An Efficient Revocable Attribute-Based Signcryption Scheme With Outsourced Unsigncryption in Cloud Computing

NINGZHI DENG\(^1\), SHAOJIANG DENG\(^1\), CHUNQIANG HU\(^2\), (Member, IEEE), AND KAIWEN LEI\(^1\)

\(^1\)School of Computer Science, Chongqing University, Chongqing 400044, China
\(^2\)School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China

Corresponding authors: Shaojiang Deng (sj\_deng@cqu.edu.cn) and Chunqiang Hu (hcq0394@gmail.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102600, in part by the National Natural Science Foundation of China under Grant 61702062, Grant 61672119, and Grant 61932006, in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2018jcyjAX0334, in part by the Key Project of Technology Innovation and Application Development of Chongqing under Grant CSTC2019jscx-mbdxX0044, in part by the Fundamental Research Funds for the Central Universities under Grant 2019CDQYRJ006, and in part by the Overseas Returnees Innovation and Entrepreneurship Support Program of Chongqing under Grant cx2018015.

**ABSTRACT** In this paper, we present an efficient revocable signcryption (CP-ABSC) with outsourced unsigncryption scheme based on ciphertext-policy attribute based encryption (CP-ABE) to secure the data sharing in cloud computing that require access control, data encryption, and authentication to ensure message integrity and confidentiality. It can be used to signcrypt a message based on the access rights specified by the message itself. A user can decrypt a ciphertext if and only if it possesses the attributes required by the access structure of the data. Thus CP-ABSC flexibly encrypts data based on the access rights of the data specified by the data itself, which differs significantly from the traditional encryption where the user can decrypt is predetermined and must be known by the data source. In addition, the proposed scheme handles attributes revocation with high efficiency and demands low computation overhead on users’ device during unsigncryption phase. Our scheme provides collusion attack resistance, message authentication, forgery prevention, and confidentiality. Furthermore, we compare the performance of our scheme with others in terms of the ciphertext, key size, computation cost and functionality.

**INDEX TERMS** Attribute-based encryption, attributes revocation, outsourced decryption, ciphertext-policy.

**I. INTRODUCTION**

Recently more and more companies and individuals take advantage of the public cloud to store and share data. For the users, outsourcing data in the cloud avoids maintaining the local storage and pay-per-use manner saves the cost of hardware and software deployment. However, when sensitive data is stored in the cloud, how to ensure the confidentiality of these data is a major obstacle to the widespread deployment and adoption of cloud computing [2]. In general, the significant threats in cloud computing are:

- **Integrity:** An adversary may modify or destruct the messages (e.g. content, timing, sequence order, etc.) between the cloud services and the data sources before they are transmitted, or the dishonest cloud services provider may discard some part of the stored data.
- **Confidentiality:** An adversary can gain accesses to unauthorized data without permission or qualifications, which could result in loss or theft of data. This attack raises a security issue that could expose a user’s private information.
- **Authentication and Access Control:** In the scenario where the user stores sensitive data in cloud computing environment, access control is the key to ensuring the legal access of sensitive data and authentication can guarantee the legitimacy of the data and data source.

The primary security requirements for sharing data through the cloud are: confidentiality, access control, authentication, and verifiability. To effectively meet the aforemen-
tioned requirements, an efficient security mechanism need to be designed. Attribute-Based Encryption (ABE) developed by [3] is a promising solution that can provide confidentiality and fine-grained access control. ABE is a one-to-many public key encryption that embedding attributes into the private key or ciphertext. There are two variants of ABE, named as key-poly AB (KP-ABE) [4] and ciphertext-policy AB (CP-ABE) [5], respectively. In CP-ABE, decryption algorithm can success only when the attribute set in private key satisfies the access policy in ciphertext. In this way, data user with some specific attributes can access the sensitive data without knowing the data owner’s unique identity information [6]. A signature analogue of ABE is Attribute-Based Signature (ABS) [7] which provides authentication and data integrity meanwhile hiding the identity of the signer. In ABS, a signer signs a message through private key which the authority embeds the signer’s attributes into it. And the signature indicates the attributes of whom authenticated it. Similarly, ABS can be categorized into Signature-Policy ABS and Key-Policy ABS.

On the basis of previous method, the Attribute-Based Signcryption (ABSC) which achieve the confidentiality, fine-grained access control and anonymous authenticity simultaneously is proposed. ABSC takes advantage of the ABE and ABS and is more efficient than “encrypt-then-sign” or “sign-then-encrypt” approach. The Ciphertext-Policy ABSC (CP-ABSC) is the combination of CP-ABE and SP-ABS. And the Key-Policy ABSC (KP-ABSC) is a combination of KP-ABE and KP-ABS. The CP-ABE schemes with public verifiability mechanism are more appropriate than KP-ABSC in cloud storage applications because the encryptor in former schemes has right to define the access policy to decide who can access the plaintext [8].

However the existing implementations of ABSC have some problems that hinder the practical applications in cloud. One of them is the high computational overhead on user sides. In many previous ABSC schemes, the user sides need to compute a large number of bilinear pairing operations which are complicated and time-consuming. In order to alleviate the computation overhead on user sides, several outsourced ABE schemes were then put forward [9]–[11]. This approach can be extended into our signcryption scheme. And another shortage of almost all ABSC is the revocation mechanism for attributes. It is a significant requirement since attributes of users can be changed dynamically in practice. However, attribute revocation is non-trivial issue in attribute-based encryption (as same as signature), since each attribute is shared by multiple users. In other words, the revocation of any attribute or user will affect the other users. In native solution, data re-encryption and user secret key updates bring too heavy computation and communication overhead, which may cause bottlenecks in the revocation process.

A. OUR CONTRIBUTIONS

In this paper, we propose a new ciphertext-policy revocable attribute-based signcryption with outsourced designcryption scheme. This scheme improves the efficiency and security related to the aforementioned issues. Our main contributions are outlines as follows:

- We formalize the framework of ciphertext-policy revocable attribute-based signcryption with outsourced designcryption scheme. The method of our scheme is integrating revocable ABE with verifiable outsourcing decryption and server-aided signature verification.
- We implement an efficiently immediate attributes revocation approach. The cloud server is delegated the task of keep the elements in ciphertext and keys up to date. Benefiting from that the computation and communication overhead Involved to revocation can be distinctly alleviated.
- The correctness and security of the proposed scheme are theoretically proved. We also analyze the efficiency and feasibility of the proposed scheme. In particular, Our scheme is proven to be confidentiality under adaptive chosen plaintext attack. And the security analysis also shows the resistant against collusion attacks and unforgeability. We further compare our scheme with existing schemes in terms of performance and functionality.

B. ORGANIZATION

The rest of the paper is organized as follows. Section II overview the related work. In section III, we will describe the preliminaries. We then present the system and security model in Section IV. Our construction is described in Section V. In Section VI and Section VII, security analysis and performance analysis are discussed. Finally, we conclude the paper in Section VIII.

II. RELATED WORK

A. ATTRIBUTE-BASED ENCRYPTION WITH ATTRIBUTE REVOCATION

Due to the users’ attributes may change frequently within the system in reality, the revocation of attributes in ABE applications for real-world applications is a necessary mechanism and hard problem. In [12], the author proposed an attribute revocable ABE schemes extended from their revocable IBE schemes [46] [20]. In their scheme, each attribute is assigned to an expiration date, which requires the authority center to reissue keys periodically. Thus, the authority center revokes attributes in the system by stopping publishing and updating new versions of properties There are two main problems with this approach: it exists a span of time which is vulnerability [13]. And periodically key updating produces a heavy load for both the authority center and all non-revoked users. Attrapadung et al. [14] proposed an ABE scheme that realized user revocation through maintaining the revoked users’ indices list. It avoids frequently updating secret keys and ciphertext. However this scheme does not consider only attributes revocation and the data owner needs to manage the revocation list.
In recent years, a new construction of CP-ABE, achieved immediate attribute revocation, was presented in [15] which improves the above drawbacks. In their construction, it allows an untrusted server (such as the cloud) to convert a ciphertext into a new ciphertext without decrypting it, which is called proxy re-encryption. Li et al. [16] presented an efficient CP-ABE scheme with user revocation, and the lower computation cost through outsourcing both encryption and decryption to cloud servers. Later they also presented a CP-ABE scheme [17] with attribute revocation, which resists collusion attack by existing users and revoked users. And Several recent schemes on attribute revocation are proposed [18]–[21]

B. OUTSOURCING DECRYPTION

Green et al. [9] proposed an efficient approach to lighten computation overhead in the decryption. It largely reduces decryption computational cost on the users’ devices, for the complex computation is outsourced to a semi-trusted proxy server. However, it exists a serious problem which is fake result from a dishonest cloud server can not be distinguished by users. On the basis of previous outsourced decryption schemes, an extended ABE scheme that allowed users to verify the correctness of transformed ciphertexts was proposed by Lai et al. [10]. In order to improve the efficiency, Qin et al. [11] introduced a key encapsulated mechanism based on et al. [10]. Ma et al. [22] provided a verifiable and exculpable ABE scheme. Their scheme outsourced complex computation in the both encryption and decryption phase. Furthermore, it provide exculpability, that is the users can not deny the results returned by the cloud server. With the development of outsourced ABE, researchers deploys the outsourced ABE technology in more ways. Li et al. [23] proposed a keyword search ABE scheme which outsourced computation in key generation and decryption. A multi-authority ABE scheme with outsourced encryption and decryption for mobile cloud computing is present by Shao et al. [24]. In [25], Liu et al. achieved highly efficient user revocation through outsourcing both of the computation of encryption and decryption to cloud servers.

C. ATTRIBUTE-BASED SIGNCRYPTION

Signcryption simultaneously implements the both signing and encryption in a single logical step. The attribute-based signcryption (ABSC), first introduced by Gagne’ et al. [26], combines attribute-based encryption and attribute-based signature with restriction that the access structure is fixed in the setup phase. Wang et al. [27] proposed an ciphertext-policy ABSC scheme, which combines data encryption and signing through an expressive access tree. In order for users to provide different rights for signature and decryption, there are two attributes sets named ciphertext-policy and claim-predicate. Emura et al. [28] present a dynamic encryptor attribute-based signcryption scheme, which allow users to be free of updating private keys. Its security is proved in the standard model. Hu et al. [29] presented an ABSC scheme for the body area network applications. In their scheme, it possesses the ability of error tolerance for the attributes. Liu et al. [30] implemented a secure sharing scheme for a personal health records system based on CP-ABSC. Later, Rao [31] proved that the scheme [30] fails to provide confidentiality and public ciphertext verifiability. And they also presented a provable secure CP-ABSC scheme for cloud-based PHR sharing system, which can protect the signcryption privacy and provide public verifiability simultaneously. To take advantage of ciphertext-policy and key-policy in encryption and signature respectively, a hybrid access policy ABSC scheme was proposed by Yu et al. [32]. A novel decentralized KP-ABSC scheme was proposed by Meng and Meng [33]. It adopted multi-authority to strengthen the security and privacy of users. Deng et al. [34] proposed an outsourced designcryption CP-ABSC scheme in the cloud-based personal health records system. However, the above mentioned ABSC schemes all do not consider revocability. In their scheme, it required large computation and communication cost to support attribute revocation.

III. PRELIMINARIES

In this section, we summarize some mathematical problems for our scheme construction and security analysis.

A. BILINEAR MAPS

Let $G_0$ and $G_T$ be two multiplicative cyclic groups with the prime order $p$. A bilinear map $e : G_0 \times G_0 \rightarrow G_T$ satisfying the following properties:

1) Bilinear: for all $P, Q \in G_0$ and $a, b \in \mathbb{Z}_p^n$, the equation $e(P^a, Q^b) = e(P, Q)^{ab}$ is true.

2) Non-degeneracy: The generator $g$ satisfies $e(g, g) \neq 1$, where $I$ is the identity element of group $G_T$.

3) Computability: There is an effective polynomial time algorithm to compute $e(P, Q) \in G_0$, for all $P, Q \in G_0$.

B. DECISIONAL BILINEAR DIFFIE-HELLMAN (DBDH) ASSUMPTION

Let $e : G_0 \times G_0 \rightarrow G_T$ be a bilinear map, both $G_0$ and $G_T$ are cyclic groups of prime order $p$. Choose a random generator $g$ of $G_0$ and random $a, b, c, \theta$ from $\mathbb{Z}_p^n$. The decisional Bilinear Diffie-Hellman (DBDH) Assumption is to distinguish between the tuples of the form $(g, g^a, g^b, g^c, e(g, g)^{abc})$ and $(g, g^a, g^b, g^c, e(g, g)^{\theta})$. An algorithm has non-negligible advantage in solving DBDH if $\Pr[B(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[B(g, g^a, g^b, g^c, e(g, g)^{\theta}) = 1] \geq \epsilon$.

C. ACCESS STRUCTURE

Let $P = \{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^P$ is monotone if $\forall B, C, \text{ we have that if } B \in \mathbb{A} \text{ and } B \subseteq C \text{ then } C \in \mathbb{A}$. An access structure (monotone access structure) is a collection (monotone collection) $\mathbb{A} \subseteq 2^P \setminus \emptyset$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.
Our scheme consists of the following probabilistic polynomial time algorithms:

**Setup**$(\lambda ^*) \rightarrow PP, msk$: This algorithm is run by the Trusted Authority. On input security parameter $\lambda$, it returns the public parameters $PP$ and master secret key $msk$.

**sExtract**$(PP, \{v_i\}_{i \in A_{uid}}, A_{uid}) \rightarrow SK_d$: This algorithm is run by the Trusted Authority. On input the public parameters $PP$, master secret key $msk$ and a user’s signing attribute set $A_{uid}$, it generates signing secret keys $SK_d$ for a user.

**dExtract**$(PP, \{v_i\}_{i \in A_{uid}}, A_{uid}) \rightarrow SK_d$: This algorithm is run by the Trusted Authority. On input the public parameters $PP$, master secret key $msk$ and a user’s decryption attribute set $A_{uid}$, it generates decryption secret keys $SK_d$ for a user.

**Signcrypt**$(m, PP, SK_s, \tilde{A}(M, \rho), \pi_a(M', \rho') \rightarrow \delta)$: This algorithm is run by a Data Owner. On input the original data $m$, public parameters $PP$, predicate $\pi_a$ of data owner and a ciphertext access structure $\tilde{A}$, it outputs the signcrypted data $\delta$.

**PartiallyUnsigncrypt**$(\delta, PP, tk) \rightarrow \delta'$: This algorithm is run by the Cloud Server. On input the ciphertext $\delta$, public parameters $PP$, the transformation key $tk$, and the signcrypted data $\delta$ access policy $\tilde{A}$, it outputs either the transformed ciphertext $\delta'$.

**FullyUnsigncrypt**$(\delta', rk) \rightarrow m$: This algorithm is run by a Data User. On input the transformed ciphertext $\delta'$ and the retrieval key $rk$, it outputs either the original data $m$ or rejection symbol $\bot$.

**UpdateKeyGen**$(a_i, v_i, V_i) \rightarrow \tilde{v}_i, \tilde{V}_i, uk_i$: This algorithm is run by the Trusted Authority. On input the revoked attribute $a_i$, secret attribute key $v_i$ and public attribute key $V_i$, it outputs new secret attribute key $\tilde{v}_i$, public attribute key $\tilde{V}_i$ and update key $uk_i$.

**ProxyUpdate**$(\delta, uk_i) \rightarrow \tilde{\delta}, \tilde{ik}$: This algorithm is run by the Cloud Server. On input the signcrypted message $\delta$ and update key $uk_i$, it outputs new signcrypted message $\tilde{\delta}$ and transformation key $\tilde{ik}$.

1) **SECURITY MODEL**

The notion of security with respect to data confidentiality can be captured by a game. If no probabilistic polynomial time adversary has a non-negligible advantage in winning the following game, it is said to be indistinguishable under chosen plaintext attacks. A challenger $C$ provides the environment for the attack, and $A$ is an adversary.

**Setup**: The challenger $C$ runs the Setup algorithm on input a security parameter $\lambda$, gives public parameters $mpk$ to the adversary $A$ and keeps the master key $msk$ secret.

**Query Phase 1**: The adversary $A$ selects a access structure $\tilde{A}(M, \rho)$, a data owner $uid$ and sends to the challenger $C$.

**Challenge**: The adversary $A$ submits two equal length messages $m_0$ and $m_1$ to the challenger $C$ with signing attribute and access policy. The attribute sets which satisfy the access policy must be not queried in **Query Phase 1**. $C$ picks $b \in \{0, 1\}$ and signcrypt the message $m_b$ by run algorithm Signcrypt. Then return $\delta^*_b$ to $A$.

---

**D. LINEAR SECRET SHARING SCHEME (LSSS)**

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

1. The shares for each party form a vector over $\mathbb{Z}_p^*$.
2. There exists a share-generating matrix $M$ for $\Pi$, where $M$ has $l$ rows and $n$ columns. For all $i = 1, 2, \ldots, l$, let the function $\rho$ defined the party labeling the $i$-th row of $M$ as $\rho(i)$. Consider the vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p^*$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p^*$ are randomly chosen. Then $Mv$ is the vector of $l$ shares of the secret $s$ according to $\Pi$. The share ($Mv$), belongs to party $\rho(i)$. Every LSSS has the property of linear reconstruction defined as follow: Suppose that $\Pi$ is an LSSS for the access structure $A$. Let $S \in A$ be any authorized set, and $I = \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then there exist constants $\{\omega_i\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret $s$ according to $\Pi$, then $\sum_{i \in I} \omega_i \lambda_i = s$. The constants $\{\omega_i\}$ can be found in polynomial time in the size of the share-generating matrix $M$.

---

**FIGURE 1.** Network architecture of our scheme.
TABLE 1. The description of notations to be used in our scheme.

| Notation     | Description                                                                 |
|--------------|-----------------------------------------------------------------------------|
| $K$          | Access structure                                                            |
| $n_d$        | $\text{Claim predicate}$                                                   |
| $H_1, H_2$   | secure hash functions such as $H_1 : \{0, 1\}^{*} \rightarrow G_1$         |
| $PP$         | public parameters                                                           |
| $msk$        | master secret key                                                            |
| $SK_d$       | User secret key for decryption                                              |
| $SK_s$       | User secret key for signing                                                 |
| $A_{uid}$    | $\text{User's attribute set}$                                               |
| $C_{i, i'}$  | Cipher text components                                                      |
| $R_{i, i'}$  | Signature components                                                        |

**Guess:** Finally, the adversary $A$ outputs a guess bit $b' \in \{0, 1\}$ and wins the game if $b = b'$.

The advantage of the adversary $A$ in the above game is defined as $\text{Adv}(A) = |Pr(b = b') - \frac{1}{2}|$.

V. OUR CONSTRUCTION

We give the detailed construction of our revocable attribute-based signcryption scheme with outsourced designcryption in this section. The notations and their semantic meanings utilized in this paper to describe our scheme are presented in Table 1.

**A. SYSTEM INITIALIZATION**

Trusted Authority run the algorithm $\text{Setup}$ to initializes the system.

$\text{Setup}(\lambda^k) \rightarrow PP, msk$. The algorithm takes a security parameter $\lambda$ as input. It choose groups $G_0$ and $G_1$ which are both cyclic groups of prime order $p$. $g$ is the generator of $G_0$. Let $e : G_0 \times G_0 \rightarrow G_1$ be a bilinear pairing. For each attribute $a_i$ in the global attribute set $\mathcal{U}$, this algorithm chooses a random number $v_i \in Z_p^*$ and calculates $V_i = g^{v_i}$. Besides, it picks another secret number $\alpha \in_R Z_p^*$ and calculate $A = e(g, g)\alpha$. And it defines two hash functions: $H_1 : \{0, 1\}^{*} \rightarrow G_1$ and $H_2 : \{0, 1\}^{*} \rightarrow Z_p^*$. It sets the master secret key as $msk = \{(v_i)_{a_i \in \mathcal{U}}, \alpha\}$, and the public parameters as $PP = \{g, G_0, p, A, \{V_i\}_{a_i \in \mathcal{U}}\}$. The Trusted Authority publish the public parameters $PP$, and keep the master secret key $msk$ secret.

**B. USER KEY GENERATION**

The Trusted Authority is in charge of assigns a unique identity $uid$ and an attributes set $Auid$ for each user. In addition the Trusted Authority run the algorithm $\text{dExtract}$ and $\text{sExtract}$ to issues secret keys for each user.

$\text{dExtract}(PP, \{v_i\}_{a_i \in Auid}, uid) \rightarrow SK_d$. The algorithm takes public parameters $PP$, attribute secret key and attributes set $Auid$ of that user as input. It selects random number $r_d, \beta \in Z_p^*$ for a decryption user $uid$. And according to the user’s attribute set $Auid$, it randomly chooses $(r_{d, i} \in Z_p^*)_{a_i \in Auid}$. Then it computes $K_d = \gamma^{r_d}, K_{d,i} = \gamma^{r_{d,i}}H_1(a_i)\gamma^{r_{d,i}/\beta}, K_{d,i}' = H_1(a_i)\gamma^{r_{d,i}/\beta}$. It sets transformation key as $tk = \{(K_{d,i}, K_{d,i}')\}_{a_i \in Auid}$, retrieval key as $rk = \beta$ and the decryption secret key as $SK_d = (K_d, tk, rk)$.

$K_d, rk$ will be send to the user $uid$. And $tk$ will be send to the cloud server. The cloud server adds the tuple $(uid, Auid, tk)$ into the user secret key list $\text{List}_uid$.

$\text{sExtract}(PP, \{v_i\}_{a_i \in Auid}, uid) \rightarrow SK_s$. The algorithm selects random number $r_s \in Z_p^*$ for each signer $uid$. And according to the signer’s attribute set $Auid$, it randomly chooses $(r_{s, i} \in Z_p^*)_{a_i \in Auid}$. Then it computes $K_s = g^{r_s-r_d}, K_{s,i} = g^{r_s}H_1(a_i)^{r_{s,i}}, K_{s,i}' = H_1(a_i)^{r_{s,i}/\beta}$. It sets the signing secret key as $SK_s = (K_s, (K_{s,i}, K_{s,i}'))$. The message exchange process is shown in Figure 2.

**C. UPLOAD DATA**

When one Data Owner wants to share data $m$ with others, it should run the algorithm $\text{Signcrypt}$ to performs the sign-cryption procedure.

$\text{Signcrypt}(m, PP, SK_i, \mathcal{A}(M, \rho), \pi(M', \rho') \rightarrow \delta)$. The algorithm takes the data $m$, public parameters $PP$, signing secret key $SK_i$ of the Data Owner, access structure $\mathcal{A}$, claim-predicate $\pi$ of the Data Owner. For the ciphertext access structure $\mathcal{A}$, the algorithm defines two corresponding $I \times N$ matrix $M$, and function $\rho$ maps each row $x$ of $M$ to an attribute $\rho(x)$. It first chooses a random secret $s \in Z_p^*$ and vector $v = (s, v_2, \ldots, v_N) \in (Z_p^*)^N$, where $v_2, \ldots, v_N$ will be used to share the secret $s$. For $i = 1$ to $I$, it computes $x_i = M_i v$, where $M_i$ denotes the $i$-th row of $M$. Thus, it computes the ciphertext as $C_0 = m \cdot \mathcal{A}^t, C_0' = g^s, C_i = g^{v_i}, C_i' = V_i^N$. Then it picks a random element $t \in Z_p^*$ to computes $V = e(C_0', g^t), h = H_2(m, \mathcal{V}, M, M')$. For the predicate $\pi_s$ of the data owner, it similarly generates $M'$ and computes $\lambda_i' = M_i' V$. Thus, the signature part is defined as $R = g^{V}\cdot K_{d,i}'$, $R_f = K_{d,i}' V_i', R_f' = (K_{d,i}')^{V_i'}$. Finally the Data Owner get output $\delta = (C_0, C_0', \{C_i, C_i'\}, h, R, (R_f, R_f'))$, $(M, M')$ and upload it to Cloud Server. The message exchange process is shown in Figure 3.

**D. ACCESS DATA**

When one data user wants to request data from the cloud server, it must finish the unsigncryption procedure. The outsourced unsigncryption technology is used here to alleviate
the user side’s computation overhead. The unsigncryption procedure consists of two phases.

phase 1: unsigncryption by cloud server

The Cloud Server run the algorithm **PartialUnsigncrypt** to partially unsigncrypts the requested data $\delta$.

**PartialUnsigncrypt**($\delta$, $P$, $tk$) $\rightarrow$ $\delta'$. The algorithm takes the signcrypted data $\delta$ and the transformation key $tk$ of user $uid$ as inputs. The attributes set $A_{uid}$ used in $tk$ satisfies the access structure $A$ in $\delta$, the algorithm proceeds as follows:

Define $I \subset \{1, 2, \ldots, l\}$ as $I = \{i : \rho(i) \in A_{uid}\}$. There must exist a set of reconstruction coefficients $\{\omega_i \in Z_p\}_{i \in I}$ satisfies $\sum_{i \in I} \omega_i \lambda_i = s$. Then, it computes

$$P = \prod_{i \in I} e(C_i, K_{d,i})^{\omega_i} = e(g, g)^{s\lambda / \beta}$$

Define $I' \subset \{1, 2, \ldots, l\}$ as $I' = \{i' : \rho(i') \in A_{uid}\}$. There must exist a set of reconstruction coefficients $\{\omega_{i'} \in Z_p\}_{i' \in I'}$ satisfies $\sum_{i' \in I'} \omega_{i'} \lambda_{i'} = s$. Then, it computes

$$S = \prod_{i' \in I'} e(V_{i'}, R_{d,i'})^{\omega_{i'}} = e(g, g)^{s\beta s}$$

The algorithm outputs a partially unsigncrypted data $\delta' = (P, S)$. Then the cloud server send the partially unsigncrypted data $\delta'$ to the data user $uid$.

phase 2: unsigncryption by data user

Once receiving partially unsigncrypted data from cloud server, the user $uid$ run the algorithm **FullyUnsigncrypt** to further unsigncrypts the partially unsigncrypted data $\delta'$. The algorithm takes the partially unsigncrypted data $\delta'$ and the retrieval key $rk$ as inputs. It recover the data $m$ and verify the signature as follows:

$$Y = e(K_d, C_0') \cdot P^\beta$$

$$m = \frac{C_0}{Y}$$

$$\mid V' = \frac{e(C_0', R)}{(Y \cdot S^{-1})^\beta}$$

Eventually, if $h = H_2(m, V', T_d, T_j)$ holds, it outputs $m$. Otherwise it outputs $\perp$. The message exchange process is shown in Figure 4.

**FIGURE 4.** Access data.

### E. ATTRIBUTE REVOCATION

To revoke an attribute $a_i$ from some users, Trusted Authority adds these users’ identity into attributes revocation list $RL_{Listi}$. Then the revocation procedure consists of two phases.

**phase 1:** key update by Trusted Authority

The Trusted Authority run the algorithm **UpdateKeyGen** to generate update key and send it to the Cloud Server for ciphertext updating.

**UpdateKeyGen**(a_i, v_i, V_i) $\rightarrow$ $\tilde{v}_i$, $\tilde{V}_i$, $uk_i$. The algorithm takes the revoked attribute $a_i$, secret attribute key $v_i$ and public attribute key $V_i$ as input. It chooses a new random element $\tilde{v}_i \in Z_p^*$ and computes $\tilde{V}_i = g^{\tilde{v}_i}$ as new attribute secret and public key for revoked attributes. Then it generates the update key $uk_i = \frac{\tilde{v}_i}{v_i}$ and send to the Cloud Service.

**phase 2:** signcrypted data and key update by Cloud Server

When the Cloud Server receives the update key $uk_i$, it run the algorithm **ProxyUpdate** to updates the signcrypted data and user secret key list $List_{tk}$.

**ProxyUpdate**(\delta, $uk_i$) $\rightarrow$ $\tilde{\delta}$, $\tilde{tk}$. The algorithm takes the signcrypted data $\delta$, update key $uk_i$ as input. It manipulates signcrypted data as:

$$\rho(i) \neq a_i : \tilde{C}_i = C_i', \tilde{R}_i = R_i'$$

$$\rho(i) = a_i : \tilde{C}_i = (C_i')^{uk_i}, \tilde{R}_i = (R_i')^{uk_i}$$

And it compute new user secret key of user $uid \neq RL_{Listi}$ in $tk$ as:

$$\rho(i) \neq a_i : \tilde{K}_{d,i} = K_{d,i}, \tilde{K}_{s,i} = K_{s}(i)$$

$$\rho(i) = a_i : \tilde{K}_{d,i} = K_{d,i}^{uk_i}, \tilde{K}_{s,i} = K_{s}(i)^{uk_i}$$

The message exchange process is shown in Figure 5.

**FIGURE 5.** Attribute revocation.

### VI. SECURITY ANALYSIS

#### A. CORRECTNESS

The correctness of decryption part in unsigncryption procedure:

$$m = \frac{C_0}{e(K_d, C_0')} \cdot P^\beta$$

$$= \frac{m \cdot (e(g, g)^{\rho a})^{\alpha s}}{e(g, g)^{(\alpha - \beta)l \omega}} \cdot \frac{1}{\prod_{i \in I} e(C_i, D_{d,i})^{\omega_i}}$$

$$= \frac{m \cdot e(g, g)^{\rho a}}{e(g, g)^{(\alpha - \beta)l \omega}} \cdot e(g, g)^{-\beta l}$$

$$= \frac{m \cdot e(g, g)^{\rho a}}{e(g, g)^{(\alpha - \beta)l \omega} + \beta l} = m.$$
The correctness of verification part in unsigncryption procedure:

\[
V' = \frac{e(C'_0, R)}{(Y \cdot S^{-1})^h} = \frac{e(g, g)^{a-r_sh} \prod_{i \in I} e(V_i, R'_i)^{a_i h}}{e(g, g)^{a-y_h} \prod_{i \in I} e(R_i, g)^{a_o y_h}} = \frac{e(g, g)^{a-y_h} e(g, g)^{r_sh}}{e(g, g)^{a-y_h}} = e(g, g)^{y_h} = V.
\]

If the data is not forged or falsified, then the equation \( h = H_2(m, V', T_d, T_t) \) holds.

\[\text{B. SECURITY PROOF}\]

1) CONFIDENTIALITY

The proposed scheme is IND-(CP-ABSC)-CPA secure under the DBDH assumption.

Theorem 1: Suppose the DBDH assumption holds in \( G_0 \) and \( G_T \). Then no polynomial time adversary can win the IND-(CP-ABSC)-CPA game with a non-negligible advantage.

Proof of Theorem 1: If an adversary \( A \) can break the IND-(CP-ABSC)-CPA secure of our scheme that makes \( q \) times queries with a non-negligible advantage \( \epsilon \), then there is another challenger \( C \) with algorithm \( B \) can solve the DBDH problem with a non-negligible disadvantage. The simulation proceeds as follows.

**Setup:** The adversary \( A \) selects a access structure \( \hat{A}(M, \rho) \), a data owner \( uid \) and sends to the challenger \( C \). The challenger \( C \) is given an instance of the DBDH tuple \((g, g^a, g^b, g^c, Z)\), where \( g \) is a random generator of \( G \) and \( a, b, c \in \mathbb{Z}_p^* \). \( C \) performs the algorithm **Setup** in the proposed scheme.

In particular, it chooses a random element \( x' \in \mathbb{Z}_p^* \) and sets \( A = e(g, g)^{y_h} = e(g, g)^{r_sh} \). \( C \) publish the public parameters to \( A \) and keep master secret key secret.

**Query Phase 1:** The adversary \( A \) selects attributes set \( \{a_i : a_i \notin \hat{A}\} \) and queries the challenger \( C \) for decryption secret key. The challenger \( C \) randomly chooses \( r' \in \mathbb{Z}_p^* \) to compute \( K_d = g^{x'-r'} \). For the \( x' = \alpha - ab \), \( r_d = ab + r' \). Then according to the algorithm **dExtract**. The decryption secret key sent to \( A \) is \((K_d = g^{x'-r'}, K_{d,i} = g^{(ab+r')/\beta} H_1(a_i)^{y_{d,i}/\beta}, K_{d,i} = H_1(a_i)^{y_{d,i}/\beta}, \beta)\).

**Challenge:** The adversary \( A \) submits two equal length messages \( m_0 \) and \( m_1 \) to the challenger \( C \). \( C \) select \( b \in \mathbb{Z}_p[0, 1] \) and signcrypt the message \( m_b \) as:

\[
s = c, C_0 = m_b e(g, g)^{a-r_sh} = m_b Z e(g, g)^{x'c},
\]

\[
[C_i, C_i']_{i \in I}, h, R, [R'_i, R'_i]_{i \in I}.
\]

The challenger \( C \) sends the generates ciphertext to the adversary \( A \). **Query Phase 2:** The adversary \( A \) makes a polynomially bounded number of above queries in the same way in **Query Phase 1** except that he cannot make **dExtract** queries on attribute sets which satisfy the ciphertext-policy \( \hat{A}(M, \rho) \).

**Guess:** The adversary \( A \) outputs a guess bit \( b' \in \{0, 1\} \). If \( b = b' \), algorithm \( B \) outputs 1 meaning \( Z = e(g, g)^{y_h} \). Otherwise it outputs 0 meaning \( Z = e(g, g)^{b'} \).

Thus, if \( |Pr[B(g, g^a, g^b, g^c, e(g, g)^{y_h}) = 1] - Pr[B(g, g^a, g^b, g^c, e(g, g)^{b'}) = 1]| \geq \epsilon \) holds, which means \( B \) solves the DBDH problem with \( \epsilon \) advantage.

2) COLLUSION ATTACK RESISTANCE

In our scheme, secret attribute keys owned by different users are independent. Thus this scheme can defend against collusion attacks although the original ABE can not. For example, assuming that colluding attackers all do not possess a sufficient number of attributes to successfully decrypt the ciphertext \( C \) alone. It is impossible to successfully collusion attack for two reasons. First, private keys possessed by users with different identity are independent since they are randomized by unique element \( r_d \). Second, cloud server can not compute the partially decrypted data by using different users’ public attribute key, because the public attribute keys are also blind by unique random elements. Thus, when users and the cloud server collude, they are still unable to combine their private key components in any useful way. Therefore we draw the conclusion that the proposed scheme is secure against collusion attacks.

3) UNFORGEABILITY

The adversary who wants to forge the signcryption ciphertext must have the secret key of sensitive data owner or know the random factor \( t \). However, the adversary cannot forge the private key \( SK_s = (g^{a-r_sh}, \{g^a H_1(a_i)^{y_i}, H_1(a_i)^{y_i, r_i}\}) \) because the secret element \( r_s \) and \( r_{s,i} \) are chosen randomly. Additionally, the attacker cannot falsify a valid signcryption ciphertext from exist ciphertexts. Even if the ciphertext is modified by the adversary, the receiver can still verify the integrity and legitimacy of the ciphertext through the algorithm FullyUnsigncrypt. Therefore we draw the conclusion that the proposed scheme is unforgeable under chosen message attacks.

**VII. PERFORMANCE ANALYSIS**

In this section, we analyze the communication and computation cost of our proposed scheme. Then we compare them with combining ABE and ABS scheme, and other previous ABSC schemes. The communication overhead is directly related to the ciphertext size, decryption key size and signing key size. And the computation overhead is directly related to the cost of computation operation in signcryption and designcryption phase. To make the analysis more understandable, the notations which we use to evaluate the efficiency of different schemes are listed in Table 2. We assume that the size of access structure and hash computation cost are not included in the calculation.

**A. COMMUNICATION OVERHEAD**

We analyze the ciphertext, decryption key and signing key size of our proposed scheme as follows. According to the
construction, the length of ciphertext $\delta$ can be computed as:

$$|C_0| + |C'_0| + n_{e}(|C_1| + |C'_1|) + |h| + |R| + n_{s}(|R_t| + |R'_t|)$$

(1)

Therefore the total ciphertext size according to equation (1) is $(2n_e + 2n_t + 2|g_0| + |g_1| + |Z_p|)$. In the same way, the decryption key and signing key is $SK_d, SK_s$ can be computed as:

$$|K_d| + n_d(|K_{d_{0,l}}| + |K'_{d_{0,l}}|) + |r_k|$$

$$|K_s| + n_s(|K_{s_{0,l}}| + |K'_{s_{0,l}}|)$$

(2)

(3)

The decryption key and signing key size according to equation (2),(3) is $(2n_d + 1)|g_0| + |Z_p|, (2n_s + 1)|g_0|$.

### B. COMPUTATIONAL OVERHEAD

The major concern for the computation cost is the exponentiation and pairing operations in signcryption and decryption algorithm. In our scheme, the Signcryption algorithm calculates $mA^e, g^e, \{g^{s_{0}}, V_{0}^{-1}\}, g^{rK_0}, \{K_{s_{0}}, K_{s_{0}'}, \}, e(C_0, g^e)$, which need $(2n_e + 2n_t + 3)$ times exponentiation operations in $G_0, 2$ times exponentiation operations in $G_T$ and one pair operation. And the FullyDesigncryption algorithm calculates $Y = e(K_{d_{0,l}}, C'_0 \cdot P^b), V' = e(C'_0, R)/(Y \cdot S^{-1})^h$, which only need one exponentiation operation in $G_0$ and one exponentiation operation in $G_T$.

### C. COMPARISON

In this section, we compare the performance of our scheme with [6], [35], [27], [28], [31], [34] in terms of ciphertext size, decryption key size, signing key size, signcryption and unsigncryption computations.

The computational overheads of the schemes in [6], [35], [27], [28], [31], [34] are analyzed as follows. In [6], [35], the decryption and signing key composed $n_d$ and $(2n_a + 1)$ elements in $G_0$. And the ciphertext contained $(3n_a + 2n_t + 1)$ elements in $G_0$, an element in $G_T$. We can see that this size is a bit larger than ours. In [27], the decryption key, signing key and ciphertext are related to attributes, which composed $(2n_a + 1)$ elements in $G_0, (2n_s + 1)$ elements in $G_0$ and $(2n_a + 2n_s + 2)$ elements in $G_0$ plus an element in $G_T$ and an element in $Z_p$ respectively. Our's key and ciphertext size is almost same with them. In [28], the decryption key, signing key and ciphertext are related to attributes, which composed $(4n_s + 1)$ elements in $G_0, (4n_s + 1)$ elements in $G_0$ and $(3l_e + 4)$ elements in $G_0$ plus an element in $G_T$ respectively. Compared to our scheme, we have shorter decryption and signing key size and a little longer ciphertext size. In [31], the decryption key, signing key and ciphertext are related to attributes, which composed $(n_a + 2)$ elements in $G_0, (n_s + 2)$ elements in $G_0$ and $(n_a + n_s + 4)$ elements in $G_0$ plus an element in $G_T$ respectively. In [34], the decryption key, signing key and ciphertext are related to attributes, which composed $(n_d + 2)$ elements in $G_0, (n_e + 2)$ elements in $G_0$ and $2(n_a + n_s + 2)$ elements in $G_0$ plus an element in $G_T$ respectively. The decryption and signing key size in [34] is shorter than ours, while the ciphertext size is larger than ours. The Comparison of ciphertext, decryption key and signing key size are summarized in Table 3.

The computational overheads of the schemes in [6], [35], [27], [28], [31], [34] are analyzed as follows. In [6], [35], the encryption and decryption algorithms need $(3n_a + 1)$ exponentiations in $G_0$ plus one exponentiation in $G_T$ and $2n_a$ pairings plus $n_d$ exponentiations in $G_T$, respectively. Adding the signing and verifying algorithm to it, there are $(2n_a + 1)$ exponentiations in $G_0$ plus one exponentiations in $G_T$ and $(2n_s \log n_s + 2)$ pairings, respectively. Thus we can claim that our scheme is superior over the [6], [35] whose overall of signcryption and designcryption cost is $(3n_a + 1)E_0 + 2E_T$ and $(2n_d + 2n_a \log n_s + 2) \cdot P_T + n_d E_T$, respectively. In [27], The signcryption and designcryption algorithms take $(2n_s + 2n_a + 3)$ exponentiations in $G_0$ plus one exponentiations in $G_T$ plus one pairing and $(1 + n_s \log n_s) + \log n_s)$ exponentiations in $G_T$ plus $(2n_d \log n_a + 2n_e \log n_s + 1)$ pairings. Our scheme’s signcryption cost is close to [27]. But the designcryption cost is much better than theirs due to the outsourced designcryption. In [28], The signcryption and designcryption algorithms take $(2n_d + 2n_a + 3)$ exponentiations in $G_0$ plus one exponentiations in $G_T$ plus $(2n_d + 2n_s \log n_s + 1)$ pairings. The signcryption cost in [28] is also close to ours. In [34], signcryption need $(n_s + 2n_a + 4)$ exponentiations in $G_0$, which is less than ours. And their designcryption need $(2 + l_e)$ exponentiations in $G_0$ plus one pairing, which is greatly less than the above scheme but still more than ours. For the computation overhead of designcryption, the computational overhead in our scheme is quite light for users. The reason is that we securely outsourced the most complex operations to a proxy cloud server and only left user’s devices a exponentiation in group $G_0, G_T$, respectively. In [6], [35], [27], [28], [31], the number of exponentiation and pairing operations of designcryption is at least linear to the number of attributes in decryption(signing) key. Although [34] also outsourcing the designcryption to a cloud server to alleviate computation overhead on user side, our scheme reduce $n_s + 1$ exponentiation and a pairing operations compared to theirs. The Comparison of signcryption cost and designcryption cost are listed in Table 4.

We conducted a simulation experiment on an Intel i3-3320 processor with 3.30GHz and 2GB memory running 32-bit ubuntu 16.04.4 system. The experiment was based on Stanford Pairing-Based Crypto (PBC) library. We adopted

| Notation | Description |
|----------|-------------|
| $g_0$ | the length of each element in $g_0$ |
| $g_1$ | the length of each element in $g_1$ |
| $Z_p$ | the length of each element in $Z_p$ |
| $n_e$ | number of attributes required for encryption |
| $n_s$ | number of attributes required for signing |
| $n_d$ | number of attributes decryption key contained |
| $n_k$ | number of attributes signing key contained |
| $E_0$ | the time of one exponentiation operation in the group $G_0$ |
| $E_T$ | the time of one exponentiation operation in the group $G_T$ |
| $P$ | the time of one pairing operation |
TABLE 3. Comparison of ciphertext size and user private key size.

| Scheme          | Ciphertext Size | Decryption Key Size | Signing Key Size |
|-----------------|-----------------|---------------------|-----------------|
| Lewko et al. [6], Cao et al. [35] | $(3n_e + 2n_s + 1)|g_0| + |g_T| | $n_d|g_0| | $(2n_s + 1)|g_0| |
| Wang et al. [27] | $(2n_e + 2n_s + |g_0| + |g_T| | $n_d|g_0| | $(2n_s + 1)|g_0| |
| Emura et al. [28] | $(3n_e + 4)|g_0| + |g_T| | $n_d|g_0| | $(4n_d + 1)|g_0| |
| Rao et al. [31] | $(n_e + n_s + 4)|g_0| + |g_T| | $n_d|g_0| | $(n_s + 2)|g_0| |
| Deng et al. [34] | $2(n_e + n_s + 2)|g_0| + |g_T| | $n_d|g_0| | $(n_s + 2)|g_0| |
| Ours            | $(2n_e + 2n_s + 2)|g_0| + |g_T| | $n_d|g_0| | $(2n_s + 1)|g_0| |

TABLE 4. Comparison of ciphertext size and user private key size.

| Scheme          | Signcryption Cost | Unsigncryption Cost |
|-----------------|-------------------|---------------------|
| Lewko et al. [6], Cao et al. [35] | $(3n_e + 2n_s + 1)E_0 + 2ET$ | $(2n_d + 2n_s + 3)E_0 + 6ET + P$ |
| Wang et al. [27] | $(2n_e + 2n_s + 3)E_0 + ET + P$ | $(1 + n_d log n_d + n_s log n_s)ET + (2n_d log n_d + 2n_s log n_s + 1)P + (2n_d + 4n_s + 3)P$ |
| Emura et al. [28] | $(2n_d + 2n_s + 3)E_0 + ET$ | $(2n_d + n_s + 2)E_0 + (n_s + 5)P + (2 + n_s)E_0 + ET$ |
| Rao et al. [31] | $(2n_e + 4n_s + 6)E_0 + ET$ | $2n_d|E_0 + ET$ |
| Deng et al. [34] | $(n_e + n_s + 4)E_0$ | $2n_d|E_0 + ET$ |
| Ours            | $(2n_e + 2n_s + 3)E_0 + 2ET + P$ | $2n_d|E_0 + ET$ |

FIGURE 6. Computational time of signcryption.

FIGURE 7. Computational time of unsigncryption.

the type A bilinear pair as the bilinear map. Under this environment, we obtained the results that a pairing operation takes 10.6ms; an exponentiation operation in $G_0$ and $G_T$ takes 6.4ms and 5.7ms, respectively.

Figure 6 depicts the computation time of signcryption on the data owner side of our and the other schemes. It shows that the time for signcryption increases linearly with the number of attributes involved. Our scheme has a relatively good performance. Our scheme’s computational cost of signcryption is lower than the schemes [6], [35] and [27] and almost same as the scheme [27], [28]. Only Deng et al.’s [34] performs better than ours. However, as expected, the experiment results demonstrate that our scheme largely decreases the designcryption overhead. From Figure 7, the computational cost of the data user keeps constant and is much lower than the other schemes [6], [35], [27], [28], [31], [34], due to the outsourcing mechanism is used in our designcryption algorithm.

Except the above comparison, our scheme supports a more important function. To handle the attribute revocation problem, the previous schemes [6], [35], [27], [28], [31], [34] require the data owner to decrypt the ciphertext and signcryption it again. However in our scheme, That task of data owners is outsourced to the cloud servers while the curious cloud servers do not obtain the plaintext of storage data. And our scheme is forward security for the user who drops an attribute fail to access the plaintext of the subsequent data after attribute revocation.

Through the comparison, although the ciphertext and key size of our proposed scheme have little advantage over above schemes, our proposed scheme outperforms in computation overhead and supports more feature which makes it more useful in practice.

VIII. CONCLUSION

In this paper, we present a new ciphertext-policy attribute-based signcryption scheme for data sharing in the cloud computing environment. It provide flexible access control and ensure the authenticity of the data without revealing the specific identity of the data owner. Besides, it can efficiently handle attributes revocation. The security analysis prove that our scheme can successfully guarantee the data communication security in cloud computing environment. For our scheme support outsourced unsigncryption, the performance analysis demonstrates that our scheme is efficient in terms of computational cost. In the future research, we aim to design a more efficient signcryption approach with less communication and computation requirement during the revocation phase.

ACKNOWLEDGMENT

This article was presented in part at the International Conference on Wireless Algorithms, Systems, and Applications 2019 [1].
REFERENCES

[1] N. Deng, S. Deng, C. Hu, and K. Lei, “An efficient revocable attribute-based signcryption scheme with outsourced designcryption in cloud computing,” in Proc. 14th Int. Conf. Wireless Algorithms, Syst., Appl. (WASA), Springer, 2019, pp. 84–97.

[2] J. Li, J. Xie, Y. Zhang, and J. Han, “KS-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage,” IEEE Trans. Services Comput., vol. 10, no. 5, pp. 715–725, Sep. 2017.

[3] 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. 2017, doi: 10.1109/tdsc.2015.2499755.

[4] C. Zuo, J. Shao, J. K. Liu, G. Wei, and Y. Ling, “Fine-grained two-factor protection mechanism for data sharing in cloud storage,” IEEE Trans. Inf. Forensics Security, vol. 13, no. 1, pp. 186–196, Jan. 2018.

NINGZHI DENG received the bachelor’s degree from the School of Computer Science, Chongqing University, Chongqing, China, in 2015, where he is currently pursuing the master’s degree with the School of Computer Science. His main research interest is information security.

SHAOJIANG DENG received the Ph.D. degree from the College of Computer Science, Chongqing University, China, in 2005. He is currently a Professor with the College of Applied Computer Science, Chongqing University. In 2007, he was a Visiting Scholar with the Institute of Applied Computer Science, Dresden University of Technology, Germany. His research interests focus on digital chaotic cryptography, information security, and hash function construction.
CHUNQIANG HU (Member, IEEE) received the B.S. degree in computer science and technology from Southwest University, Chongqing, China, in 2006, the M.S. and Ph.D. degrees in computer science and technology from Chongqing University, in 2009 and 2013, respectively, and the second Ph.D. degree in computer science from The George Washington University, Washington, DC, USA, in 2016. He was a Visiting Scholar with The George Washington University, from 2011 to 2012. He is currently a Faculty Member of the School of Big Data and Software Engineering, Chongqing University. His research interests include privacy-aware computing, big data security and privacy, wireless and mobile security, applied cryptography, and algorithm design and analysis. He was honored with the Hundred-Talent Program by Chongqing University.

KAIWEN LEI received the bachelor’s degree from the School of Computer Science, Tianjin University of Science and Technology, Tianjin, China, in 2017. She is currently pursuing the master’s degree with the School of Computer Science, Chongqing University. Her main research interest is information security.