\vec{M}_e^{(i)})} \hbar_{\varphi_{e(i)}}^{\varepsilon\cdot k_i'})} \\
&= \frac{e(g, g)^{a\cdot\varepsilon} \cdot e(v, g)^{\beta\cdot\varepsilon} \cdot e(\prod_{i\in[l_e]} \hbar_{\varphi_{e(i)}}^{k_i'}, g)^{\beta\cdot\varepsilon}}{e(g, v)^{\beta\cdot \sum_{i\in[l_e]} k_i'\cdot(\vec{\alpha}\cdot\vec{M}_e^{(i)})} \cdot e(g, \prod_{i\in[l_e]} \hbar_{\varphi_{e(i)}}^{k_i'})^{\beta\cdot\varepsilon}} \\
&= e(g, g)^{a\cdot\varepsilon} \\
&= Y^{\varepsilon}
\end{aligned}
$$

Finally, the PHR user can obtain the correct plaintext with the following algorithm **SE-Dec**$(SEK \| d, E_2) = M_{phr}$.

SignVerify$_{out}$ $(PK, TSK_{\theta_s}, SCT')$: Firstly, the PHR user re-randomizes the transformation secret key TSK_{θ_s} as follows: $TK_s^R = TK_s \cdot v^{a'}, TK_s^{R'} = TK_s' \cdot g^{a'}, TK_{s,\omega}^R = TK_{s,\omega} \cdot \hbar_{\omega}^{a'}$, $\forall \omega \in \theta_s$. Then, the PHR user recalculates the signature $S_3' = S_3{}^s$. Besides, he/she selects a secret value $t' \in_R \mathbb{Z}_q^*$, and keeps it by himself/herself. Finally, he/she deliveries the transformation signature $TS_3 = S_3' \cdot g^{t'}$ to the cloud server CTS.

The **SignVerify**$_{out}$ algorithm takes as input the public parameters PK and the partial ciphertext $SCT' = (TS_3, TS_2, E_1, S_1)$. Since $\sum_{i\in[l_s]} c_i \cdot \vec{M}_s^{(i)} = \vec{1}_{n_s}$, $\sum_{i\in[l_s]} d_i \cdot \vec{M}_s^{(i)} = \vec{0}_{n_s}$, $\sum_{i\in[l_e]} k_i' \cdot \vec{M}_e^{(i)} = \vec{1}_{n_e}$, $TS_2 = S_2^{(i)} = g^{d_i + T\cdot c_i}$ (where $T = \gamma s + a'$) and $TS_3 = g^{t'} \cdot S_3{}^s$ (where $T = \gamma s + a'$). Then, the CTS can calculate the verification result VR by the following formula:

$$
\frac{e(TS_3, g)}{(\prod_{i\in[l_s]} e(v^{\vec{\varpi}_i} \cdot \hbar_{\varphi_{s(i)}}, TS_2^{(i)})) \cdot e(\mu_0 \prod_{i\in[l]} \mu_i^{f_i}, E_1) e((\eta_1\eta_2^{\varrho})^{\psi}, S_1)} = Y^s \cdot e(g, g)^{t'}
$$

Finally, the CTS returns the verification result $VR = Y^s \cdot e(g, g)^{t'}$ to the corresponding PHR user.

Decrypt$_{out}$ $(PK, TSK_{\theta_d}, E_1, E_3)$: The **Decrypt**$_{out}$ algorithm takes as input the public parameters PK, the transformation secret key TSK_{θ_d}, the partial ciphertext E_1 and E_2. Then, the cloud server CTS calculates the transformed ciphertext TCT by the following formula $\frac{e(TK_d \cdot \prod_{i\in[l_e]} TK_{d,\varphi_{e(i)}}^{k_i'}, E_1)}{e(TK_d', \prod_{i\in[l_e]} (E_3^{(i)})^{k_i'})} = Y^{t\cdot\varepsilon}$.

Finally, The CTS returns the transformed ciphertext $TCT = Y^{t\cdot\varepsilon}$ to the corresponding PHR user.

DecSignVerify$_{user}(RSK_{\theta_d}, RSK_{\theta_s}, TCT, VR)$: The **DecSignVerify**$_{user}$ algorithm takes as input the retrieving secret key $RSK_{\theta_d} = t$, $RSK_{\theta_s} = s$, a verification result $VR = Y^s \cdot e(g, g)^{t'}$ and the corresponding transformed ciphertext $TCT = Y^{t\cdot\varepsilon}$. Then, the PHR users can use their own secret key t' to verify the result of correctness returned from the cloud server CTS. If $Y^s \cdot e(g, g)^{t'} \neq VR$, $S_0 = TCT$, it outputs $\perp$. Otherwise, if $S_0 = u'^{H(SSK)} v'^{H(d)}$, which means the PHR user who satisfies the condition can successfully recover the plaintext **SE-Dec**$(key, E_2) = M_{phr}$.

V. SECURITY PROOF

Theorem 1 (Confidentiality): Suppose the security of Rao's scheme in [8] is guaranteed, then the proposed scheme is secure.

Proof: Assume an adversary $\mathcal{A}$ with non-negligible advantage can attack the above CP-OABSC scheme. Similarly, the scheme in [8] can also be attacked by an algorithm $\mathcal{S}$ with non-negligible advantage.

Let $\mathcal{C}$ be the challenger associated with algorithm $\mathcal{S}$ in the selectively CPA-secure game of Rao's scheme in [8]. $\mathcal{S}$ runs $\mathcal{A}$ to execute the following steps.

1) **Setup**: $\mathcal{C}$ executes **Setup** algorithm in [8] to get the public parameters

$$
PK' = (\Sigma, \Delta, \vartheta, \gamma_1, \gamma_2, y_0, \{y_i\}_{i\in[\ell]}, \{h_x\}_{x\in U}, H_2, H_3, H_4, \Pi_{SE}, \mathsf{KDF}, M, U)
$$

and sends it to $\mathcal{S}$. Also, $\mathcal{S}$ runs **Setup** algorithm in this paper to get the public parameters

$$PK = (D, Y, \nu, u', \nu', \eta_1, \eta_2, \mu_0, \{\mu_i\}_{i\in[\iota]}, \{\hbar_\omega\}_{\omega\in A}, H_1, H_2, H_3, \Pi_{SE}, \mathsf{KDF}, S, A)$$

Finally, it gives PK to $\mathcal{A}$.

2) **Query Phase 1**: Firstly, $\mathcal{S}$ initializes an empty table R and an empty set L. Then, $\mathcal{A}$ adaptively issues the following queries:
   a) **DecKeyGen Query**: If $\mathcal{A}$ makes a decryption key query for a set of attributes θ_d. $\mathcal{S}$ sends θ_d to $\mathcal{C}$ and obtains the decryption key SK_{θ_d}. Then, $\mathcal{S}$ sets $L = L \cup \{\theta_d\}$ and sends SK_{θ_d} to $\mathcal{A}$.
   b) **DecKey**$_{blind}$ **Query**: If $\mathcal{A}$ makes a transformation secret key query for a set of attributes θ_d. $\mathcal{A}$ will search the entry $(\theta_d, SK_{\theta_d}, TSK_{\theta_d}, RSK_{\theta_d})$ in R. If the entry which satisfies the query exits, $\mathcal{S}$ returns TSK_{θ_d}; otherwise, $\mathcal{S}$ chooses a random exponent $t \in \mathbb{Z}_q^*$ and sets $TSK_{\theta_d} = (TK_d, TK'_d, \{TK_{d,\omega}\}_{\omega\in\theta_d})$. Let $TK_d = g^{at}\nu^{\beta t}$, $TK'_d = g^{\beta t}$, $TK_{d,\omega} = \hbar_\omega^{\beta t}, \forall\omega \in \theta_d$. Then TSK_{θ_d} can be another type of SK_{θ_d}. Therefore, in the view of the data owner, it is similar between cloud computing center and users. Finally, $\mathcal{S}$ will store the entry $(\theta_d, *, SK'_{\theta_d}, *)$ in R and return TSK_{θ_d} to $\mathcal{A}$.
3) **Challenge Phase**: $\mathcal{A}$ sends $\mathcal{S}$ the challenge access structure χ_e^*. Then, $\mathcal{S}$ picks two (equal length) messages m_0, m_1 and sends them to $\mathcal{C}$ to obtain a challenge ciphertext $SCT'_{\chi_e} = (\chi_e, E_1, E_2, E_3, E_4, S_0, S_1, S_2, S_3, tt)$ by running **Signcrypt** algorithm of [8]. Finally, $\mathcal{S}$ sends SCT'_{χ_e} to $\mathcal{A}$ as its challenge ciphertext.
4) **Query Phase 2**: $\mathcal{A}$ requests a second series of queries, $\mathcal{S}$ answers these queries in the same way as it simulated in **Query Phase 1**, and returns the answer as **Query Phase 1**.
5) **Guess**: $\mathcal{A}$ outputs its guess b. $\mathcal{S}$ also outputs b.

According to the above discussion, if $\mathcal{A}$ can attack our CP-OABSC scheme in the selectively CPA-secure model with non-negligible advantage. Similarly, $\mathcal{A}$ can attack the scheme in [8]. ■

Theorem 2: (**Verifiability**) In a prime order bilinear group, if the *DL* assumption still holds, the proposed scheme is verifiable.

Proof: Suppose a *PPT* $\mathcal{A}$ can attack the verifiability of our proposed CP-OABSC scheme with non-negligible advantage, we can simulate an algorithm $\mathcal{S}$ to solve the *DL* problem in the prime order bilinear group system with non-negligible advantage.

A tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \widehat{e}, g, \beta = g^x)$ is given to $\mathcal{S}$ and $\mathcal{S}$ wants to calculate $x = log_g\beta$. So, $\mathcal{S}$ interacts with $\mathcal{A}$ as follows:

1) **Setup**: $\mathcal{S}$ chooses $\{\eta_i\}_{i=1}^2$, $\{\mu_i\}_{i\in[\iota]}$, $\{\hbar_w\}_{w\in A}$, $\{H_i\}_{i=1}^3$, ν, μ_0, u', ν', and picks a KDF and a deterministic collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_q^*$. Then, $\mathcal{S}$ sets the public parameters as:

$$PK = (D, Y, \nu, u', \nu', \nu, \eta_1, \eta_2, \mu_0, \{\mu_i\}_{i\in[\iota]}, \{\hbar_\omega\}_{\omega\in A}, H_1, H_2, H_3, \Pi_{SE}, \mathsf{KDF}, S, A)$$

The master secret key is $MSK = g^a$. Finally, $\mathcal{S}$ sends PK to $\mathcal{A}$.
2) **Challenge**: $\mathcal{S}$ runs **DecKeyGen**(PK, MSK, θ_d) algorithm to get the decryption key SK_{θ_d} based on an attribute set θ_d and send it to $\mathcal{A}$.
3) **Output**: $\mathcal{A}$ outputs an encryption predicate χ_e where $\chi_e(\theta_d) = 1$. $\mathcal{S}$ obtains $key^* = (T'_{1,1})^\varepsilon(T'_{1,2})$ as well as $key^{*\prime} = (T'_{2,1})^\varepsilon(T'_{2,2})$, where ε is $TSK^*_{\theta_d}$, then computes $\mathsf{KDF}(key^*, \iota) = SEK^*||d^*$ and $\mathsf{KDF}(key^{*\prime}, \iota) = SEK^{*\prime}||d^{*\prime}$. If $\mathcal{A}$ wins the game (means $H(SEK^*) \neq H(SEK^{*\prime})$), then $\mathcal{S}$ computes $g^{xH(SEK^*)+yH(d^*)} = u'^{H(SEK^*)}\nu'^{H(d^*)} = T'_1 = u'^{H(SEK^{*\prime})}\nu'^{H(d^{*\prime})} = g^{xH(SEK^{*\prime})+yH(d^{*\prime})}$.

Because of the property of KDF, with overwhelming probability, $SEK^* \neq SEK^{*\prime}$. Since H is a deterministic collision-resistant hash function, with overwhelming probability, $H(SEK^*) \neq H(SEK^{*\prime})$, SEK^*, d^*, $SEK^{*\prime}$, $d^{*\prime}$, y and H are known by $\mathcal{S}$. Then, $\mathcal{S}$ computes $x = \frac{y(H(d^{*\prime})-H(d^*))}{H(SEK^*)-H(SEK^{*\prime})}$ as the solution for the *DL* assumption. ■

Theorem 3: (**Unforgeability**) The CP-OABSC scheme in this paper is unforgeable under the assumption of CDH.

Proof: Suppose an adversary $\mathcal{F}$ can break the scheme of this paper with non-negligible advantage, then an algorithm $\mathcal{B}$ can be built to solve the CDH problem. Given $\{g, g^x, g^y\}$ as a random CDH instance, the purpose of $\mathcal{B}$ is to output g^{xy} such that x, y are selected from $\mathbb{Z}_q^*$ at random.

1) **Setup**: The algorithm $\mathcal{B}$ sets $(g_1 = E_1 = g^x, g_2 = \mu_0 = g^y)$ and delivers (g, g_1, g_2) to $\mathcal{F}$ as the public key.
2) **Queries**: Besides a table R and an empty set L, $\mathcal{B}$ adaptively issues the following queries:
   a) **DecKeyGen** and **DecKey**$_{blind}$ **Queries**: These queries are identical to those in the above CPA-secure game.
   b) **SignKeyGen Query**: If $\mathcal{F}$ makes a signing key query for a set of attributes θ_s. $\mathcal{B}$ sends θ_s to $\mathcal{C}$ and obtains the signing key SK_{θ_s}. Then, $\mathcal{B}$ sets $L \cup \{\theta_d\} \cup \{\theta_s\}$ and gives SK_{θ_s} to $\mathcal{F}$.
   c) **SignKey**$_{blind}$ **Query**: If $\mathcal{F}$ makes a transformation secret key query for a set of attributes θ_s. $\mathcal{F}$ will search the entry $(\theta_d, SK_{\theta_d}, SK_{\theta_s}, TSK_{\theta_d}, RSK_{\theta_d})$ in R. If the entry which satisfies the query exits, $\mathcal{B}$ returns TSK_{θ_s}; otherwise, $\mathcal{B}$ chooses a random exponent $s \in \mathbb{Z}_q^*$ and sets $TSK_{\theta_s} = (\theta_s, TK_s, TK'_s, \{TK_{s,\omega}\}_{\omega\in\theta_s})$. Let $TK_s = g^{as}\nu^{\gamma s}$, $TK'_s = g^{\gamma s}$, $TK_{s,\omega} = \hbar_\omega^{\gamma s}, \forall\omega \in \theta_s$. Then TSK_{θ_s} can be another type of SK_{θ_s}. So, in the view of the data owner, it is similar between cloud computing center and users. Finally, $\mathcal{B}$ will store the entry $(\theta_s, *, SK'_{\theta_s}, *)$ in R and return TSK_{θ_s} to $\mathcal{F}$.

TABLE 1. Comparison of communication overheads.

Scheme	Access structure	Signing key size	Decryption key size	Ciphertext size
[8]	Monotone Span Program	$(\|\theta_s\|+2)L_{G_1}$	$(\|\theta_d\|+2)L_{G_1}$	$(l_s+l_e+4)L_{G_1}$
[36]	Monotone Tree	$(4\|\theta_s\|+1)L_{G_1}$	$(4\|\theta_d\|+1)L_{G_1}$	$(4+3l_e)L_{G_1}$
[37]	Monotone Tree	$\|\theta_s\|(L_{Z_q}+2L_{G_1}+2L_{G_2})$	$\|\theta_d\|(L_{Z_q}+2L_{G_1}+2L_{G_2})$	$2L_{G_1}+(2l_e+2l_s+1)L_{G_2}+L_{Z_q}$
[38]	Threshold Policy	$(2\|\theta_s\|+1)L_{G_1}$	$(2\|\theta_d\|+1)L_{G_1}$	$(l_e+2l_s+1)L_{G_1}$
[39]	Threshold Policy	$\|\theta_s\|\cdot L_{G_1}$	$\|\theta_d\|\cdot L_{G_1}$	$(2l_e+1)L_{G_1}+l_s\cdot L_{Z_q}$
[40]	Threshold Policy	$5\|\theta_s\|\cdot L_{G_1}$	$5\|\theta_d\|\cdot L_{G_1}$	$(l_e+3l_s+3)L_{G_1}$
[41]	LSSS-realizable	$(2\|\theta_s\|+2)L_{G_1}$	$(2\|\theta_d\|+2)L_{G_1}$	$(3l_e+3l_s+5)L_{G_1}$
″[13]+[46]″	Threshold Policy	$(8\|\theta_s\|+2)L_{G_1}+2L_{Z_q}$	$2\|\theta_d\|\cdot L_{G_1}$	$(4l_s+l_e+4)L_{G_1}$
[42]	Monotone Span Program	$(\|\theta_s\|+2)L_{G_1}$	$(\|\theta_d\|+2)L_{G_1}$	$(2l_e+2l_s+4)L_{G_1}$
Our scheme	Monotone Span Program	$(\|\theta_s\|+2)L_{G_1}$	$(\|\theta_d\|+2)L_{G_1}$	$2(l_e+l_s+2)L_{G_1}$

‡ L_{Z_q}, L_{G_1} and L_{G_2} denote the length of an element in $\mathbb{Z}_q^*$, $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. $l_s(l_e)$, $|\theta_s|(|\theta_d|)$ indicate the number of attributes in a signing (encryption) predicate, the number of signing (decryption) key attributes respectively. Here we assume the signcryption system supports up to 20 attributes

TABLE 2. Comparison of computation overheads.

Scheme	Signc.	Designc.	Cloud Server
[8]	$(3+2l_s+l_e)E_1$	$(5+l_s)P+(l_e^2+2l_e+l_s+3)E_1$	− − −
[36]	$(2l_e+2l_s+3)E_1$	$(3+2l_e+4l_s)P$	− − −
[37]	$P+2E_1+(2l_e+2l_s+1)E_2$	$7P+(2l_e+l_s)E_1$	− − −
[38]	$P+(l_e+1)E_1+2l_s\cdot E_2$	$(2l_e+3l_s)P$	− − −
[39]	$(l_s+1)E_1+2l_e\cdot E_2$	$(l_e+l_s+2)P$	− − −
[40]	$(l_e+3l_s+2)E_1+P$	$(2l_e+l_s+3)P$	− − −
[41]	$(3l_e+3l_s+4)E_1$	$(5+l_s)P+2l_e\cdot E_1$	− − −
″[13]+[46]″	$(4l_s+l_e+4)E_1+l_s+P$	$2P+(4l_s+2)E_1+l_s$	$(2+2l_e)P+(2+l_e+2l_s)E_1$
[42]	$P+(2l_e+2l_s+3)E_1$	$(3l_s+2l_e+4)P$	− − −
Our scheme	$(4+l_e+2l_s)E_1$	$P+(2+l_s)E_1$	$(2+2l_s)P+(l_s+l_e+2)E_1$

‡ P represents a pairing computation, E_1 and E_2 represents a modular exponentiation computation in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Signc. and Designc. represents the signcryption and designcryption computational cost.

d) **SignVerify**$_{out}$: When $\mathcal{F}$ queries the cloud server to return the verification result VR associated with attribute set θ_s, $\mathcal{C}$ can answer this query by executing the **SignVerify**$_{out}$ algorithm in case $\theta_s \neq \theta_s'$. If it satisfied $\theta_s = \theta_s'$, $\mathcal{C}$ simply aborts.

3) **Forgey**: At this stage, $\mathcal{F}$ outputs a forged verification result $VR^* = (Y^s)^* \cdot (e(g,g)^{t'})^*$ under the attribute set θ_s^* where $\theta_s^* \notin R$. If $\theta_s^* \neq \theta_s'$, $\mathcal{C}$ aborts. Otherwise, the submitted VR^* is valid. Since

$$\frac{e(TS_3, g)\cdot e((\eta_1\eta_2^{\varrho})^{\psi}, S_1)^{-1}}{(\prod_{i\in[l_s]} e(v^{\vec{\varpi}_i}\cdot\hbar_{\varphi_{s(i)}}, TS_2^{(i)}))\cdot e(\mu_0\prod_{i\in[l]}\mu_i^{f_i}, E_1)} = (Y^s)^*\cdot(e(g,g)^{t'})^*$$

It is obvious that

$$\frac{e(TS_3, g)\cdot e((\eta_1\eta_2^{\varrho})^{\psi}, S_1)^{-1}}{(\prod_{i\in[l_s]} e(v^{\vec{\varpi}_i}\cdot\hbar_{\varphi_{s(i)}}, TS_2^{(i)}))\cdot e(g^y\cdot\prod_{i\in[l]}\mu_i^{f_i}, g^x)} = (Y^s)^*\cdot(e(g,g)^{t'})^*$$

According to our setting,

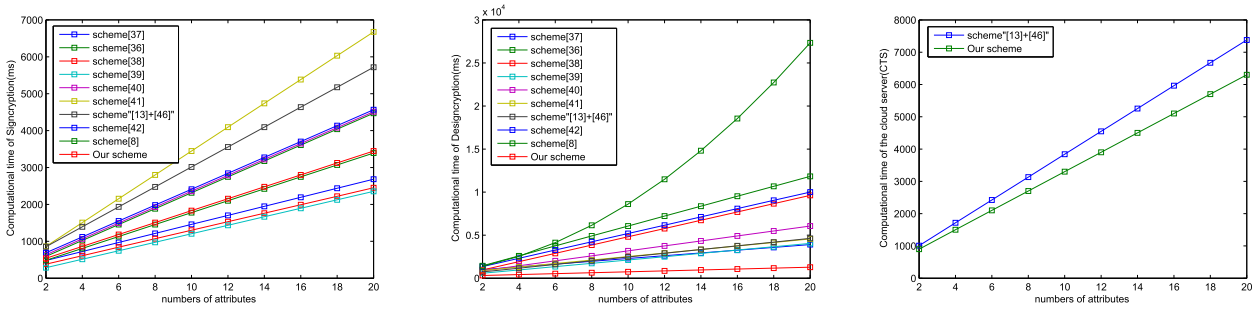
$$g^{xy} = \frac{TS_3}{\prod_{i\in[l_s]}(v^{\vec{\varpi}_i}\cdot\hbar_{\varphi_{s(i)}})^{d_i+Tc_i}\cdot\prod_{i\in[l]}\mu_i^{f_i}\cdot(\eta_1\eta_2^{\varrho})^{\psi\cdot\varepsilon\cdot\delta}\cdot(Y^s)^*\cdot(g^{t'})^*}$$

can be calculated as the solution of the given CDH instance. ■

VI. PERFORMANCE COMPARISON

In this section, we compare the performance of several existing ABSC schemes [8], [36]–[41] " [13]+[46]" [42] with ours in terms of access structure, signing key size, decryption key size and ciphertext size in TABLE 1. In TABLE 2, we compared the time consumed for signcryption and designcryption of ABSC schemes for different number of attributes.

The results in TABLE 1 and TABLE 2 show that our CP-OABSC is an efficient and expressive scheme supporting more generic policies which are desirable in practical applications. Similar to our scheme, the schemes in [8] and [42] adopt monotone span programs, and its expression is rich. Most of the previous works [38]–[40] "[13]+[46]" adopt the threshold policy as their access structure which is only a coarse-grained level and supports only simple predicates. While the schemes in [36] and [37] support monotone tree policy, the construction of this kind of policy is a little more complex. And the scheme in [41] adopts LSSS-realizable monotone access structure that is significantly more efficient than existing ABSC schemes in terms of computation cost and ciphertext size. Since we consider the fact that some attributes may appear more than once in the access structure which is not taken into account by previous works [36]–[40] " [13]+[46]". The size of signing key and decryption key grow linearly with the number of attributes (must be used in signing or decryption). But, both



(a) (b) (c)

FIGURE 3. Performance Evaluation Comparison. (a) Computational time of Signcryption on the PHR owner side. (b) Computational time of Designcryption on the PHR user side. (c) Computational time of the cloud server.

the signing key and decryption key sizes are less than others if the number of attributes are small. On the other hand, it is obvious that the designcryption cost of our scheme is less than those in [8], [36]–[40] " [13]+[46]" [41], [42].

The data in TABLE 1 and TABLE 2 is obtained through simulation experiment. As for the comparison of computation efficiency, our experiment was conducted on an Intel $i7-6700$ processor with $2.53GHz$ and $8GB$ memory running $64-bit$ Windows 10 operation system. For the overall security of our algorithm, we used $SHA-3$ as a hash function. Then, we implemented our experiment in VC++ 6.0 with Stanford Pairing-Based Crypto (PBC) library [47], and set the size of $\mathbb{G}_1$ and $\mathbb{Z}_q$ to $64B(512bits)$. As well as, we set the size of $\mathbb{G}_2$ to $128B$ $(1024bits)$. In addition, the bilinear map is discussed in section 3.1 and the type A bilinear pair is adopted [47]. So, we used the elliptic curve $y^2 = x^3 + x$ defined on providing ECC group. With the above setting, we obtain the results that a pairing operation costs $96.2ms$; an exponentiation operation in $\mathbb{G}_1$ and $\mathbb{G}_2$ costs $53.85ms$ and $30.6ms$, respectively.

As expected, the experimental results demonstrate that our scheme largely eliminates the designcryption overhead for PHR users. As shown in the following figures, Fig. 3(a) depicts the computation time of signcryption on the PHR owner side of CP-OABSC and the other schemes. We found that the time for signcryption increases linearly with the number of attributes, but our scheme still has a lower computational cost of signcryption than the other schemes [8], [36]–[40] " [13]+[46]" [41], [42]. Observation from Fig. 3(b) demonstrates that the computational cost of the PHR user is much lower than the original scheme [8] due to the verifiable outsourcing mechanism is used in our designcryption algorithm. From Fig. 3(c), the computational overhead of the cloud server side is much higher since the bulk of the decryption and verification operation is performed by the cloud server. In addition, it should be noted that in almost all schemes [9], [10], [18] that support outsourcing functionality. Its communication overhead is always increasing relative to the scheme [1]. As shown in Fig. 1, the PHR user only needs to interact with the cloud server (CTS) twice. So, the extra communication overhead in our scheme is actually tolerable.

VII. CONCLUSION

To eliminate the computational overhead of the designcryption process at PHR user side, we studied the attributed-based signcryption scheme [8] and presented an efficient and secure CP-ABSC with verifiable outsourced designcryption scheme. With the help of cloud servers (CSS and CTS), our scheme only needs small modular exponentiation operation to PHR user. Thus, the user saves both bandwidth and local computation time significantly. It greatly improves the efficiency of PHR system. Furthermore, we provided the security proof to show that our scheme is CPA-secure. And the experimental evaluation result demonstrates that the proposed scheme is secure and practicable.

Despite our scheme has only achieved CPA security, we argue that most existing ABSC schemes [37], [39]–[41] also can only achieve CPA security. Only the schemes in [7] and [42] are proved to be CCA2-secure. However, the scheme [7] adopts threshold access structure, and its form is relatively simple. Whereas the security in [42] are proved to be insecure in Rao's scheme [8]. So, the CPA security model has been widely accepted in the public key cryptosystem recently. Furthermore, in both of the CPA and CCA2 security model, the adversary can query the decryption key and transformation secret key of any non-targeted user, reflecting that in the real world, the adversary has the ability to collect decryption key and transformation secret key of any non-targeted user. But beyond that, in the CCA2 security model, the adversary can also decrypt any non-target challenge ciphertext. It also reflects that in the real world adversaries can obtain the plaintext information in any non-challenge ciphertext.

To achieve the CCA2 security, one more decryption oracle should be available to the CPA adversary during the Phase 1 and Phase 2. Since the decryption oracle is added to the CPA security model, the adversary can query decryption oracle to any non-challenge ciphertext. Then, the adversary may use a challenge ciphertext to replace a non-challenge ciphertext

and perform a decryption query on this ciphertext. In this way, the adversary can obtain the target plaintext, and then the adversary can win the security game trivially. In order to prevent the adversary from partially replacing the challenge ciphertext, the plaintext information is obtained by querying decryption oracle. A common method is to sign the generated ciphertext during the encryption process and perform authentication during the decryption process [48]. Our future work consists of designing efficient and provably secure ABSC scheme, which achieves CCA2 security.

REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT*, vol. 3494. Berlin, Germany: Springer, 2005, pp. 457–473.

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

[3] A. Lewko and B. Waters, "Unbounded HIBE and attribute-based encryption," in *Advances in Cryptology—EUROCRYPT*, vol. 6632. Berlin, Germany: Springer, 2011, pp. 547–567.

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

[5] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Advances in Cryptology—EUROCRYPT*, vol. 6110. Berlin, Germany: Springer, 2010, pp. 62–91.

[6] H. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures: Achieving attribute-privacy and collusion-resistance," IACR Cryptol. ePrint Arch., Tech. Rep. 2008/328, 2008, p. 328. [Online]. Available: http://eprint.iacr.org/2008/328

[7] M. Gagné, S. Narayan, and R. Safavi-Naini, "Threshold attribute-based signcryption," in *Security and Cryptography for Networks*, vol. 6280. Berlin, Germany: Springer, 2010, pp. 154–171.

[8] Y. S. Rao, "A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing," *Future Gener. Comput. Syst.*, vol. 67, pp. 133–151, Feb. 2017.

[9] 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.

[10] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1384–1393, Jul. 2015.

[11] 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. 2015.

[12] 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. Depend. Sec. Comput.*, vol. 13, no. 5, pp. 533–546, May 2016.

[13] 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. 2014.

[14] H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 6, pp. 679–692, Nov./Dec. 2015.

[15] W. Wu, Y. Mu, W. Susilo, and X. Huang, "Server-aided verification signatures: Definitions and new constructions," in *Proc. Int. Conf. Provable Secur.*. Berlin, Germany: Springer, 2008, pp. 141–155.

[16] S. S. M. Chow, M. H. Au, and W. Susilo, "Server-aided signatures verification secure against collusion attack," *Inf. Secur. Tech. Rep.*, vol. 17, no. 3, pp. 46–57, 2013.

[17] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Annu. Int. Cryptol. Conf.*. Berlin, Germany: Springer, 1991, pp. 129–140.

[18] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Secur. Symp.*, 2011, p. 34.

[19] J. Xu, Q. Wen, W. Li, and Z. Jin, "Circuit ciphertext-policy attribute-based hybrid encryption with verifiable delegation in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 119–129, Jan. 2016.

[20] R. Zhang, H. Ma, and Y. Lu, "Fine-grained access control system based on fully outsourced attribute-based encryption," *J. Syst. Softw.*, vol. 125, pp. 344–353, Mar. 2017.

[21] H. Wang, D. He, J. Shen, Z. Zheng, C. Zhao, and M. Zhao, "Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing," *Soft Comput.*, vol. 21, no. 24, pp. 7325–7335 , 2016.

[22] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 715–725, Sep./Oct. 2017.

[23] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 785–796, Sep./Oct. 2016.

[24] J. Li, F. Sha, Y. Zhang, X. Huang, and J. Shen, "Verifiable outsourced decryption of attribute-based encryption with constant ciphertext length," *Secur. Commun. Netw.*, vol. 2017, Jan. 2017, Art. no. 3596205.

[25] H. Wang, D. He, and J. Han, "VOD-ADAC: Anonymous distributed fine-grained access control protocol with verifiable outsourced decryption in public cloud," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2017.2687459.

[26] A. Burnett, F. Byrne, T. Dowling, and A. Duffy, "A biometric identity based signature scheme," *IJ Netw. Secur.*, vol. 5, no. 3, pp. 317–326, 2007.

[27] P. Yang, Z. Cao, and X. Dong, "Fuzzy identity based signature," IACR Cryptol. EPrint Arch., Tech. Rep. 2008/002, 2008, p. 2. [Online]. Available: http://eprint.iacr.org/2008/002

[28] G. Shanqing and Z. Yingpei, "Attribute-based signature scheme," in *Proc. IEEE Int. Conf. Inf. Secur. Assurance (ISA)*, Apr. 2008, pp. 509–511.

[29] S. F. Shahandashti and R. Safavi-Naini, "Threshold attribute-based signatures and their application to anonymous credential systems," in *Progress in Cryptology—AFRICACRYPT*, vol. 9. Berlin,ăGermany: Springer, 2009, pp. 198–216.

[30] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur.*, 2010, pp. 60–69.

[31] A. Escala, J. Herranz, and P. Morillo, "Revocable attribute-based signatures with adaptive security in the standard model," in *Progress in Cryptology—AFRICACRYPT*. Berlin,ăGermany: Springer, 2011, pp. 224–241.

[32] J. Herranz, F. Laguillaumie, B. Libert, and C. Ràfols, "Short attribute-based signatures for threshold predicates," in *Topics in Cryptology—CT-RSA*, vol. 7178. Berlin, Germany: Springer, 2012, pp. 51–67.

[33] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology—CT-RSA*, vol. 6558. Berlin, Germany: Springer, 2011, pp. 376–392.

[34] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in *Proc. Int. Workshop Public Key Cryptogr.*. Berlin, Germany: Springer, 2011, pp. 35–52.

[35] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *Public-Key Cryptography—PKC*. Berlin, Germany: Springer, 2013, pp. 125–142.

[36] K. Emura, A. Miyaji, and M. S. Rahman, "Dynamic attribute-based signcryption without random oracles," *Int. J. Appl. Cryptogr.*, vol. 2, no. 3, pp. 199–211, 2012.

[37] C. Wang and J. Huang, "Attribute-based signcryption with ciphertext-policy and claim-predicate mechanism," in *Proc. IEEE 7th Int. Conf. Comput. Intell. Secur. (CIS)*, Dec. 2011, pp. 905–909.

[38] C. Hu, N. Zhang, H. Li, X. Cheng, and X. Liao, "Body area network security: A fuzzy attribute-based signcryption scheme," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 37–46, Sep. 2013.

[39] Z. Guo, M. Li, and X. Fan, "Attribute-based ring signcryption scheme," *Secur. Commun. Netw.*, vol. 6, no. 6, pp. 790–796, 2013.

[40] J. Wei, X. Hu, and W. Liu, "Traceable attribute-based signcryption," *Secur. Commun. Netw.*, vol. 7, no. 12, pp. 2302–2317, 2014.

[41] Y. S. Rao and R. Dutta, "Expressive attribute based signcryption with constant-size ciphertext," in *Proc. Int. Conf. Cryptol. Africa*. Berlin, Germany: Springer, 2014, pp. 398–419.

[42] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Generat. Comput. Syst.*, vol. 52, pp. 67–76, Nov. 2015.

[43] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography—PKC*, vol. 6571. Berlin, Germany: Springer, 2011, pp. 53–70.

[44] J. Chen, H. W. Lim, S. Ling, L. Su, and H. Wang, "Spatial encryption supporting non-monotone access structure," *Des., Codes Cryptogr.*, vol. 73, no. 3, pp. 731–746, 2014.
[45] C. Chen, J. Chen, H. W. Lim, Z. Zhang, and D. Feng, "Combined public-key schemes: The case of ABE and ABS," in *Proc. Int. Conf. Provable Secur.*. Berlin, Germany: Springer, 2012, pp. 53–69.
[46] X. Chen, J. Li, X. Huang, J. Li, Y. Xiang, and D. S. Wong, "Secure outsourced attribute-based signatures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3285–3294, Dec. 2014.
[47] B. Lynn *et al.* (2011). *PBC: The Pairing-Based Cryptography Library*. [Online]. Available: https://crypto.stanford.edu/pbc/
[48] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Proc. Int. Workshop Public Key Cryptogr.*. Berlin, Germany: Springer, 2008, pp. 360–379.

FUHU DENG received the B.Eng. and M.Eng. degrees in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2006 and 2009, respectively, and the Ph.D. degree in wireless communications from the Dublin Institute of Technology, Ireland, in 2014. His research interests include network security, wireless communications and resource management. He is a member of the ACM.

YALI WANG received the B.S. degree from GanSu Agricultural University. She is currently pursuing the M.S. degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China. Her research interests include attribute-based signcryption and malicious code detection.

LI PENG received the B.S. degree from Guangxi University. He is currently pursuing the M.S. degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China. His research interests include attribute-based encryption and malicious code detection.

HU XIONG received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC) in 2009. He is currently an Associate Professor at UESTC. His research interests include cryptography and ad hoc networks security.

JI GENG received the M.S. degree from Southwest Jiaotong University and the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC). He has long been involved in distributed computing and information security research. He is currently a Professor with the School of Information and Software Engineering, UESTC.

ZHIGUANG QIN (M'04) received the Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), in 1996. He is currently a Full Professor and the Dean of the School of Information and Software Engineering, UESTC. His research interests include computer networking, information security, and cryptography.