An Efficient Revocable and Searchable MA-ABE Scheme with Blockchain assistance for C-IoT

Jiguo Yu, *Fellow, IEEE*, Suhui Liu, *Student Member, IEEE*, Minghui Xu, *Member, IEEE*, Hechuan Guo, Fangtian Zhong, Wei Cheng, *Senior Member, IEEE*

***Abstract*—Internet of Things (IoT) devices usually stores data on clouds for computational overhead offloading and easy data sharing. The data owners, as a result, usually have concerns about the security and privacy of their data stored in such cloud-assisted IoT (C-IoT) systems. Traditional encryption and search primitives, including Attribute-Based Encryption (ABE) and Public-key Encryption with Keyword Search (PEKS), however, suffer from high overheads in decryption and revocation, and privacy leakage in search. To address these issues, we propose an efficient revocable and searchable multi-authority ABE (MA-ABE) scheme named ERS-ABE, which utilizes blockchain technology to implement keyword-based search and dynamic user management. ERS-ABE also adopts cloud-assisted decryption to improve the efficiency of IoT devices. It has been proven to be secure against the selective replayable chosen-ciphertext attacks and the chosen-keyword attacks under the random oracle model. The feasibility and efficiency of ERS-ABE have been evaluated through theoretical analysis and extensive simulation studies. The results indicate that EAR-ABE performs better over the state-of-the-art in both storage and computational overheads. Particularly, the operations that are usually done by a central server but taken by a blockchain in EAR-ABE cost only a few seconds.**

***Index Terms*—Blockchain, Cloud-assisted IoT, Attribute-based Encryption, Public Key Encryption with Keyword Search, Revocation.**

I. Introduction

INTERNET of Things (IoT) applications are providing services in the fields of smart homes, smart factories, smart health, smart cities, etc. To efficiently manage and process IoT data, the tasks of storage, computation, and management are usually conducted by cloud service providers because of the resource constraints of IoT devices [1]. The Cloud-assisted IoT (C-IoT), however, is questioned due to the ever-growing security and privacy concerns regarding the trust of cloud servers. Storing encrypted data and enforcing access control is an obvious solution, however, performing searches over ciphertext is difficult to realize without leaking information to the search server.

Attribute-Based Encryption (ABE) can realize data confidentiality and one-to-many fine-grained access control within one primitive, thus it is treated as the most promising method to protect cloud-stored data. Existing ABE schemes can be divided into two types based on encryption policy. In Key-policy ABE (KP-ABE), attributes are used to annotate the ciphertexts. The formulas over these attributes are ascribed to users' secret keys. While in Ciphertext-policy ABE (CP-ABE), attributes are used to represent users' credentials. The formulas over these credentials are attached to the ciphertext by the encrypting party [2]. Compared with KP-ABE, CP-ABE is more applicable for the C-IoT applications as a data owner can directly describe who can decrypt its ciphertext.

However, implementing CP-ABE in C-IoT systems still faces the following challenges. (1) A single attribute authority cannot efficiently and securely handle all attributes from a large number of interconnected IoT devices. (2) The resource-limited IoT devices cannot afford ABE's resource-consuming decryption tasks, especially for some real-time IoT scenarios. (3) Users' attributes change dynamically in the real world and thus a practical ABE scheme has to support user revocation and attribute update. (4) An efficient and secure search method is needed as sharing is the only way to gain value from IoT data while downloading all ciphertexts and decrypting them locally for a search is impossible for resource-limited IoT devices.

Several researchers have proposed searchable ABE schemes, where the search power is constrained in a fine-grained way. Nevertheless, access-before-search not only leads to low efficiency but also falls short of realistic requirements. In other words, most IoT data sharing applications gain profit from sharing thus not restricting the search authority and causing the user's desire to access is the most commonly used method. Unfortunately, for now, there are few papers that focus on this problem. Furthermore, one fatal risk caused by centralized keyword-based searches that controlled the cloud server is that the cloud server may gain the privacy information of the user by pattern analysis and link algorithm.

To address the abovementioned issues, this paper presents an efficient revocable Multi-Authority Attribute-Based Encryption (ERS-ABE) scheme with a blockchain-enabled coarse-grained keyword-based search feature for C-IoT. The main contributions of the paper are summarized as follows:

J. Yu is with the Big Data Institute, Qilu University of Technology, Jinan, Shandong, 250353, P.R. China; Shandong Fundamental Research Center for Computer Science, Jinan, Shandong, 250353, P.R. China. E-mail: jiguoyu@sina.com

S. Liu (the corresponding author) is with School of Cyber Science and Engineering, Southeast University, Nanjing, 211102, Jiangsu, China. E-mail: suhuiliu@seu.edu.cn

M. Xu and H. Guo are with School of Computer Science and Technology, Shandong University, Qingdao, 266237, P.R. China. Email: mhxu@sdu.edu.cn, ghc@mail.sdu.edu.cn.

F. Zhong is with School of Information Science and Technology, The Pennsylvania State University, University Park, Email: zjf5176@psu.edu

W. Cheng is with Tacoma School of Engineering and Technology, University of Washington, Email: uwcheng@uw.edu

This work was partially supported by National Key R&D Program of China with grant No. 2019YFB2102600, and the NSF of China under Grants 61832012, 61771289, 61672321, and 61373027.

- A new primitive that combines MA-ABE and PEKS (Public key Encryption with Keyword Search) is built to achieve one-to-many fine-grained access control and keyword-based coarse-grained search. We also prove that our blockchain-assisted ABE scheme is secure against the replayable chosen-ciphertext attacks and the chosen-keyword attacks under the random oracle model.
- The ERS-ABE adopts blockchain to replace the central KGC in traditional ABE to generate the public parameters, support dynamic user revocation, manage search permissions, and perform the keyword-based search. Instead of using a central server to provide the search service, the decentralized verifiers utilize consensus to ensure that no single node can acquire extra information from the search requests or perform an unauthorized search.
- For IoT applications, the ERS-ABE adopts multiple attribute authorities to manage a large number of attributes. Besides, it enormously reduces the computational overhead of decryption and update of resource-limited IoT devices by offloading the tasks to the cloud without privacy leakage.

The rest of this paper is organized as follows: Section II summarizes the related works. The related mathematical preliminaries are presented in Section III. Section IV illustrates the system model and the security model. The design of our proposed scheme is detailed in Section V. Section VI and Section VII analyze the security property and evaluate the performance of the proposed scheme, respectively. Finally, the paper is concluded in Section VIII.

II. Related Works

This section summarizes the related works along the lines of outsourced ABE, MA-ABE, dynamic user and attributes revocation, secure cloud-based search, and blockchain-based search.

An ABE scheme is computationally expensive due to its bilinearity property of pairing-based operations. In order to apply ABE to IoT, several studies were carried out to outsource ABE's computational tasks to clouds. Green et al. [3] outsourced the heavy decryption tasks to a cloud server. In [4], Zuo et al. designed an outsourced CP-ABE scheme supporting only "AND" structure. Liu et al. [5] presented an MA-ABE scheme, which not only achieves outsourced decryption but also realizes user traceability by a white-box algorithm. A lightweight attribute-based signcryption (ABSC) scheme was proposed by Yu et al. [6] to perform outsourced decryption by adopting a fog-cloud-assisted architecture. Subsequently, Liu et al. [7] designed an outsourced ABSC scheme for wireless body area networks, which utilizes online/offline encryption to save the power resource of mobile devices. For edge intelligent Internet of Vehicles (IoV), Feng et al. [8] designed an ABE scheme with a parallel outsourced decryption method, which can be applied to any ABE scheme with a tree structure. Chen et al. [9] designed an efficient CP-ABE, where semi-authorized users can work together to obtain the messages. Wang et al. [10] presented an efficient CP-ABE with constant-sized keys and ciphertexts regardless of the number of attributes.

In most cases, relying on one attribute authority to manage all attributes in an IoT system is unrealistic and ineffective. Lewko et al.'s MA-ABE scheme [11] utilized multiple attribute authorities to control multiple independent attribute sets separately. Rahulamathavan et al. [12] designed a decentralized KP-ABE scheme that preserves a user's privacy from multiple authorities. Rouselakis and Waters [13] developed a statically-secure MA-ABE with a large universe, where any string can be transformed as an attribute. Belguith et al. [14] proposed an MA-ABE with cloud pre-decryption and a hidden policy without considering the dynamic revocation of attributes. Similarly, Liu et al. [15] designed an outsourced MA-ABE scheme with white-box traceability. Recently, Datta et al. [16] presented a decentralized MA-ABE for a nontrivial class of access policies.

Dynamic user revocation and attribute update are the practical issues that need to be addressed. Fan et al. [17] proposed an ABE scheme that aims at dynamic membership management with arbitrary states. Imai et al. [18] classified the revocation mechanisms into direct and indirect. In the direct type, each data owner maintains a revocation list during the encryption process. The list is used to grant the revocation right to the encryptors directly [19]. In indirect revocation, authorized authorities are responsible for regularly issuing update keys to unrevoked users. This mechanism requires no prior knowledge of revocation lists. As a result, it greatly reduces the data owner's responsibility. In [20], Cui et al. proposed a server-aided revocable ABE scheme, where data users cannot generate the complete decryption key without the update key issued by a trusted revocation server. Zhong et al. [21] designed an MA-ABE with a hiding policy, which adopts the direct revocation mechanism by letting the data owner insert a revocation list in the ciphertext during encryption. Ge et al. [22] presented a revocable ABE with integrity verification, where a cloud cannot deceive data owners.

To address the security concerns in cloud-based search, Boneh et al. [26] proposed the public key encryption with keyword search (PEKS). Zheng et al. [27] presented a novel cryptographic solution by combining the keyword search with ABE technology, which is named verifiable attribute-based keyword search (VABKS). Liang et al. [28] designed a searchable key-policy attribute-based proxy re-encryption system, which accomplishes fine-grained both access and search. In [29], Li et al. presented a novel cryptographic primitive to provide a fine-grained keyword search. Similarly, Wang et al.'s scheme [24] combines ABE and PEKS to achieve efficient data search and access control, overlooking the privacy concerns on the access policy. An anti-quantum scheme was studied for cloud Industrial IoT (IIoT) by utilizing the lattice-based technology to secure the PEKS scheme [30]. All these searchable schemes share one common drawback: searchers or data users must trust the central search server.

To avoid the security concerns caused by a central search server, researchers studied several blockchain-based search schemes. Blockchain, which originated from the Bitcoin [31], is a tamper-proof distributed ledger technology. Its integration with C-IoT has been extensively studied [32]–[34] to enhance IoT security [35] and to extend trust from on-chain to off-

TABLE I
FUNCTION COMPARISON.

Scheme	Access Structure	MA	Out-Dec	User Revocation	Attrib Update	Searchable	Security
[4]	AND	×	✓	×	×	×	RCCA
[14]	LSSS	✓	✓	×	×	×	Selective RCPA
[15]	LSSS	✓	✓	×	×	×	Selective RCCA
[23]	LSSS	×	✓	✓	−	×	Selective RCCA
[24]	LSSS	×	✓	✓	−	Coarse-grained	Selective CPA
[25]	LSSS	×	✓	BC-assisted	×	BC-assisted Fine-grained	Selective CPA
ERS-ABE	LSSS	✓	✓	BC-assisted	Partially Re-Enc	BC-assisted Coarse-grained	Selective RCCA

Abbreviations: MA: Multi-authority. AND: 'AND' gate only. LSSS: Linear Secret Sharing Schemes. ✓: Accomplished. ×: Not accomplished. RCCA: Replayable chosen-ciphertext attacks. CPA: Chosen-plaintext attacks. -: Not Considered.

TABLE II
NOTATIONS.

Notation	Meaning
$\mathbb{G}$,$\mathbb{G}_T$	Two multiplicative cyclic groups of prime order p.
$\mathbb{Z}_p$ ($\mathbb{Z}_p^*$)	The integer group modulo p (without the zero element).
Ψ	An access structure $\Psi = (\mathbb{M}, \rho)$
GID	The global id of a user
AA_j	An attribute authority
I_w	Keyword indexes
T_w	The trapdoor of keyword w
S_Ψ	The attribute set related to Ψ
S_{GID}	A set of attributes related to user GID
S_{AA_j}	The attribute set related to AA_j
S_w	A set of keywords related to file Fl
S_{AA}	The universe set of attribute authorities
S'_{AA}	The set of corrupted attribute authorities
$S_{\mathrm{A}}/S_{\mathrm{R}}$	The universe/revoked set of attributes
PP	The public parameters
$\mathrm{PK}_{\mathrm{AA}_j}/\mathrm{SK}_{\mathrm{AA}_j}$	The public/ private key related to AA_j
$\mathrm{IPK}_{\mathrm{GID}}/\mathrm{ISK}_{\mathrm{GID}}$	The public/private id key related to user GID
$\mathrm{AK}_{\mathrm{GID}}$	Authorization key related to user GID
$\mathrm{SK}_{\mathrm{GID}}$	Decryption key
OPK/OSK	The public/private outsourced decryption key
AUK/CUK	The attribute/ciphertext update key

chain [36]. Guo et al. [23] proposed an online/offline revocable CP-ABE scheme to support cloud-assisted decryption and blockchain-assisted search for the Internet of Medical Things (IoMT) ecosystem. Zhang et al. [37] adopted blockchain to record every keyword request made by a user to resist online keyword guessing attacks (KGA) for PEKS. Guan et al. [38] integrated permissioned blockchain and searchable encryption to protect sensitive data in e-commerce. Liu et al. [25] designed a distributed searchable ABE with blockchain-aided search authorization and fine-grained access. Li et al. [39] proposed a blockchain-assisted electronic certificate sharing system that utilizes searchable ABE to achieve fine-grained data access. Niu et al. [40] combined the attribute-based searchable encryption (ABSE) with blockchain and stored the keyword index on the blockchain for immutability. For healthcare cyber-physical systems, Mamta et al. [41] harnessed ABSE and blockchain to avoid untrusted authorities and heavy computational overhead.

None of the existing ABE and blockchain-based searchable schemes has considered the combination of fine-grained ABE and coarse-grained search. Table I illustrates the comparisons between our scheme and the most related ones. From Table I, we observe that our ERS-ABE scheme with expressive access policy not only is replayable-CCA secure but also achieves efficient user revocation, coarse-grained blockchain-enabled search, and index indistinguishability. In particular, compared with the searchable ABE scheme [25], our ERS-ABE scheme combines multi-authority and blockchain parameter generation. Furthermore, our scheme's coarse-grained search is not based on the complex attribute-based access structure, which can efficiently complete the search for a large amount of data while filtering user-interested information.

III. PRELIMINARIES

3.1 Notations: Table II lists the most important notations used in this paper.

3.2 Bilinear Diffie Hellman problem (BDH): Given a multiplicative cyclic group $\mathbb{G}$ of a prime order p, a generator g of $\mathbb{G}$, and three group elements $g^a, g^b, g^c \in \mathbb{G}$ with three randomly-selected elements $a, b, c \in \mathbb{Z}_p$. The problem of calculating $e(g,g)^{abc} \in \mathbb{G}$ from g^a, g^b and g^c is called Bilinear Diffie Hellman problem.

Note that our scheme's chosen keyword-attack security of indexes is proven based on the difficulty of the BDH problem.

3.3 Pedersen (k,n) Secret Sharing Algorithm: Assume that there are n participants $P = (P_1, ..., P_n)$ in the Pedersen (k,n) secret sharing algorithm [42] and all operations are performed on the finite field $\mathbb{G}_p$, where p is a large prime number. The algorithm follows the four steps below to share a secret S [25]:

(1) Generating the Master Secret S: Each participant P_i independently selects a sub-secret $S_i \in_R \mathbb{Z}_p^*$. Then, the master secret is calculated as $S = \sum_{i=1}^{n} S_i$.

(2) Generating Sub-share Value s_{ij}: Each participant P_i randomly selects a $k-1$th degree polynomial $f_i(x)$ satisfying $f_i(0) = S_i$. Next, the participant P_i calculates its n sub-shares by $\{s_{ij} = f_i(x_j)\}_{j=1,...,n}$, to share each s_{ij} with the related P_j through a secret channel.

(3) Generating the Master Share s_i: Each participant P_i can calculate its master share based on the n sub-shares $\{s_{ji}\}_{j=1,...,n}$ received from other nodes as $s_i = \sum_{j=1}^{n} s_{ji}$. Finally, each participant publishes the parameter g^{s_i}.

(4) Reconstructing the Master Secret: The master secret can be reconstructed by any participant by combining more than k participants' parameters g^{s_i} ($P_{RE} \subset P$, $k \leq |P_{RE}| \leq n$):

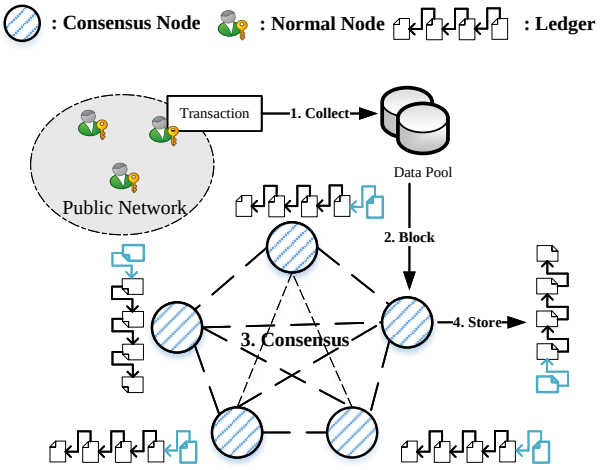


Fig. 1. Permissioned Blockchain Model

$$g^S = \prod_{P_i \in P_{RE}} (g^{s_i})^{L(i)} = g^{\sum_{P_i \in P_{RE}} s_i \cdot L(i)},$$

where $L(i) = \prod_{P_i, P_j \in P_{RE}, i \neq j} \frac{j}{j-i} (mod\ p)$.

The Pedersen (k, n) secret sharing algorithm can share a secret among several participants without a central distributor. Thus, in our scheme, blockchain nodes jointly perform this algorithm to generate system parameters and implement the index-trapdoor-match search.

3.4 Blockchain.

Generally, a blockchain consists of a group of nodes, a data pool used to deposit transactions, and a tamper-resistant ledger constructed by hash-linked blocks. The greatest advantage of blockchain lies in that no malicious user, who controls less than 50 percent capacity, can modify the data stored in the ledger without being noticed. According to different permission conditions, blockchains can be categorized as public blockchains, private blockchains, and coalition blockchains. In a public blockchain (permissionless blockchain), such as Bitcoin and Ethernet, anyone can join its peer-to-peer network and contribute to the ledger maintenance. While in a private blockchain, which belongs to a private organization, only the permissioned nodes can access the ledger and participate in the consensus.

An eclectic type refers to a permissioned blockchain as demonstrated in Fig. 1, where all nodes can read the ledger but only a set of pre-defined consensus nodes can add new blocks into the ledger by using the consensus algorithm. As shown in Fig. 1, data processing contains three steps. (1) The normal nodes are responsible for data generation by submitting transactions to the data pool. (2) The consensus nodes select and pack selected transactions into a block. (3) A consensus algorithm is performed among all consensus nodes to decide whether the block should be accepted or rejected. The consensus algorithm accepts a bock only if more than 50 percent of the consensus nodes approve it. Once a block has been accepted, it will be stored in the ledger with hash-linked blocks. As a result, the data stored in a blockchain cannot be tempered unless an adversary can control more than 50 percent of the consensus nodes.

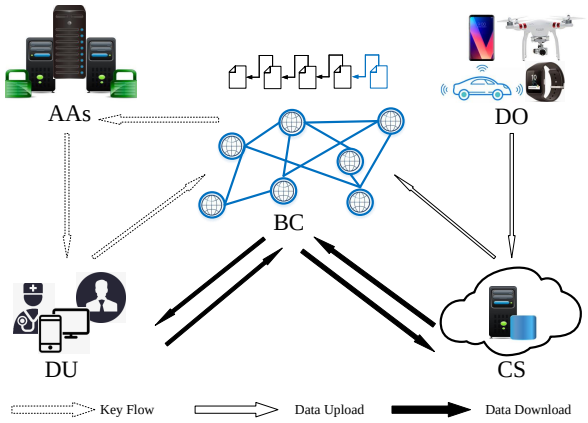


Fig. 2. System Model

In our scheme, a permissioned blockchain is used to replace the traditional central key generation center so that the single-point security threat can be avoided. Moreover, efficient search and user revocation are achieved through the blockchain.

IV. Scheme Design

A. Participants

Fig. 2 depicts the basic model of the ERS-ABE scheme. The five participants are described as follows.

(1) The attribute authorities (AAs) control the disjoint attribute sets, which are used to generate the decryption keys for legitimate users, as well as the attribute update key and the ciphertext update key when an attribute needs to be updated.

(2) The data owners (DO) have the right to encrypt data with their self-defined access policies. They build keyword indexes, store the indexes in the blockchain, and send ciphertexts to the cloud.

(3) The permissioned blockchain (BC) consists of a set of pre-defined trusted nodes, which are responsible for public parameter generation, user registration, search, and user revocation.

(4) The data users (DU) need to register to the BC with their unique global IDs and attribute sets. They get their decryption keys from the associated attribute authorities. To request a search and access, a user generates its outsourced decryption key and sends it along with its authorization key and the trapdoor of a keyword to the BC. After receiving the semi-decrypted files from the cloud, a user can fully decrypt them with its secret key.

(5) The cloud server (CS) is responsible for storing the encrypted files, semi-decrypting for users, and re-encrypting the related ciphertexts when an attribute is updated.

B. Workflow

This section describes the basic workflow of the ERS-ABE, as shown in Fig. 3. Details of the concrete construction can be found in Section V.

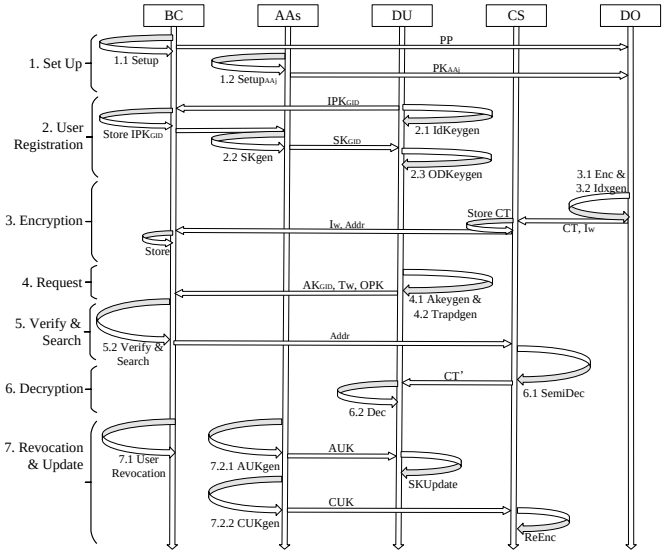


Fig. 3. Workflow

1. Set Up

1.1 $\mathrm{BC.Setup}(\lambda) \rightarrow \mathrm{PP}$: This algorithm is performed by the blockchain nodes, which generate and publish the public parameter PP based on the input security parameter λ.

1.2 $\mathrm{AA}_j.\mathrm{Setup}_{\mathrm{AA}}(\mathrm{PP}, j) \rightarrow (\mathrm{PK}_{\mathrm{AA}_j}, \mathrm{SK}_{\mathrm{AA}_j})$: Each attribute authority AA_j runs this algorithm to generate a pair of keys $(\mathrm{PK}_{\mathrm{AA}_j}, \mathrm{SK}_{\mathrm{AA}_j})$, where the public key will be published for all participants.

2. User Registration

2.1 $\mathrm{DU.IdKeygen}(\mathrm{PP}, \mathrm{GID}) \rightarrow (\mathrm{ISK}_{\mathrm{GID}}, \mathrm{IPK}_{\mathrm{GID}})$: A legitimate user with global identification GID performs this algorithm to get its id key pairs. Then, it sends its public id key to the blockchain for registration.

2.2 $\mathrm{AA}_j.\mathrm{SKgen}(\mathrm{PP}, \mathrm{GID}, S_{\mathrm{GID}}, \mathrm{PK}_{\mathrm{AA}_j}, \mathrm{SK}_{\mathrm{AA}_j}) \rightarrow \mathrm{SK}_j$: Each related attribute authority runs this algorithm to generate the decryption key SK_j and sends it to the user to construct the user decryption key $\mathrm{SK}_{\mathrm{GID}}$.

2.3 $\mathrm{DU.ODKeygen}(\mathrm{PP}, \mathrm{GID}, \mathrm{SK}_{\mathrm{GID}}) \rightarrow (\mathrm{OPK}, \mathrm{OSK})$: A user runs this algorithm to generate its outsourced decryption key that is used by the cloud to out-decrypt.

3. Encryption

3.1 $\mathrm{DO.Enc}(\mathrm{PP}, \{\mathrm{PK}_{\mathrm{AA}_j}\}, (\mathbb{M}, \rho), Fl) \rightarrow \mathrm{CT}$: The DO runs this algorithm with a self-specified access policy $(\mathbb{M}, \rho)$ to generate the ciphertext CT.

3.2 $\mathrm{DO.Idxgen}(\mathrm{PP}, S_w) \rightarrow \mathrm{I}_w$: The DO runs this algorithm to generate the keyword index set I_w of a keyword set S_w, which is extracted from the file Fl.

4. Request

4.1 $\mathrm{DU.AKeygen}(\mathrm{PP}, \mathrm{ISK}_{\mathrm{GID}}) \rightarrow \mathrm{AK}_{\mathrm{GID}}$: Before sending search requests to the blockchain, a user first performs this algorithm to get the authorization key for identity verification.

4.2 $\mathrm{DU.Trapdgen}(\mathrm{PP}, w) \rightarrow \mathrm{T}_w$: Users take this algorithm to produce the trapdoor T_w of a search keyword w.

Finally it sends $(\mathrm{AK}_{\mathrm{GID}}, \mathrm{T}_w, \mathrm{OPK})$ to the BC for search.

5. Verify & **Search**

5.1 $\mathrm{BC.Verify}(\mathrm{AK}_{\mathrm{GID}}, \mathrm{IPK}_{\mathrm{GID}}) \rightarrow (0, 1)$: The BC runs this algorithm to verify the user's identity and the trapdoor.

5.2 $\mathrm{BC.Search}(\mathrm{T}_w, \mathrm{I}_w) \rightarrow \{0, 1\}$: If a user's request passes the verification, the BC performs this algorithm to search for files that contain the search keyword.

6. Decryption.

6.1 $\mathrm{CS.SemiDec}(\mathrm{PP}, \mathrm{OPK}, \mathrm{CT}) \rightarrow \mathrm{CT}'$: The CS performs this algorithm to semi-decrypt files for users. It sends the semi-decrypted ciphertext to the DU.

6.2 $\mathrm{DU.Dec}(\mathrm{CT}', \mathrm{OSK}) \rightarrow Fl$: The user runs this algorithm to get the requested file Fl.

7. Revocation & Update

This phase achieves the goals of user revocation and attribute updates.

7.1 User Revocation: In our scheme, user revocation is super easy. All we have to do is send a new transaction to the blockchain to make the user's public id key invalid. As a result, the subsequent requests from the user cannot pass verification.

7.2 Attribute Update: The attribute i of the user GID can be updated by the following two steps:

7.2.1 Key Update: The attribute authority AA_j, which controls the to-be-updated attribute i, runs the $\mathrm{AUKgen}(\mathrm{SK}_{\mathrm{AA}_j}) \rightarrow \mathrm{AUK}$ algorithm to generate the attribute update key and updates its $\mathrm{PK}_{\mathrm{AA}_j}$. Then, it sends AUK to all the users, whose attribute set contains this attribute, except the revoked ones. Finally, each of these users runs the $\mathrm{SKUpdate}(\mathrm{SK}, \mathrm{AUK}) \rightarrow \mathrm{SK}'$ algorithm to update its SK.

7.2.2 Ciphertext update: The attribute authority, which controls the to-be-updated attribute, runs the $\mathrm{CUKgen}(\mathrm{CT}, \mathrm{SK}_{\mathrm{AA}_j}) \rightarrow \mathrm{CUK}$ algorithm to generate the ciphertext update key CUK and sends it to the CS. to run the $\mathrm{ReEnc}(\mathrm{CUK}, \mathrm{CT}) \rightarrow \mathrm{CT}'$ algorithm to update those ciphertexts whose access policy contains this to-be-updated attribute i. Note that our ciphertext updating method can improve the efficiency as it only partially re-encrypts a ciphertext.

C. Security Models

1) Message Confidentiality: Our ERS-ABE scheme adopts the selective replayable chosen-ciphertext-attack (RCCA) security model in [3]. Let Pedersen (k, n) be the Pedersen algorithm defined in Section III, S_{AA} be the universal set of attribute authorities, and S'_{AA} be the set of corrupted authorities. We have the following three assumptions:

(1) The adversary can query any decryption key except those that are used to decrypt the challenge ciphertext within polynomial time.

(2) The adversary can only corrupt the authority statically as in [11].

(3) The adversary can compromise at most $k-1$ consensus nodes, where k is the threshold in the Pedersen algorithm.

Our multi-authority ABE scheme is RCCA-Secure if (1) the Pedersen (k, n) secret sharing algorithm described in Section III [42] is selective CPA-secure, and (2) there is no probabilistic polynomial time (PPT) adversary that can win the confidentiality security experiment (interacted between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$) with non-negligible advantage.

1) **Init:** $\mathcal{A}$ chooses a challenge access structure $\Psi^* = (\mathbb{M}^*, \rho^*)$ and a set of revoked attributes S_{R}. Then, $\mathcal{A}$ sends (Ψ^*, S_{R}) to $\mathcal{C}$.

2) **Set Up:** $\mathcal{A}$ chooses a set of corrupted attribute authorities $S'_{\mathrm{AA}} \subset S_{\mathrm{AA}}$. $\mathcal{C}$ invokes the Pedersen secret sharing algorithm to run the $\mathrm{Setup}(\lambda)$ algorithm to generate the global parameter PP (assuming that a set of pre-defined blockchain nodes are responsible for performing the Pedersen algorithm and the adversary can compromise at most $k-1$ of them). $\mathcal{A}$ and $\mathcal{C}$ respectively run the $\mathrm{Setup}_{\mathrm{AA}}$ algorithm to generate public and private keys for the corrupt authorities and the honest authorities. For each attribute in the revocation set $i \in S_{\mathrm{R}}$, $\mathcal{C}$ updates its key pair and returns the public keys to $\mathcal{A}$.

3) **Query Phase 1:** An empty table $\mathbb{T}$ and an empty set $\mathbb{D}$ are initialized by $\mathcal{C}$. Then, $\mathcal{A}$ conducts m queries:

a) Hash Query: $\mathcal{A}$ queries the random oracle H or H_3.

b) Key Query: This phase includes queries of the decryption key $\mathrm{SK}_{\mathrm{GID}_k}$ and the outsourced decryption key $\mathrm{OPK}_{\mathrm{GID}_k}$ associated with an attribute set S_{GID_k} (for the kth query). Note that all attributes contained in S_{GID_k} are controlled by the honest attribute authorities and all S_{GID_k} cannot satisfy the challenge access structure $\Psi^* = (\mathbb{M}^*, \rho^*)$. $\mathcal{C}$ searches the entry $(S_{\mathrm{GID}_k}, \mathrm{SK}_{\mathrm{GID}_k}, \mathrm{OPK}_{\mathrm{GID}_k})$ in the table $\mathbb{T}$. If it exists, $\mathcal{C}$ returns $(\mathrm{SK}_{\mathrm{GID}_k}, \mathrm{OPK}_{\mathrm{GID}_k})$ to $\mathcal{A}$; otherwise, $\mathcal{C}$ runs the $\mathrm{SKgen}(\mathrm{PP}, \mathrm{GID}_k, S_{\mathrm{GID}_k}, \mathrm{PK}_{\mathrm{AA_j}}, \mathrm{SK}_{\mathrm{AA_j}})$ algorithm to generate S_{GID_k}. Finally $\mathcal{C}$ sets $\mathbb{D} = \mathbb{D} \cup S_{\mathrm{GID}_k}$.

Next, $\mathcal{C}$ randomly chooses an element $h \in \mathbb{Z}_p^*$ to run the $\mathrm{ODKeygen}(\mathrm{PP}, \mathrm{GID}_k, \mathrm{SK}_{\mathrm{GID}_k})$ algorithm. Then, it stores the entry $(S_{\mathrm{GID}_k}, \mathrm{SK}_{\mathrm{GID}_k}, \mathrm{OPK}_{\mathrm{GID}_k})$ in the table $\mathbb{T}$. In the meantime, $\mathcal{C}$ runs the $\mathrm{AUKgen}(\mathrm{PP}, \mathrm{CT}, \mathrm{SK}_{\mathrm{AA_j}}, i)$ algorithm to generate the AUK_i for each attribute $i \in S_{\mathrm{R}}$. Finally, $\mathcal{C}$ sends $(\mathrm{SK}_{\mathrm{GID}_k}, \mathrm{OPK}_{\mathrm{GID}_k}, \mathrm{AUK})$ back to $\mathcal{A}$.

c) Decryption Query: $\mathcal{C}$ firstly searches the Table $\mathbb{T}$ for an entry $(S_{\mathrm{GID}_k}, \mathrm{SK}_{\mathrm{GID}_k}, \mathrm{OPK}_{\mathrm{GID}_k})$. If not, abort; otherwise, $\mathcal{C}$ runs the $\mathrm{SemiDec}(\mathrm{PP}, \mathrm{OPK}_{\mathrm{GID}_k}, \mathrm{CT}) \rightarrow \mathrm{CT}'$ algorithm and the $\mathrm{Dec}(\mathrm{CT}', \mathrm{OSK}_{\mathrm{GID}_k}) \rightarrow Fl$ algorithm to return Fl to $\mathcal{A}$.

4) **Challenge:** $\mathcal{A}$ chooses two length-equaled messages, K_0 and K_1, and sends them to the challenger. Next, $\mathcal{C}$ randomly chooses a bit $b \in \{0,1\}$ to encrypt K_b under the access policy $(\mathbb{M}^*, \rho^*)$ and gets CT_b. Finally $\mathcal{C}$ sends CT_b back to $\mathcal{A}$.

5) **Query Phase 2:** After receiving CT_b, $\mathcal{A}$ can continue to adaptively request a polynomial-bounded number of queries as described in Query Phase 1 with two limitations: (a) the queried attribute set cannot satisfy the challenge access structure as defined before, and (b) the response of the decryption query can be neither K_0 nor K_1.

6) **Guess:** $\mathcal{A}$ gives a guess of b according to CT_b. The following equation is defined as the adversary advantage of winning this security experiment:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{conf}}(1^\lambda) = |\Pr[b = b'] - \frac{1}{2}|.$$

Definition 1: A blockchain-assisted revocable MA-ABE scheme with outsourced decryption is selective RCCA-Secure against static corruption of the attribute authorities if $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{conf}}(1^\lambda)$ is negligible for all PPT adversaries.

2) Index Confidentiality: Our ERS-ABE scheme is secure against chosen-keyword attacks (CKA-Secure) [26], which implies that our scheme does not reveal any index information unless the associated trapdoor is available. The third assumption in Section IV-C1 is still applicable in this model. As defined in Section IV-A, the consensus nodes in the blockchain are a set of secure and trusted nodes defined in advance, but these nodes may be offline or corrupted. Because no single node can independently complete the reconstruction of a system secret parameter, the entire blockchain can be regarded as secure as long as Pedersen's (k, n) secret sharing scheme is CPA-secure, which means that the adversary cannot collude with more than k consensus nodes.

The experiment of CKA-security between a PPT attacker $\mathcal{A}$ and the challenger $\mathcal{C}$ is described as follows:

1) **Set Up:** $\mathcal{C}$ invokes the Pedersen secret sharing algorithm to run the $\mathrm{Setup}(\lambda)$ algorithm as described in the Confidentiality security experiment. $\mathcal{C}$ sends the global parameter PP to $\mathcal{A}$.

2) **Query Phase 1:** $\mathcal{A}$ queries a polynomial-bounded number of queries as follows:

a) $\mathrm{H}_1, \mathrm{H}_2$ Query: $\mathcal{A}$ queries the random hash oracles H_1 and H_2.

b) Trapdoor Query: $\mathcal{A}$ sends a set of keywords to $\mathcal{C}$. $\mathcal{C}$ runs the $\mathrm{Trapdgen}(\mathrm{PP}, w)$ algorithm to generate the corresponding trapdoors for these keywords and returns them back to $\mathcal{A}$.

3) **Challenge:** $\mathcal{A}$ submits the challenge keyword pair (w_0^*, w_1^*) to $\mathcal{C}$ where neither w_0^* nor w_1^* has been queried in Phase 1. After receiving the challenge keyword pair, $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$ to run the $\mathrm{Idxgen}(\mathrm{PP}, S_w)$ algorithm to generate $I_{w_b}^*$ and returns it back to $\mathcal{A}$.

4) **Query Phase 2:** $\mathcal{A}$ queries a polynomial-bounded number of queries as in Query Phase 1 after receiving $I_{w_b}^*$.

5) **Guess:** $\mathcal{A}$ outputs a guess b' of b. If $b' = b$, we say that the adversary $\mathcal{A}$ wins this experiment. We define the adversary's advantage of winning this security experiment as:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{cka}}(1^\lambda) = |\Pr[b = b'] - \frac{1}{2}|.$$

Definition 2: A revocable MA-ABE scheme with blockchain-enabled search is CKA-Secure if $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{cka}}(1^\lambda)$ is negligible for all PPT adversaries.

V. ERS-ABE Construction

I: Set Up

1) The blockchain nodes run the Pedersen secret sharing algorithm to generate three parameters (g^μ, g^c, g^γ). Then, the node, who performs the final step in the pedersen algorithm, continues the following steps: (a) chooses two multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with a generator $g \in \mathbb{G}$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Let $\mathbb{D} = (\mathbb{G}, \mathbb{G}_T, p, g, e)$; (b) selects four hash functions $\mathrm{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*, \mathrm{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}, \mathrm{H}_2 : \mathbb{G}_T \rightarrow \{0,1\}^{log_p}, \mathrm{H}_3 : \{0,1\}^* \rightarrow \{0,1\}^k$, which can be modeled as random oracles; and (3) publishes the public parameter:

$$\mathrm{PP} = \{\mathbb{D}, \mathrm{H}, \mathrm{H}_1, \mathrm{H}_2, \mathrm{H}_3, g^\mu, g^c, g^\gamma\}.$$

2) Each attribute authority AA_j with a unique attribute set S_{AA_j} chooses a random number $h_j \in \mathbb{Z}_p^*$ and three random

elements $\alpha_i, v_i, \beta_i \in \mathbb{Z}_p^*$ for each attribute $i \in S_{\mathrm{AA}_j}$. Finally it publishes its public key and keeps the private key secret:

$$\mathrm{PK}_{\mathrm{AA}_j} = (\{P_{i,1} = g^{\alpha_i v_i}, g^{\beta_i}\}_{i \in S_{\mathrm{AA}_j}}, g^{h_j}),$$
$$\mathrm{SK}_{\mathrm{AA}_j} = (\{\alpha_i, v_i, \beta_i\}_{i \in S_{AA_j}}, h_j).$$

II: User Registration

1) A unique identity GID and an attribute set S_{GID} are assigned to a data user. The DU first randomly chooses an element $X \in \mathbb{Z}_p^*$ to calculate its id key pair and sends its public id key $\mathrm{IPK}_{\mathrm{GID}}$ to the BC for registration:

$$\mathrm{IPK}_{\mathrm{GID}} = g^{\mu/X}, \quad \mathrm{ISK}_{\mathrm{GID}} = X.$$

2) After user registration, each related attribute authority, who controls one or several attributes in S_{GID}, generates the decryption key SK_j for the user GID:

$$\begin{aligned}\mathrm{SK}_j &= \{\mathrm{SK}_i\}_{i \in S_{\mathrm{GID}}} \\ &= \{g^{\alpha_i v_i} \mathrm{H}_1(\mathrm{GID})^{\beta_i}\}_{i \in S_{\mathrm{GID}}}.\end{aligned}$$

The DU will collect all SK_j to construct its complete decryption key $\mathrm{SK}_{\mathrm{GID}}$.

3) The DU chooses a random number $z \in \mathbb{Z}_p^*$ as its outsourced private decryption key OSK and computes the components of the outsourced public decryption key $\mathrm{OPK} = (\{\mathrm{OPK_i} = (\mathrm{SK_i})^{1/z}\}, \mathrm{OPK}_2 = (\mathrm{H}_1(\mathrm{GID}))^{1/z}, \mathrm{OPK}_3 = \mathrm{g}^{1/z})$.

III: Encryption

1) The DO chooses a random value $R \in \mathbb{G}_T$ and runs the symmetric encryption algorithm to encrypt the file Fl with a symmetric key $K = \mathrm{H}_3(R)$:

$$C_0 = \mathrm{Enc}_{sym}(K, Fl) = \mathrm{H}_3(R) \oplus Fl.$$

Then, the DO encrypts the random value R with an access policy $\Psi = (\mathbb{M}, \rho)$. The DO calculates the shared secret $s = \mathrm{H}(R, Fl)$ and chooses two random vectors $v = (s, v_2, v_3, ..., v_l) \in \mathbb{Z}_p^l$ and $\omega = (0, \omega_2, \omega_3, ..., \omega_l) \in \mathbb{Z}_p^l$ to compute $\lambda_i = \mathbb{M}_i v$ and $\omega_i = \mathbb{M}_i \omega$. For each row $\mathbb{M}_i$ of $\mathbb{M}$, the DO selects a random number $r_i \in \mathbb{Z}_p$ to compute the rest of the ciphertext as follows:

$$\begin{cases} C = Re(g,g)^s, \\ C_{i,1} = (g^{\alpha_{\rho(i)} v_{\rho(i)}})^{r_i} g^{\lambda_i}, \\ C_{i,2} = (g^{\beta_{\rho(i)}})^{r_i} g^{\omega_i}, \\ C_{i,3} = g^{r_i}. \end{cases}$$

Let $\mathrm{CT} = (C_0, C, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i \in [l]}, (\mathbb{M}_{l \times n}, \rho))$ be the ciphertext.

2) The DO first extracts a keyword set $S_w = (w_1, w_2, ..., w_{n'})$ from the file Fl. Then, it selects a random number $\xi_i \in \mathbb{Z}_p^*$ for each keyword $w_i \in S_w$. Next, it computes $\mathrm{I}_w = \{\mathrm{I}_{w_i}\}_{w_i \in S_w} = \{[I_1, I_2]\}$ as follows:

$$I_1 = g^{\xi_i}, \quad I_2 = \mathrm{H}_2(e((g^\gamma)^{\xi_i}, \mathrm{H}_1(w_i))).$$

In the end, the DO sends $(\mathrm{CT}, \mathrm{I}_w)$ to the CS, which stores CT and sends $(\mathrm{I}_w, Addr)$ to the BC, where $Addr$ is the address of the ciphertext stored in the cloud. Then, the BC saves $(\mathrm{I}_w, Addr)$ in the ledger.

IV: Request

1) The DU first randomly selects a group element $\delta \in \mathbb{Z}_p^*$ to calculate its authentication key $\mathrm{AK}_{\mathrm{GID}} = (\mathrm{AK}_1, \mathrm{AK}_2)$ as:

$$\mathrm{AK}_1 = (g^c)^\delta (g^\mu)^{1/X}, \quad \mathrm{AK}_2 = g^\delta.$$

2) The DU selects a random number $\tau \in \mathbb{Z}_p^*$ to generate the half-baked trapdoor $\mathrm{T}_w = (T_{w_1}, T_{w_2}, T_{w_3})$ for a keyword w as:

$$\begin{cases} T_{w_1} = \mathrm{H}_2(e(g^\gamma, (g^c)^\tau)), \\ T_{w_2} = g^\tau, \\ T_{w_3} = \mathrm{H}_1(w). \end{cases}$$

Finally, the user sends $(\mathrm{AK}_{\mathrm{GID}}, \mathrm{T}_w, \mathrm{OPK})$ to the BC.

V: Verify & **Search**

1) The blockchain nodes perform the Pedersen algorithm to generate the parameter $(\mathrm{AK}_2)^c$. Then, the node, which carries out the final step in the Pedersen algorithm, verifies whether the following equation holds:

$$\mathrm{AK}_1 / (\mathrm{AK}_2)^c = \mathrm{PK}_{\mathrm{GID}}.$$

If it does not hold, the search request is denied; otherwise, the legality of the user is verified and the BC continues the search as follows.

2) The blockchain nodes perform the Pedersen algorithm to generate the parameters $((\mathrm{H}_1(w))^\gamma, (T_{w2})^c)$. Then, the node, which carries out the final step in the pedersen algorithm, computes $\varphi = \mathrm{H}_1(e((T_{w2})^c, g^\gamma))$ and $\theta = (\mathrm{HK}_1(w))^\gamma \oplus T_{w_1}$ to search by pairing the following equation:

$$\mathrm{H}_2(e(I_1, \theta \oplus \varphi)) = I_2.$$

Finally, the BC node sends $(\mathrm{OPK}, \{Addr\})$ to the CS.

VI: Decryption

1) The CS first gains the access structure from CT to identify the set of attributes $A' = \{i : (\rho(i) \cap S_{\mathrm{GID}})_{i \in [n]}\}$ required for the decryption. Then, it chooses a set of constants $\{o_i\}_{i \in [1,n]}$ such that $\sum_i o_i \mathbb{M}_i = [1, 0, ..., 0]$ to decrypt CT using OPK, where $i \in [n]$:

$$\begin{aligned}\mathrm{CT}' &= \prod_{i=1}^{n} \left(\frac{e(\mathrm{C}_{i,1}, \mathrm{OPK}_3) e(\mathrm{C}_{i,2}, \mathrm{OPK}_2)}{e(\mathrm{C}_{i,3}, \mathrm{OPK}_i)}\right)^{o_i} \\ &= e(g,g)^{s/z}.\end{aligned} \tag{1}$$

The decryption process begins only when the user's attribute set satisfies the access policy. The correctness of Eq. (1) can be derived directly with $\sum_{i=1}^n \lambda_i o_i = \sum_{i=1}^n \mathbb{M}_i \vec{v} o_i = \vec{v}[1, 0, ..., 0] = s$ and $\sum_{i=1}^n w_i o_i = \sum_{i=1}^n \mathbb{M}_i \vec{w} o_i = \vec{w}[1, 0, ..., 0] = 0$, where $\lambda_i = \mathbb{M}_i \vec{v}$ and $w_i = \mathbb{M}_i \vec{w}$.

Finally, the CS sends (C_0, C, CT') back to the DU.

2) The DU first computes the following equation to recover the random number $R = C/(\mathrm{CT}')^{\mathrm{OSK}}$. Then, the user recovers the file with the symmetric decryption key $K = \mathrm{H}_3(R)$:

$$Fl = \mathrm{Dec}_{sym}(K, C_0) = C_0 \oplus \mathrm{H}_3(R).$$

The DU verifies whether $\mathrm{CT}' = e(g,g)^{s/z}$ holds with $s = \mathrm{H}(R, Fl)$ and $\mathrm{OSK} = z$. If not, the decryption fails; otherwise, the DU successfully acquires the file Fl.

VII: User Revocation & **Attribute Update:**

1) User Revocation: In our scheme, user revocation is extremely easy to accomplish. Compared with other complex

revocation methods, such as tree-based user management [25], our ERS-ABE scheme only needs to send a transaction to the BC to mark the identity key of the revoked user as invalid/revoked. As a result, the verification cannot succeed when a revoked user tries to inquire for a search.

2) Attribute update: In the attribute update part, for each attribute $i \in S_{\mathrm{R}}$ of the user GID, the corresponding attribute authority AA_j conducts the following two steps:

a) AA_j generates the AUK of the attribute i: $\mathrm{AUK}_i = g^{\alpha_i(v_i'-v_i)}$, to update its public key by computing:

$$P_{i,1} = g^{\alpha_i v_i} \cdot g^{\alpha_i(v_i'-v_i)} = g^{\alpha_i v_i'}.$$

Then, it sends the AUK to the users, who have the attribute i except the revoked user GID, to update their secrete keys by computing:

$$\begin{aligned}\mathrm{SK}_i &= g^{\alpha_i v_i}\mathrm{H}(\mathrm{GID})^{\beta_i} \cdot g^{\alpha_i(v_i'-v_i)} \\ &= g^{\alpha_i v_i'}\mathrm{H}(\mathrm{GID})^{\beta_i}.\end{aligned}$$

b) AA_j generates $\mathrm{CUK}_i = (C_{i,3})^{\alpha_i(v_i'-v_i)}$ and sends it to the cloud to re-encrypt the ciphertext. Note that our scheme improves the update efficiency because it only needs to re-encrypt $C_{i,1}$ instead of the whole CT:

$$\begin{aligned}C_{i,1} =& C_{i,1} \cdot \mathrm{CUK}_i \\ =& (g^{\alpha_{\rho(i)}v_{\rho(i)}})^{r_i} g^{\lambda_i} \cdot (C_{i,3})^{\alpha_i(v_i'-v_i)} \\ =& (g^{\alpha_{\rho(i)}v'_{\rho(i)}})^{r_i} g^{\lambda_i}.\end{aligned}$$

VI. Security Analysis

A. Message Confidentiality

Theorem 1: If Lewko et al.'s decentralized ABE scheme [11] and the Pedersen (k, n) secret sharing algorithm [42] are selectively secure against the chosen-plaintext attacks, the ERS-ABE scheme is selectively secure against the replayable chosen-ciphertext attacks according to Definition 1.

The proof of this theorem is defined in an experiment between the PPT adversary $\mathcal{A}$ and the challenger $\mathcal{C}$, which is similar to that in another MA-ABE scheme [15]. Both of them are based on the security of Lewko et al.'s decentralized ABE scheme [11], but our security model requires an extra assumption as demonstrated in Section IV-C1 (3). Define the advantage of the PPT adversary to win the confidentiality experiment in Lewko et al.'s scheme as $\mathsf{Adv}^{\mathrm{lewko}}_{\mathrm{Conf}}(1^\lambda)$ and the advantage of the PPT adversary to win the experiment defined in Section IV-C1 as $\mathsf{Adv}^{\mathrm{ers}}_{\mathrm{Conf}}(1^\lambda)$, this proof aims at showing that $\mathsf{Adv}^{\mathrm{ers}}_{\mathrm{Conf}}(1^\lambda)$ is smaller than $\mathsf{Adv}^{\mathrm{lewko}}_{\mathrm{Conf}}(1^\lambda)$.

Due to the similarity, the proof details are omitted here. It is worth noting that the differences in the message confidentiality proof between scheme [15] and ours are, in the setup phase, $\mathcal{C}$ in our scheme needs to invoke the Pedersen secret sharing algorithm to accomplish generating public parameter. Same as in [15], assuming there is an algorithm $\mathcal{B}$ that is used to run Lewko et al.'s scheme [11]. Recall that the security of Lewko et al.'s scheme is based on the difficulty of the Decisional Bilinear Diffie Hellman (DBDH) problem, which means $\mathsf{Adv}^{\mathrm{lewko}}_{\mathrm{Conf}}(1^\lambda) \approx \varepsilon$ (ε is negligible). Because $\mathcal{B}$winning the confidentiality experiment in Lewko et al.'s scheme [11] is a condition for $\mathcal{A}$to win our message confidentiality experiment, one can get that the advantage of the PPT adversary to win the confidentiality experiment in our ERS-ABE scheme is negligible.

B. Index Confidentiality

Theorem 2: The ERS-ABE scheme is semantically secure against the chosen keyword attacks under the random oracle model if the BDH assumption holds and the Pedersen (k, n) secret sharing algorithm [42] is selectively secure against the chosen plaintext attacks.

Proof: Let $\mathcal{A}$ be the PPT adversary, and $\mathcal{C}$ be the challenger who tries to solve the BDH problem with advantage $\varepsilon' = 2\varepsilon/(eq_{H_2}q_T)$, where ε is the advantage of the adversary $\mathcal{A}$ winning this experiment, e is the base of the natural logarithm, and q_{H_2} and q_T are maximum numbers of oracle queries about the hash function H_2 and the trapdoor (q_{H_2} and q_T are positive numbers). As defined in Section IV-C2, the CPA-security of the Pedersen (k, n) secret sharing scheme guarantees that the adversary cannot collude with more than k blockchain nodes. The following proof aims to show that the adversary cannot distinguish the indexes of two keywords it has never queried.

1) **Set Up:** $\mathcal{C}$ receives a BDH challenge: $(u_1 = g^a, u_2 = g^b, u_3 = g^d \in \mathbb{G})$. The aim of $\mathcal{C}$ is to compute $e(g, g)^{abd} \in \mathbb{G}_T$. First, $\mathcal{C}$ contrusts $\mathbb{D}$ by choosing two multiplicative cyclic groups of prime order p, $\mathbb{G}$ and $\mathbb{G}_T$, a generator g of $\mathbb{G}$, a bilinear map $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$, and two collusion resistant hash functions, H_1^* and H_2^*. Next, $\mathcal{C}$ invokes the Pedersen algorithm to get the parameters g^c and $g^\gamma = g^{a\cdot t_1} = u_1^{t_1}$ with the input g^a. $\mathcal{C}$ returns $\mathrm{PP} = \{\mathbb{D}, \mathrm{H}_1, \mathrm{H}_2, g^\gamma, g^c\}$ to $\mathcal{A}$.

2) **Query Phase 1:** $\mathcal{A}$ adaptively requests a polynomial-bounded number of queries as follows:

a) H_1 Query: $\mathcal{A}$ can query this random oracle whenever she wants. $\mathcal{C}$ first initializes an empty H_1 list of entry (w_i, h_i, e_i, c_i). When $\mathcal{A}$ queries H_1 of a keyword $w_i \in \{0,1\}^*$, $\mathcal{C}$ performs the following action: $\mathcal{C}$ first searches the H_1 list. If the parameter w_i already exists, it returns $\mathrm{H}_1(w_i) = h_i \in \mathbb{G}$; otherwise, $\mathcal{C}$ chooses a random number $c_i \in \{0,1\}$ so that $\Pr[c_i = 0] = \frac{1}{q_T+1}$. If $c_i = 0$, $\mathcal{C}$ computes $h_i = u_2^{e_i} \in \mathbb{G}$; if $c_i = 1$, $\mathcal{C}$ computes $h_i = g^{e_i} \in \mathbb{G}$, where $e_i \in \mathbb{Z}_p^*$ is randomly selected. Finally, $\mathcal{C}$ returns $\mathrm{H}_1(w_i) = h_i$ while storing the entry (w_i, h_i, e_i, c_i) in the H_1 list.

b) H_2 Query: Similarly, assuming $\mathcal{A}$ can query this random oracle whenever she wants. $\mathcal{C}$ first initializes an empty H_2 list of entry (t_i, V_i). When $\mathcal{A}$ queries H_2 of any $t_i \in \mathbb{G}_T$, $\mathcal{C}$ performs the following action. $\mathcal{C}$ first searches the H_2 list. If the element t_i already exists, it returns $\mathrm{H}_2(t_i) = V_i \in \{0,1\}^{log_p}$; otherwise, $\mathcal{C}$ randomly selectes an element $V_i \in \{0,1\}^{lop_p}$ and returns $\mathrm{H}_2(t_i) = V_i$ to $\mathcal{A}$ while storing the entry (t_i, V_i) in the H_2 list.

c) Trapdoor Query: $\mathcal{A}$ queries keywords $w_i \in \{0,1\}^*$ to $\mathcal{C}$. $\mathcal{C}$ first queries the oracle H_1 to get $\mathrm{H}_1(w_i) = h_i \in \mathbb{G}$ and the corresponding entry (w_i, h_i, e_i, c_i). Then, $\mathcal{C}$ performs the following action: If $c_i = 0$, $\mathcal{C}$ aborts. Else, $h_i = g^{e_i} \in \mathbb{G}$. Then, $\mathcal{C}$ randomly selects a number $\tau \in \mathbb{Z}_p^*$ to query the H_2 oracle to get $\mathrm{H}_2(e(g^\gamma, (g^c)^\tau))$. Then, $\mathcal{C}$ computes:

TABLE III
STORAGE OVERHEADS.

Scheme	[14]	[24]	[23]	ERS-ABE
Secret Key Length	$2N_u\|\mathbb{G}\|$	$1\|\mathbb{G}\|+2\|\mathbb{Z}_p\|$	$(N_u+2)\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$N_u\|\mathbb{G}\|+2\|\mathbb{Z}_p\|$
OutDecryption Key Length	$(N_d+2)\|\mathbb{G}\|+1\|\mathbb{Z}_p^*\|$	$2N_u\|\mathbb{G}\|$	$(N_u+2)\|\mathbb{G}\|$	$(N_u+2)\|\mathbb{G}\|$
Ciphertext Length	$(3N_e+1)\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$(2N_e+1)\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$2N_e\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$3N_e\|\mathbb{G}\|+1\|\mathbb{G}_T\|$
Index Length	-	$N_w(\|\mathbb{G}\|+\|log_p\|)$	-	$N_w(\|\mathbb{G}\|+\|log_p\|)$
Trapdoor Length	-	$2\|\mathbb{G}\|$	-	$2\|\mathbb{G}\|$

Abbreviations: N_e/N_d: number of attributes required to encrypt/decrypt. N_u: number of attributes in S_{GID}. N_w: number of keywords in S_w.

TABLE IV
COMPUTATION OVERHEADS.

Scheme	[14]	[24]	[23]	ERS-ABE
Key Gen	$3N_uE$	$(2N_u+1)E$	$(N_u+6)E+2P$	N_uE
Encryption	$(6N_e+1)E+1E_T+N_eP_e$	$(3N_e+1)E+1E_T$	$(3N_e+1)E$	$(5N_e+1)E+1E_T$
Index Generation	-	$2N_wE+N_wPe$	-	$2N_wE+N_wPe$
Trapdoor Generation	-	$3E$	-	$2E+1P_e$
Search	-	$E+Pe$	-	$2Pe$
OutDecryption	$N_dE_T+3N_dPe$	$N_dE_T+2N_dPe$	$2N_dE_T+2N_dPe$	$N_dE_T+3N_dPe$
User Decryption	$1E_T$	$1Pe$	$1E_T$	$1E_T$
Key Update	-	$1E$	$(N_u+3)E+Pe$	$0E+0E_T+0Pe$
CT Update	-	-	-	$0E+0E_T+0Pe$

Abbreviations: E, E_T: one exponentiation in $\mathbb{G}$, $\mathbb{G}_T$. P_e: one pairing in map $e:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$.

$$\begin{cases} T_{w1}^* &= \mathrm{H}_2(e(g^\gamma,(g^c)^\tau))=\mathrm{H}_2(e(g,g)^{\gamma c\tau}),\\ T_{w2}^* &= g^\tau,\\ T_{w3}^* &= \mathrm{H}_1(w_i)=g^{e_i}. \end{cases}$$

Finally, $\mathcal{C}$ returns $\mathrm{T}_w^*=(T_{w1}^*,T_{w2}^*,T_{w3}^*)$ back to $\mathcal{A}$.

3) **Challenge:** $\mathcal{A}$ submits the challenge keyword pair (w_0^*,w_1^*) ensuring that none of them has been queried in Phase 1. After receiving the challenge keyword pair, $\mathcal{C}$ computes an index as follows:

i) $\mathcal{C}$ queries H_1 twice to get two results $h_0,h_1\in\mathbb{G}$, which satisfy $\mathrm{H}_1(w_0)=h_0,\mathrm{H}_1(w_1)=h_1$. Then, it sets the corresponding tuples $\{w_i,h_i,e_i,c_i\}_{i=(0,1)}$ on the H_1 list. $\mathcal{C}$ declares abortion if both c_0 and c_1 equal to 0 or 1.

ii) One can know that at least one of c_0 and c_1 is equal to 0. Thus, $\mathcal{C}$ can decide a bit $b\in\{0,1\}$ to set $c_b=0$.

iii) $\mathcal{C}$ returns the keyword index $\mathrm{I}_w^*=(I_{w1}^*,I_{w2}^*)$ in two steps. First $\mathcal{C}$ randomly chooses an element $t_2\in\mathbb{Z}_p^*$ to set $I_{w1}^*=(u_3)^{1/t_2}$ with the implicit condition ($\xi=d/t_2$). It worth noting that d is unknown. Then, $\mathcal{C}$ chooses a random number $Z\in\{0,1\}^{log_p}$ to let $I_{w2}^*=Z$. One can know that I_w^* is a valid keyword index for keyword w_b.

4) **Query Phase 2:** $\mathcal{A}$ queries a polynomial-bounded number of queries as in Query Phase I after receiving I_w^*.

5) **Guess:** $\mathcal{A}$ outputs a guess b' of b.

Note that, in the H_1 query, the value $h_i=u_2^{e_i}$ is set with probability $\frac{1}{q_T+1}$. And, in the H_2 query, $\mathcal{A}$ sets the value with the same probability $\frac{1}{q_T+1}$:

$$\begin{aligned} e((g^\gamma)^\xi,\mathrm{H}_1(w_b)) &= e((g^{at_1})^{d/t_2},g^{be_b})\\ &= e(g,g)^{abd(e_bt_1/t_2)}. \end{aligned}$$

Then, $\mathcal{C}$ randomly selects a pair (t_i,V_i) from the H_2 list. Finally, it outputs $t^{e_bt_1/t_2}$ as its guess for $e(g,g)^{abd}$. ■

Probability Analyses: If the advantage of the adversary $\mathcal{A}$ to win this experiment $\mathrm{Exp}_{\mathcal{A}}^{\mathrm{CKA}}(1^\lambda)$ is non-negligible, then the challenger can solve the BDH problem with probability at least $\frac{2\varepsilon}{eq_Tq_{H_2}}$ [43]. According to the BDH assumption defined in Section III, the probability $\frac{2\varepsilon}{eq_Tq_{H_2}}$ is negligible, which implies:

$$\Pr[\mathrm{Exp}_{\mathcal{A}}^{\mathrm{CKA}}(1^\lambda)]=\Pr[\mathrm{Exp}_{\mathcal{C}}^{\mathrm{BDH}}(1^\lambda)]=\frac{2\varepsilon}{\mathrm{eq_Tq_{H_2}}}\approx\varepsilon.$$

Therefore, our scheme is CKA-secure under the BDH assumption.

TABLE V
SIMULATION ENVIRONMENT.

	Pedersen algorithm	ABE algorithm
CPU	IntelCore i7-9700K@2.2GHz	IntelCore i5-1035G4@1.5GHz
OS	Ubuntu 16.4 LTS 64-bit	Ubuntu 20.04.2.0 LTS 64-bit
RAM	8GB	8GB

VII. PERFORMANCE ANALYSIS

This section analyzes the computational and storage costs of the ERS-ABE scheme. The detailed comparison with some related schemes is summarized in Table III and Table IV, where the computational and storage costs of the symmetric cryptography, multiplication operations, and hash operations are omitted.

Regarding the storage costs presented in Table III, the lengths of the secret key, out-decryption key, and ciphertext of our scheme are linearly related with the associated attribute set. For our scheme, the index size is related to the size of the keyword set, and the length of a trapdoor for one keyword is only the length of two elements in the group $\mathbb{G}$.

One can see that the performance of the following procedures is promising: key generation, user decryption, key update, and ciphertext update. Table IV demonstrates that the computational cost of generating the decryption key for a user

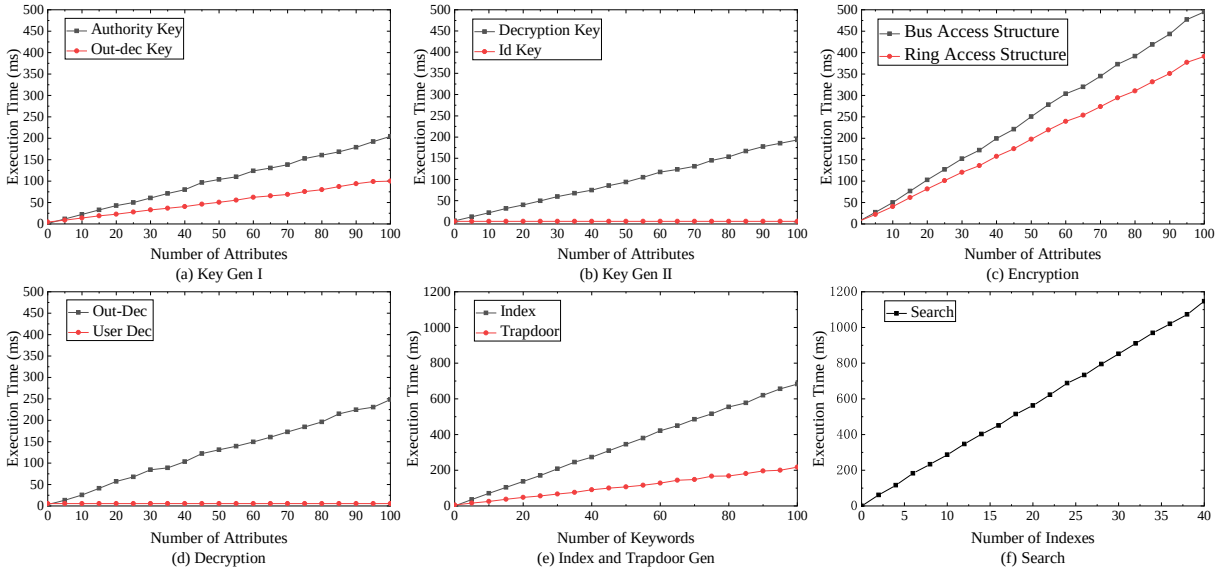


Fig. 4. Computational Overheads.

TABLE VI
OVERHEAD OF PEDERSEN ALGORITHM.

Node Number	(3,5)	(6,10)	(12,20)
Time	463.76 ms	1148.52 ms	2519.07 ms

in our scheme grows slowly with the size of the user attribute set. On the other hand, the fully-decryption overhead of data users in our scheme is only one exponentiation over $\mathbb{G}_T$. To update an attribute i of a user GID, the associated attribute authority who controls the attribute i costs one exponentiation over the group $\mathbb{G}$ to generate the attribute update key or the ciphertext update key, and one point multiplication to update its public key. After receiving the attribute update key, each non-revoked user requires one point multiplication to update its secret key (zero exponentiation, zero pairing operation), and the cloud server only needs to partially re-encrypt a ciphertext (two multiplications without exponentiations and pairing operations).

The superior performance of our ERS-ABE scheme has been justified by the theoretical analysis. To further demonstrate the overhead in terms of execution time, we conducted simulations and analyzed the results from the following two aspects. (1) Our scheme utilizes a permissioned blockchain to set up the system and generate public parameters by running the Pedersen secret sharing algorithm. We simulate the Pedersen algorithm on the most popular permissioned blockchain platform, Hyperledger Fabric. (2) The encryption algorithm-related simulations were conducted under a Python-based prototyping framework, Charm (version 0.50) [44], which is designed for the rapid construction of cryptographic schemes. The simulation environments of these two parts are different, as presented in Table V.

To share a parameter g^S, the Pedersen algorithm conducts three steps: (1) each node constructs a sub-share value s_{ij} and packs it into a transaction; (2) each node calculates the master

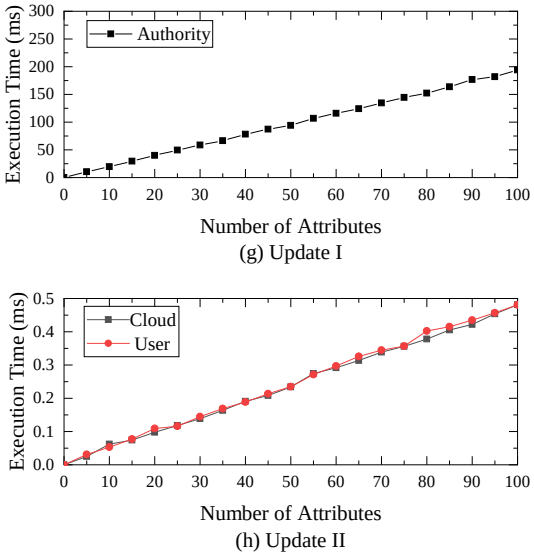


Fig. 5. Update Overheads.

share s_i and g^{s_i}, then packs g^{s_i} into a transaction to broadcast; and (3) the objective parameter g^S can be re-constructed by the Lagrange interpolation algorithm. Assume that the waiting time for generating a block can be ignored, which means that each transaction is immediately formed into a block as soon as it is submitted without the need of waiting for enough other transactions, we can get that the simulation results of the Pedersen algorithm are related to the number of nodes, as shown in Table VI (results are the average of 100 repetitions).

Our scheme's execution time simulation results are reported in Fig. 4 and Fig. 5. Specifically, Fig. 4. (a) is for the authority key and user out-decryption key generation; Fig. 4. (b) is for user id key and decryption key generation; Fig. 4. (c) is for encryption with bus access structures (only "AND" gates) and

with ring access structures (only "OR" gates); Fig. 4. (d) is for outsourced decryption and user decryption; Fig. 4. (e) is for index and trapdoor generation; and Fig. 4. (f) is for search. Fig. 5. (g) is for update in authority part; and Fig. 5. (h) is for update in cloud part and in user part. Each of the reported results is the average of 100 calculations.

Fig. 4. (a-b) indicate that the execution times of the three key generation algorithms grow linearly with the number of attributes, while the user identity key generation overhead is a constant (about 1.07 ms). We simulated the encryption algorithm with both the bus access structure and the ring access structure. The corresponding results are presented in Fig. 4. (c). Results for the outsourced decryption and the user decryption are illustrated in Fig. 4. (d), where one can observe that the full decryption at the user side in our scheme costs only 5.48 ms (not related to the number of attributes).

Fig. 4. (e) indicates that the time overheads of the index generation and token generation are linear with the number of keywords. More specifically, it costs 0.12s to generate an index containing 15 keywords and a search token containing 50 keywords. Regarding the search algorithm demonstrated in Fig. 4. (f), the simulation result only includes the matching of indexes and tokens but not the consensus process after the search. Assume that each index contains 20 keywords, we can get that it spends about 1.15s to match one keyword out of 400 keywords.

We divided the evaluation of our attribute update phase into three parts, namely authorities, cloud, and users as presented in Fig. 5. (g-h). It can be observed that the costs to the cloud and to the users are extremely low.

In summary, the theoretical analysis illuminates that our scheme performs superiorly in key generation, user decryption, and update at a higher security level. Moreover, the simulation results indicate that our ERS-ABE is practical as its time overhead is reasonable.

VIII. Conclusion

In this paper, we propose a blockchain-assisted efficient, revocable and searchable scheme (ERS-ABE) combing PEKS and MA-ABE to accomplish one-to-many access control and coarse-grained search for C-IoT. The coarse-grained keyword-based search can reduce the whole system's communication overhead and efficiently filter a large amount of data. Trusted token generation and search are realized by using a permissioned blockchain, which also helps to generate public parameters and achieve efficient user revocation. Besides, as the attribute update in our scheme can be accomplished by only partially changing the ciphertext, the computational overhead on the cloud can be further reduced. Due to the resource constraints of IoT devices, our scheme outsources the time-consuming pairing operations of decryption to a cloud server, which significantly reduces users' computational overhead. Our ERS-ABE scheme is proved to be selectively secure against RCCA and CKA. Simulation results indicate that replacing the central server with a blockchain does not bring much overhead to ERS-ABE in terms of computation time, demonstrating the feasibility and efficiency of ERS-ABE.

References

[1] S. Xu, G. Yang, Y. Mu, and X. Liu, "A secure iot cloud storage system with fine-grained access control and decryption key exposure resistance," *Future Generation Computer Systems*, vol. 97, pp. 284–294, 2019.

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

[3] M. Green, S. Hohenberger, B. Waters *et al.*, "Outsourcing the decryption of abe ciphertexts." in *USENIX security symposium*, vol. 2011, no. 3, 2011.

[4] C. Zuo, J. Shao, G. Wei, M. Xie, and M. Ji, "Cca-secure abe with outsourced decryption for fog computing," *Future Generation Computer Systems*, vol. 78, pp. 730–738, 2018.

[5] S. Liu, J. Yu, C. Hu, and M. Li, "Outsourced multi-authority abe with white-box traceability for cloud-iot," in *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 2020, pp. 322–332.

[6] J. Yu, S. Liu, S. Wang, Y. Xiao, and B. Yan, "Lh-absc: A lightweight hybrid attribute-based signcryption scheme for cloud-fog-assisted iot," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7949–7966, 2020.

[7] S. Liu, L. Chen, H. Wang, S. Fu, and L. Shi, "O3hsc: Outsourced online/offline hybrid signcryption for wireless body area networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022, doi:10.1109/TNSM.2022.3153485.

[8] C. Feng, K. Yu, M. Aloqaily, M. Alazab, Z. Lv, and S. Mumtaz, "Attribute-based encryption with parallel outsourced decryption for edge intelligent iov," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 784–13 795, 2020.

[9] N. Chen, J. Li, Y. Zhang, and Y. Guo, "Efficient cp-abe scheme with shared decryption in cloud storage," *IEEE Transactions on Computers*, vol. 71, no. 1, pp. 175–184, 2020.

[10] Z. Wang, D. Huang, Y. Zhu, B. Li, and C.-J. Chung, "Efficient attribute-based comparable data access control," *IEEE Transactions on computers*, vol. 64, no. 12, pp. 3430–3443, 2015.

[11] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2011, pp. 568–588.

[12] Y. Rahulamathavan, S. Veluru, J. Han, F. Li, M. Rajarajan, and R. Lu, "User collusion avoidance scheme for privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2939–2946, 2015.

[13] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 315–332.

[14] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, "Phoabe: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot," *Computer Networks*, vol. 133, pp. 141–156, 2018.

[15] S. Liu, J. Yu, C. Hu, and M. Li, "Traceable multiauthority attribute-based encryption with outsourced decryption and hidden policy for ciot," *Wireless Communications and Mobile Computing*, vol. 2021, 2021.

[16] P. Datta, I. Komargodski, and B. Waters, "Decentralized multi-authority abe for dnfs from lwe," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 177–209.

[17] C.-I. Fan, V. S.-M. Huang, and H.-M. Ruan, "Arbitrary-state attribute-based encryption with dynamic membership," *IEEE Transactions on Computers*, vol. 63, no. 8, pp. 1951–1961, 2013.

[18] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," in *IMA international conference on cryptography and coding*. Springer, 2009, pp. 278–300.

[19] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM symposium on information, computer and communications security*, 2010, pp. 261–270.

[20] H. Cui, R. H. Deng, Y. Li, and B. Qin, "Server-aided revocable attribute-based encryption," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 570–587.

[21] H. Zhong, W. Zhu, Y. Xu, and J. Cui, "Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage," *Soft Computing*, vol. 22, no. 1, pp. 243–251, 2018.

[22] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[23] R. Guo, G. Yang, H. Shi, Y. Zhang, and D. Zheng, "O3-r-cp-abe: An efficient and revocable attribute-based encryption scheme in the cloud-assisted iomt system," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8949–8963, 2021.

[24] S. Wang, J. Ye, and Y. Zhang, "A keyword searchable attribute-based encryption scheme with attribute update for cloud storage," *PloS one*, vol. 13, no. 5, 2018.

[25] S. Liu, J. Yu, Y. Xiao, Z. Wan, S. Wang, and B. Yan, "Bc-sabe: Blockchain-aided searchable attribute-based encryption for cloud-iot," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7851–7867, 2020.

[26] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 506–522.

[27] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *IEEE INFOCOM 2014-IEEE conference on computer communications*. IEEE, 2014, pp. 522–530.

[28] K. Liang and W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1981–1992, 2015.

[29] J. Li, X. Lin, Y. Zhang, and J. Han, "Ksf-oabe: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 715–725, 2016.

[30] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "Fs-peks: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1019–1032, 2019.

[31] H. Shi, S. Wang, Q. Hu, X. Cheng, J. Zhang, and J. Yu, "Fee-free pooled mining for countering pool-hopping attack in blockchain," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, July/August 2021.

[32] M. Xu, F. Zhao, Y. Zou, C. Liu, X. Cheng, and F. Dressler, "Blown: A blockchain protocol for single-hop wireless networks under adversarial sinr," 2021. [Online]. Available: https://arxiv.org/abs/2103.08361

[33] M. Xu, C. Liu, Y. Zou, F. Zhao, J. Yu, and X. Cheng, "wchain: A fast fault-tolerant blockchain protocol for multihop wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6915–6926, October 2021.

[34] M. Xu, S. Liu, D. Yu, X. Cheng, S. Guo, and J. Yu, "Cloudchain: A cloud blockchain using shared memeory consensus and rdma," *IEEE Transactions on Computers*, 2022.

[35] C. Liu, M. Xu, H. Guo, X. Cheng, Y. Xiao, D. Yu, B. Gong, A. Yerukhimovich, and S. andWeifeng Lv, "Tokoin: A coin-based accountable access control scheme for internet of things," 2020. [Online]. Available: https://arxiv.org/abs/2011.04919

[36] C. Liu, H. Guo, M. Xu, S. Wang, D. Yu, J. Yu, and X. Cheng, "Extending on-chain trust to off-chain – trustworthy blockchain data collection using trusted execution environment (tee)," *IEEE Transactions on Computers*, 2022. [Online]. Available: https://arxiv.org/abs/2106.15934

[37] Y. Zhang, C. Xu, J. Ni, H. Li, and X. S. Shen, "Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019, doi:10.1109/TCC.2019.2923222.

[38] Z. Guan, N. Wang, X. Fan, X. Liu, L. Wu, and S. Wan, "Achieving secure search over encrypted data for e-commerce: A blockchain approach," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–17, 2020.

[39] X. Li and M. Tan, "Electronic certificate sharing scheme with searchable attribute-based encryption on blockchain," in *Journal of Physics Conference Series*, vol. 1757, no. 1, 2021, p. 012161.

[40] S. Niu, L. Chen, and W. Liu, "Attribute-based keyword search encryption scheme with verifiable ciphertext via blockchains," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9, 2020, pp. 849–853, doi:10.1109/ITAIC49862.2020.9338962.

[41] Mamta, B. B. Gupta, K.-C. Li, V. C. M. Leung, K. E. Psannis, and S. Yamaguchi, "Blockchain-assisted secure fine-grained searchable encryption for a cloud-based healthcare cyber-physical system," *IEEE/CAA Journal of Automatica Sinica*, pp. 1–14, 2021, doi:10.1109/JAS.2021.1004003.

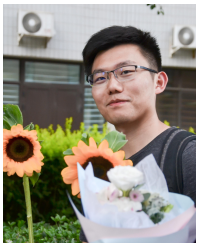
[42] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1991, pp. 522–526.

[43] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *Journal of Systems and Software*, vol. 83, no. 5, pp. 763–771, 2010.

[44] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

Jiguo Yu received the Ph.D. degree from the School of Mathematics, Shandong University, in 2004. He became a Full Professor in the School of Computer Science, Qufu Normal University, Shandong, China, in 2007, and currently is a Full Professor at the Qilu University of Technology (Shandong Academy of Sciences), Jinan, Shandong China. His main research interests include blockchain, IoT security, privacy-aware computing, distributed computing, and graph theory. He is a Fellow of IEEE, a member of ACM, and a Senior Member of China Computer Federation (CCF).

Suhui Liu received her B.S. degree and M.S. degree in Computer Science, Qufu Normal University, Shandong, China in 2018 and 2021, respectively. She is currently pursuing a Ph.D. in the School of Cyber Science and Engineering, Southeast University, Jiangsu, China. Her main research interests include cloud-assisted IoT data security, functional cryptography, and blockchain technology.



Minghui Xu received the PhD in Computer Science from The George Washington University, Washington DC, USA, in 2021, and the BS degree in Physics from the Beijing Normal University, Beijing, China, in 2018. Dr. Xu is currently an Assistant Professor in the School of Computer Science and Technology, Shandong University, China. His research focuses on blockchain, distributed computing, and applied cryptography.

Hechuan Guo is a PhD candidate in Computer Science at Shandong University, Qingdao, China. He received his BS Degree in Computer Science in 2017 and MS degree in Engineering in 2020, both from Beijing Normal University, Beijing, China. His current research focuses on blockchain, consensus algorithms, and applied cryptography.

Fangtian Zhong holds a Ph.D. in Computer Science from George Washington University and completed his undergraduate study in Software Engineering from Northeast Normal University. He is currently a postdoctoral scholar in Information Science and Technology at The Pennsylvania State University. His main research interests are based on program analysis and machine learning for solving problems in software security, and system security. He has also published some papers in prestigious journals and conferences, i.e., IEEE Transactions on Computers, IEEE Internet of Things Journal, ACM CIKM.

Wei Cheng received his Ph.D from the George Washington University. His research interests span the areas of smart city, edge computing, and cybersecurity. In particular, he is working on localization in GPS-denied environments, HCI and Security, public safety networks, underwater networks, and RFID systems for smart transportation. He served as the technical program committee chair/member and the editorial board member for several top international conferences and journals, respectively.