Achieving Fair and Efficient Revocable Access Control for IIoT Data Sharing: A Blockchain-Enabled Approach

Bei Li, Hong Zhong, Jing Zhang, Qingyang Zhang, Jiaxin Li, and Jie Cui, *Senior Member, IEEE*

***Abstract*—With the advancement of computing and communication technologies, Industrial Internet of Things (IIoT) has emerged accordingly. In IIoT environments, efficient data sharing is achieved through collaboration among end devices, edge servers, and cloud servers. However, ensuring the security, efficiency, and fairness of service data access for end devices remains a significant challenge. To address this, we propose a fair and efficient revocable access control scheme based on blockchain. The proposed scheme leverages smart contracts to establish a fair payment mechanism, ensuring fairness for IIoT data sharing. In addition, a proxy-assisted decryption approach is employed to minimize the decryption overhead on end devices. Moreover, the scheme supports efficient user revocation without requiring updates to the private keys of end users. This enhances the overall security and usability of the system. Finally, a thorough security and performance analysis indicate that the proposed scheme fits well within IIoT scenarios.**

***Index Terms*—Attribute-based encryption (ABE), data sharing, fair payment, user revocation, verifiable outsourced decryption.**

I. Introduction

With the development of cloud computing and communication technologies, the Industrial Internet of Things (IIoT) has been widely adopted. In the IIoT, devices are characterized by limited computational resources and large quantities, making traditional centralized cloud-based data-sharing architectures increasingly inadequate for meeting the demands of low latency and high quality of service. An edge computing-based IIoT architecture can facilitate data transmission between cloud servers and industrial devices, thereby reducing computational and transmission delays. However, this architecture raises a number of security concerns, particularly in access control and trustworthy distribution [1], [2].

First, it is imperative to preserve the confidentiality of service data and support flexible access control during sharing and transmission. To this end, Attribute-based encryption (ABE) has emerged as a widely used technique due to its ability of fine-grained access control [3]. Nevertheless, in practical deployments, if an end device is compromised, the keys it holds may be misused, thereby endangering the security of the data and models. Consequently, it is essential for the system to support dynamic user revocation. Devices in the IIoT often have limited resources, and cannot conduct too many sophisticated processes. Therefore, the computational cost of the device should be considered when achieving revocation.

By transferring the time-consuming work to a proxy server, outsourced ABE reduces the computational overhead on the data user side [4], [5]. However, outsourced ABE poses data security and online payment issues. For example, a user may pay for an outsourced decryption service, but a proxy server corrupts the integrity of the data during data computation or provides an incorrect result. Malicious users can even obtain free outsourced services by accusing the proxy server of providing incorrect results. Consequently, obtaining fair payments in outsourced computing is challenging [6].

A possible solution involves introducing a trusted third-party institution as an intermediary to enable fair payments [7], [8]. However, conventional payment methods suffer from several limitations. First, the requirement that trusted third parties always participate in an exchange can lead to a payment bottleneck. Second, some users who value their privacy are concerned about the transparency of transactions during the payment process, even if a trusted authority operates fairly.

Blockchain has recently received considerable attention owing to its decentralization and self-control features [9]. Blockchain-based platforms, such as Ethereum, can operate autonomously and independently of any single party. Both parties can use smart contracts to achieve fair payments; the smart contracts verify the results of the outsourced computation to ensure fair transaction execution. Although blockchain-based fair payments have been used in public data audits [10], [11], searchable encryption [12], [13], and machine learning [14], [15], blockchain technology is rarely applied in a universal and fair manner for outsourced computing in the IIoT environment.

Received 18 April 2025; revised 22 May 2025; accepted 23 June 2025. Date of publication 26 June 2025; date of current version 25 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62272002, Grant 62202008, Grant 62202005, and Grant 62472003; in part by the Natural Science Foundation of Anhui Province, China under Grant 2408085JX010; and in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2022-049. *(Corresponding author: Hong Zhong.)*

Bei Li, Hong Zhong, Jing Zhang, Qingyang Zhang, and Jie Cui are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, and the Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China (e-mail: zhongh@ahu.edu.cn).

Jiaxin Li is with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, and the Security Research Institute, New H3C Group, Anhui University, Hefei 230088, Anhui, China (e-mail: li.jiaxin@h3c.com).

Digital Object Identifier 10.1109/JIOT.2025.3583317

A. Motivation and Contribution

Currently, the process of data sharing in IIoT should satisfy the following security requirements. The first security requirement involves quickly repairing the system and revoking corresponding users in the event of a data leakage. The second security requirement concerns maintaining data integrity while ensuring secure and efficient data sharing. The third security requirement is the ability to make fair payments for resource-constrained devices in IIoT contexts when adopting data outsourcing services. Based on these requirements, we develop an efficient and fair revocable access control scheme to enable data sharing for IIoT. To provide revocable access control with data integrity checks, we combine revocable ABE with verifiable outsourced computation, and then leverage blockchain to enable fair payment in outsourced computation. The contributions are as follows.

1) A server-aided revocable ABE scheme with data integrity and decryption verification is proposed to alleviate the security and privacy challenges of resource-restricted devices in data sharing. The proposed scheme employs proxy servers to offer data users outsourced computing and achieve efficient user revocation. Data integrity and decryption verification address the issue of incorrect computing results generated by outsourced computing.
2) A blockchain-aided data sharing for the IIoT context is designed to address the fair payment issue caused by the deployment of proxy servers. Blockchain technology is integrated with the proposed revocable ABE scheme to ensure fair and efficient data sharing in resource-restricted contexts while addressing the three security requirements described above.
3) A formal security proof and detailed performance analyses are presented to validate the security and usability of the designed system. The results show that the proposed scheme achieves both secure and efficient, and the simulated smart contract results further highlight the feasibility and utility of the blockchain-integrated data sharing solution.

B. Related Work

To achieve the verifiability of outsourced computation results, Lai et al. [16] first proposed a verifiable outsourced ABE model, and then provided a particular structure based on Green et al.'s approach [17]. However, the recommended solution demands additional storage and outsourced computation expenditure due to the extra ciphertext generated during encryption. Lin et al. [18] developed an ABE scheme with verifiable outsourced decryption using the key encapsulation technique. The solution combines symmetric encryption technology and a commitment technology to optimize the computation overhead of encryption and outsourced decryption. Ning et al. [19] developed an auditable outsourced ABE technique that also accomplished user anonymity and a restricted number of fine-grained access while resisting key disclosure concerns. Li et al. [20] constructed a verifiable outsourced decryption scheme, in which a ciphertext embedding a redundant random message is generated during encryption to enable verification of the decryption result by both authorized and unauthorized users. Miao et al. [21] proposed a verifiable outsourced ABE scheme, where the verifiable tag mechanism is utilized to achieve the outsourced result verification. Sun et al. [22] introduced a verifiable data sharing scheme based on a commitment mechanism, and leveraged a proxy server to convert attribute-based ciphertexts into public-key ciphertexts. Although the aforementioned approaches all take into account the verifiability of computing results in access control, they do not achieve the malicious user revocation and the fair payment in outsourced computing.

Cui et al. [4] introduced a proxy-assisted user revocation scheme, Qin et al. [5] introduced a dual-server-assisted revocation technique to address decryption key leakage issue. Guo et al. [23] leveraged blockchain technology to tackle the key escrow problem in ABE scheme, and introduced a group manager to manage users and support revocation. In [24], the cloud server updates ciphertexts associated with the data owner to achieve user revocation. Meng and Cheng [25] designed an efficient tracing mechanism for proxy-assisted revocation schemes, enabling user revocation by updating the proxy server's transformation key. To address the issues of user revocation and verifiability of outsourced decryption, Ge et al. [26] proposed a revocable ABE scheme, in which user revocation is achieved by updating the ciphertext access policy on the cloud server. Additionally, verifiability of the decryption result is ensured by generating redundant ciphertext components during encryption. Chen et al. [27] proposed a revocable access control scheme with support for decryption result verification. It ensures verifiability by generating ciphertexts containing redundant messages and employing a commitment mechanism, while user revocation is achieved by updating the ciphertext access policy. However, this approach introduces additional overhead on the user side. Liu et al. [28] also employed a commitment mechanism to achieve verifiability of outsourced computation results, and utilized puncturable encryption to enable data file erasure.

Fair payment has been proposed and intensively researched in order to alleviate the trust problem in outsourced computing and preserve the interests of the participants [29]. Chen et al. [7] developed a fair payment approach addressing the trust problem of distributed outsourced computing. Because the method is based on the regular electronic payment system, a semi-trusted third party is required to complete the transaction, which may be a payment bottleneck. Cui et al. [30] proposed a blockchain-assisted method for outsourced functional encryption that provided fair payment while ensuring the verifiability of the outsourced computing results. Lin et al. [6] used blockchain and cryptography technologies to create an optimal fair payment system. Because the approach does not reliant on zero-knowledge proof, it performs better than the prior solutions. Liu et al. [31] proposed a fair hybrid data transaction system used the blockchain, in which cloud servers offer data storage and blockchain facilitates fair and transparent data transactions. Ge et al. [32] proposed an efficient verifiable outsourced decryption scheme, and blockchain only requires to perform a small amount of meta computation to validate

TABLE I
COMPARISON OF THE RELATED WORK

Schemes	Verifiability	OutDec	Rev	FairPay
[16], [18]–[22]	Yes	Yes	No	No
[4], [5], [23], [25]	No	Yes	Yes	No
[26]–[28]	Yes	No	Yes	No
[6], [7], [29], [30]	No	Yes	No	Yes
[32]	Yes	Yes	No	Yes
[33], [34]	No	No	No	Yes
[35]	Yes	Yes	Yes	Yes
Our	Yes	Yes	Yes	Yes

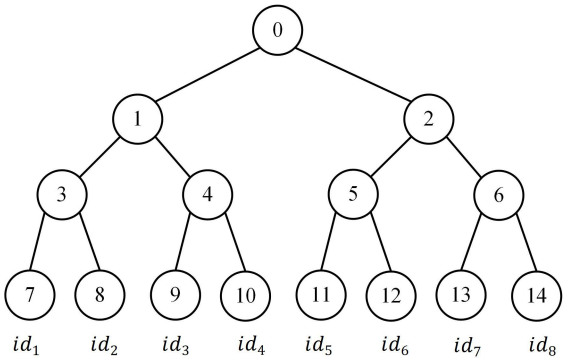


Fig. 1. Example of binary tree with eight users.

the correctness of the outsourced computation results, while ensuring fair payment between proxy servers and data users. Feng et al. [33] constructed a fair data trading scheme based on blockchain, and ABE is adopted to enforce fine-grained access control. Merkle tree and commitment techniques are employed to ensure data integrity verification. Chen et al. [34] also employed Merkle trees to implement fair data trading based on a trusted third party and blockchain. However, they do not consider user revocation or proxy decryption. Ruan et al. [35] proposed a blockchain based revocable outsourcing access control scheme that achieves fairness between outsourcing servers and data users. The comparison is shown in Table I, where the OutDec stands for outsourced computation, Rev means revocation, and FairPay means fair payment.

II. PRELIMINARIES

A. Revocation Mechanism

This tree structure was introduced in [36] to achieve efficient revocation. Here we briefly recall the revocation techniques. Given a binary tree BT with at least N leaves, and the leaf node is associated with the user. For a node θ, path(θ) is the path from θ to the root *root*. And cover(rl) denotes a minimum nodes set, which can cover all the nonrevoked users. For any nonrevoked user associated with leaf node θ_i, there must exist one node $x_i = \text{path}(\theta_i) \cap \text{cover}(rl)$.

As shown in Fig. 1, the tree has eight leaves related to the users. Suppose the current revocation list $rl = \{\text{id}_1, \text{id}_3\}$, then $\text{cover}(rl) = \{8, 10, 2\}$ and $\text{path}(\text{id}_5) = \text{path}(11) = \{11, 5, 2, 0\}$. For the nonrevoked user id_5, he can find a node $x_i = \text{cover}(rl) \cap \text{path}(\text{id}_5) = \{2\}$. Note that the minimum set outputted by cover(rl) is actually the same as that of *KUNodes* in [4], the difference is we use the minimum set to generate the updated ciphertexts instead of the user's secret key.

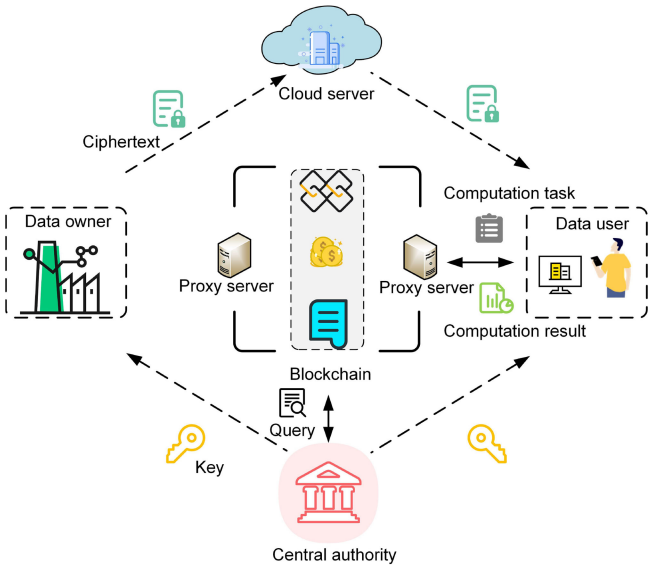


Fig. 2. System model.

B. Commitment Scheme

A commitment scheme *Com* consists a pair of algorithms $\text{Com} = (\text{Init}, C, V)$. The *Com.Init* initializes the commitment scheme and generates parameters *cp*. When a sender wants to commit to a message m, he generates a commitment $cm = \text{Com}.C(m, r)$ and returns it to the receiver. The receiver can verify the commitment through $\text{Com}.V(cm, m, r) \stackrel{?}{=} 1$. The receiver accepts the commitment if $\text{Com}.V(cm, m, r) = 1$. The correctness of the commitment requires that $\forall m \in M$ if $cm \leftarrow \text{Com}.C(m, r)$ then $\Pr[\text{Com}.V(cm, m, r) = 1] = 1$. And it needs to satisfy the following properties.

Hiding: It requires that the commitment reveals nothing about the message m.

Binding: It requires that the commitment value computed by the sender can not be opened as two inconsistent messages. This implies it is infeasible for an adversary to choose two different messages $m_0 \neq m_1$ and the commitment value $\text{Com}.C(m_0, r) = \text{Com}.C(m_1, r) = cm$ and $\text{Com}.V(cm, m_0, r) = \text{Com}.V(cm, m_1, r) = 1$.

III. SYSTEM MODEL

This section presents the architecture of the proposed blockchain-aided data sharing framework. Furthermore, the definition of server-aided revocable ABE with verifiable decryption (SR-ABE-VD) is presented. After that, we present the security definition of SR-ABE-VD.

A. System Architecture

As shown in Fig. 2, SR-ABE-VD is composed of six components: central authority, data owner, data user, cloud server, proxy server and blockchain.

1) *Central Authority:* The completely trusted central authority is responsible for the system initialization and the user key generation.
2) *Data Owner:* It is the owner of the shared digital assets. To make better use of the data, it will encrypt and upload the data to the cloud for analysis by the appropriate data user.
3) *Data User:* It is the one who uses the data. When it needs to access the data, the corresponding ciphertext is retrieved from the cloud, and it then outsources part of the decryption procedure to the proxy server to decrease computational cost. It will pay the associated computation fee if the proxy server produces the correct computation result.
4) *Cloud Server:* It is in charge of maintaining shared data and contains a significant number of storage and computational power.
5) *Proxy Server:* It has sufficient computing capability to handle outsourced computing tasks. It is semi-trusted and may deliver inaccurate or partial computing results. When the proxy server honestly executes the outsourced computing tasks published by the data user, it will earn the relevant service charge, and vice contrary, it will be fined.
6) *Blockchain:* It is a decentralized and reliable platform maintained by a large number of record nodes. Any blockchain supporting smart contracts can be integrated into the proposed system. Smart contracts are deployed on the blockchain and performed automatically when particular events occur (e.g., data users submitting computing tasks), and can be used to ensure fair payments between data users and proxy servers. Both parties must pay a deposit when initiating a task, data users when submitting tasks and proxy servers when accepting them. Upon mutual agreement, the smart contract transfers the payment to the proxy server. In case of disputes, it resolves them fairly. If the proxy returns incorrect results, its deposit is compensated to the user. If the user maliciously accuses the proxy, their deposit is transferred to the proxy.

B. Definition

Next, based on the system architecture, we give the definition of the proposed SR-ABE-VD.

1) *Setup(1^λ) $\rightarrow$ (msk, pp):* Taking as input λ, the central authority executes this algorithm to initialize the system and outputs the public parameters pp and the master secret key msk.
2) *UKGen(pp, id) $\rightarrow$ ($pk_{\mathrm{id}}, sk_{\mathrm{id}}$):* Taking as input pp, the central authority executes this algorithm to generate the public secret key pair $(pk_{\mathrm{id}}, sk_{\mathrm{id}})$ and (pk_{pxy}, sk_{pxy}) for the proxy server.
3) *PxyKGen($msk, U_{\mathrm{id}}, pk_{\mathrm{id}}, pk_{pxy}$) $\rightarrow$ $PxyK_{\mathrm{id}}$:* Taking as input msk, a set of attributes U_{id} and the proxy server's public key pk_{pxy}, the central authority executes this algorithm to generate the proxy decryption key $PxyK_{\mathrm{id}}$ for the proxy server.
4) *Encrypt($pp, (\mathcal{M}, \pi)$, msg, rl) $\rightarrow$ CT_{rl}:* Taking as input pp, an access structure $(\mathcal{M}, \pi)$, a message msg and a revocation list rl, the data owner executes this algorithm to generate the ciphertext CT_{rl}.
5) *CTUpd(CT_{rl}, rl') $\rightarrow$ $CT_{rl'}$:* Taking as input the ciphertext CT_{rl} associated with the revocation list rl and an updated revocation list rl', the central authority runs this algorithm to update the ciphertext. The updated ciphertext $CT_{rl'}$ will replace the old ciphertext CT_{rl} and be stored in the cloud server.
6) *PxyDec(CT_{rl}, id, $PxyK_{\mathrm{id}}$) $\rightarrow$ CT':* Taking as input the ciphertext CT_{rl}, the user identity id and the proxy decryption key $PxyK_{\mathrm{id}}$ associated with user id, the proxy server generates the partially decrypted ciphertext CT'_{rl}.
7) *Decrypt($CT_{rl}, CT'_{rl}, sk_{\mathrm{id}}$) $\rightarrow$* msg*:* On input the original ciphertext CT_{rl}, a partially decrypted ciphertext CT'_{rl} and the user's secret key sk_{id}, the data user executes this algorithm to recover the underlying message msg or an invalid symbol $\perp$. The data user accepts the message if it can pass the decryption verification.
8) *Audit($CT_{rl}, CT'_{rl}, msk, pk_{\mathrm{id}}$) $\rightarrow$ $\{0, 1\}$:* This algorithm takes pk_{id} of the data user id, the master secret key msk, the original ciphertext CT_{rl} and a partially decrypted ciphertext CT'_{rl} as input, and the central authority run it to determine whether the proxy server returns an incorrect result.
9) *Revoke(*id, rl) $\rightarrow$ rl':* Taking as input rl and a revoked user identity id, the central authority executes this algorithm to update the revocation list $rl \rightarrow rl'$.

Note that in the SR-ABE-VD, the proxy decryption key does not need to be updated. To enforce forward and backward security, we utilize the ciphertext update mechanism. Concretely, encryption consists of two steps: one generates access policy-related ciphertext and the other generates ciphertext on the revocation state. When a revocation event occurs, the central authority needs to modify rl and the revocation-related ciphertext components. To achieve verifiable decryption, a commitment to the message is generated during the encryption. This commitment can be utilized to verify the integrity of the message when the data user executes the ciphertext decryption.

Definition 1 (Correctness): A SR-ABE-VD scheme $\Pi_{SR-ABE-VD}$ is correct, implying that for any λ and all msg selected from the message space, if the attributes of id satisfy the access policy and the user is not revoked, we have that $Decrypt(CT_{rl}, CT'_{rl}, sk_{\mathrm{id}}) = \mathrm{msg}$ where $CT'_{rl} \leftarrow PxyDec(Enc(pp, (\mathcal{M}, \pi), \mathrm{msg}, rl), \mathrm{id}, PxyK_{\mathrm{id}})$.

C. Security

Considering the security of SR-ABE-VD, we expect the scheme to meet two requirements: indistinguishability against chosen plaintext attacks (indistinguishability under chosen-plaintext attacks (IND-CPA)) and ciphertext decryption verifiability.

To prove the IND-CPA security, we reduce it to the IND-CPA security of SR-ABE-VD without verifiable decryption (SR-ABE) and the computationally binding property of the

underlying commitment scheme. And the IND-CPA security of the SR-ABE can be described by a game as follows.

Setup: $\mathcal{B}$ runs *setup* to generate *pp*, *msk*, $rl \in \emptyset$ for revocation, (pk_{pxy}, sk_{pxy}) for a proxy server and a user tree *BT*. $\mathcal{B}$ returns *pp* to $\mathcal{A}$ and keeps *msk* privately.

Phase 1: The simulator generates two empty list L_1, L_2 and $\mathcal{A}$ can adaptively queries to $\mathcal{B}$.

1) *UKGen(*id*):* $\mathcal{A}$ requests a private key query with an identity id, $\mathcal{B}$ executes the *UKGen* algorithm and inserts $(\text{id}, pk_{\text{id}}, sk_{\text{id}})$ into L_1.
2) *PxyKGen(*id, U_{id}*):* $\mathcal{A}$ requests a proxy decryption key with a set of attributes U_{id} and an identity id, $\mathcal{B}$ executes the *PxyKGen* algorithm to obtain the attribute related proxy decryption key $PxyK_{\text{id}}$. Then the tuple $(\text{id}, U_{\text{id}}, PxyK_{\text{id}})$ will be inserted into L_2.
3) *Corrupt(*id*):* The simulator inserts $(\text{id}, pk_{\text{id}}, sk_{\text{id}})$ into L_1 if the entry does not exist, otherwise it aborts and returns $\perp$.
4) *Corrupt(*$PxyK_{\text{id}}$*):* $\mathcal{B}$ inserts $(\text{id}, U_{\text{id}}, PxyK_{\text{id}})$ into L_2 if the entry does not exist, otherwise it aborts and returns $\perp$.
5) *Revocation(*id*):* $\mathcal{A}$ requests with an identity id, $\mathcal{B}$ executes the *Revoke* algorithm and returns an updated revocation list rl'.

Challenge: The adversary $\mathcal{A}$ chooses two equal length messages $\text{msg}_0, \text{msg}_1$ and an access structure $(\mathcal{M}^*, \pi^*)$ with the following restrictions.

1) For any id^* has been queried to *UKGen*(id) and $(\text{id}^*, U_{\text{id}^*})$ has been queried to the *PxyKGen*$(\text{id}, U_{\text{id}})$. If the attribute set U_{id^*} satisfies $(\mathcal{M}^*, \pi^*)$, the *revocation*(id) needs to be queried.
2) For a nonrevoked user id^* whose attribute set U_{id^*} satisfies the access structure $(\mathcal{M}^*, \pi^*)$, then the id^* should not be queried to the *UKGen*(id) previously.

$\mathcal{B}$ chooses $b \in_R \{0, 1\}$ and runs the *Encrypt* on message msg_b to obtain the challenge ciphertext CT^*_{rl}. Finally, CT^*_{rl} will be sent to adversary $\mathcal{A}$.

Phase 2: $\mathcal{A}$ requests queries with the same restrictions as in *Phase* 1.

Guess: $\mathcal{A}$ outputs a bit b', it wins at the condition that $b' = b$. And its advantage in winning the game is $Adv = |\Pr[b = b'] - 1/2|$.

Definition 2: An SR-ABE scheme satisfies the IND-CPA security if no efficient $\mathcal{A}$ can win the game defined above with a non-negligible advantage. The selective security can be achieved by putting a *Init* step before *Setup* step where $\mathcal{A}$ outputs an access structure.

Definition 3: We say that the proposed SR-ABE-VD is selective secure if the SR-ABE is selectively IND-CPA security and the underlying commitment scheme satisfies computationally hiding property.

The ciphertext decryption verifiability of the proposed SR-ABE-VD can be captured by the following game.

1) *Setup:* The simulator $\mathcal{B}$ obtains *pp* and *msk*, and forwards *pp* to the adversary $\mathcal{A}$.
2) *Phase 1:* $\mathcal{A}$ can query to $\mathcal{B}$ as in the IND-CPA security game.
3) *Challenge:* $\mathcal{A}$ queries on a challenge value $(\mathcal{M}^*, \pi^*)$ and a message msg^*, the simulator $\mathcal{B}$ executes *Encrypt* algorithm to obtain (CT^*_{rl}, cm^*) and sends them to the adversary.
4) *Phase 2:* $\mathcal{A}$ makes queries as in Phase 1.
5) *Output:* $\mathcal{A}$ submits a partially decrypted ciphertext CT^*_T. The adversary wins if $\text{Dec}(CT^*_{rl}, CT^*_T, sk^*_{\text{id}}) \notin \{\text{msg}^*, \perp\}$.

Definition 4: An SR-ABE-VD scheme achieves the decryption verifiability if no adversary can win the above verifiability game with a non-negligible advantage.

IV. Main Construction

In this section, a concrete construction of the proposed SR-ABE-VD is presented, then we integrate it with the blockchain to implement the fair payment.

A. Generic Construction

Setup(1^λ): This algorithm calls a group generator $Gen(\lambda)$ to generate a bilinear group description $\mathcal{G} = (p, G, G_T, e)$, then it chooses $g, u, w, h, v \in G$ and $\alpha \in Z_p$ and sets as the public parameters $pp = (\mathcal{G}, g, u, w, h, v, e(g, g)^\alpha)$ and the master secret key $msk = \alpha$. Finally, it initializes an empty revocation list *rl* and a binary tree *BT* with at least *N* leaf nodes, where *N* denotes the maximal number of system users. All the nodes *i* of the *BT* are embedded with a tuple (y_i, g_{y_i}), where $g_{y_i} = g^{y_i}$ is public and $y_i \in Z_p$ need to be kept privately.

UKGen(*pp*, id): The central authority takes as input the public parameters *pp* and a user identity id, it chooses a hash function $H : \{0, 1\}^* \to Z_p$ and computes $x_{\text{id}} = H(\text{id})$ and sets $(pk_{\text{id}} = g^{x_{\text{id}}}, sk_{\text{id}} = x_{\text{id}})$ as the public secret key pair of user id. And this algorithm can also generate the key pair $(pk_{pxy} = g^{x_{pxy}}, sk_{pxy} = x_{pxy})$ for the proxy server.

PxyKGen(*msk*, U_{id}, pk_{id}, pk_{pxy})*:* The central authority takes as input the master secret key *msk*, a set of attributes $U_{\text{id}} = \{U_1, U_2, \ldots, U_k\}$, the user's public key pk_{id} and the proxy server's public key pk_{pxy}, it selects $\beta, r, r_1, r_2, \ldots, r_k \in Z_p$ and computes

$$K_0 = g^{\beta r} pk_{\text{id}}^{\alpha} pk_{pxy}^{\beta} w^r, K_1 = g^r, K_2 = g^\beta$$
$$\left\{K_{\tau,2} = g^{r_\tau}, K_{\tau,3} = \left(u^{U_\tau} h\right)^{r_\tau} v^{-r}\right\}_{\tau \in \{1,2,\ldots,k\}}.$$

Then it fetches the leaf node θ_i associate with the user id and retrieves $g_{y_{\theta_i}}$ from the leaf node. Then it chooses $b \in Z_p$ to compute $(D = g^{\beta r} g_{y_{\theta_i}}^b, E = g^b)$. Finally, the proxy decryption key for the user id is set to be $PxyK_{\text{id}} = (K_0, K_1, K_2, D, E, \{K_{\tau,2}, K_{\tau,3}\}_{\tau \in \{1,2,\ldots,k\}})$.

Encrypt(*pp*, $(\mathcal{M}, \pi)$, msg, *rl*): The data owner takes as input the public parameters *pp*, the message msg, a revocation list *rl* and an access structure $(\mathcal{M}, \pi)$ satisfies $\mathcal{M} \in Z_p^{l_1 \times l_2}, \pi : [l_1] \to Z_p$, where $[l_1]$ denotes $1, 2, \ldots, l_1$. It chooses the column vector $\vec{v} = (s, v_2, v_3, \ldots, v_{l_2})$ and computes the shares $\{\lambda_i = \mathcal{M}_i \vec{v}\}_{i \in [l_1]}$. Then it chooses $r_m, t_1, t_2, \ldots, t_{l_1} \in Z_p$ and two hash function $H_1 : G_T \to Z_p, H_2 : \{0, 1\}^* \to Z_p$ to compute

$$C = (\text{msg} || r_m) \oplus H_1\left(e(g, g)^{\alpha s}\right)$$
$$C_0 = g^s, cm = g^{H_2(\text{msg})} h^{H_2(r_m)}$$
$$\left\{C_{\tau,1} = w^{\lambda_\tau} v^{t_\tau}, C_{\tau,2} = \left(u^{\pi(\tau)}\right)^{-t_\tau}, C_{\tau,3} = g^{t_\tau}\right\}_{\tau \in [l_1]}.$$

Let cover(rl) be the minimum cover set of the revocation list rl, then for all nodes $j \in \text{cover}(rl)$ it computes $C_j = g^s_{y_j}$. Finally, it sets $CT_{rl} = ((\mathcal{M}, \pi), C, C_0, cm, C_{\tau,1}, C_{\tau,2}, C_{\tau,3_{\tau\in[l_1]}}, \{C_j\}_{j\in\text{cover}(rl)}, rl)$ as the final ciphertext.

$CTUpd(CT_{rl}, rl')$: The central authority runs this algorithm to update the ciphertext. Given the current revocation list rl' and the old revocation list rl, let $cover(rl')$ denotes the minimum cover set associated with rl'. For node $j' \in \text{cover}(rl')$, if there exists $j \in \text{cover}(rl)$ and $j = j'$, then set $C'_j = C_j$. Else, if node $j \in \text{cover}(rl)$ and $j \in \text{path}(j')$, i.e. $\text{path}(j') = \text{path}(j) \cup \{y_{\text{dep}(j)+1}, \ldots, y_{\text{dep}(j')}\}$, it computes $C_{y_{j'}} = (C_{y_j})^{(y_{j'}/y_j)} = g^s_{y_{j'}}$. Finally, it updates the ciphertext as follows: $CT_{rl'} = ((\mathcal{M}, \pi), C, C_0, cm, C_{\tau,1}, C_{\tau,2}, C_{\tau,3_{\tau\in[l_1]}}, \{C_{j'} = g^s_{j'}\}_{j'\in\text{cover}(rl')}, rl')$.

$PxyDec(CT_{rl}, \text{id}, PxyK_{\text{id}})$: The proxy server takes as input the ciphertext CT_{rl}, the user id, and the proxy decryption key $PxyK_{\text{id}}$, the proxy server abort the algorithm if the attributes in $PxyK_{\text{id}}$ do not satisfy the access structure in CT_{rl} or the user has been revoked, i.e. $\text{id} \in rl$. Otherwise, it can compute a set of constants $\omega_i \in Z_p$ for $I = \{i : \pi(i) \in U_{\text{id}}\}$, and the constants satisfy the equation $\sum_{i\in I} \omega_i \mathcal{M}_i = (1, 0, \ldots, 0)$. Then the proxy server needs to request the proxy decryption key update module by submitting $(\text{id}, E = g^b)$ to the central authority. When the user id is not revoked, then the central authority retrieve a node $j \in \text{path}(\theta_i) \cap cover(rl)$, where θ_i is a leaf node associated with id and $\text{path}(\theta_i)$ is the path from the root to the node θ_i. The central authority computes $E' = E^{y_{\theta_i}/y_j}$ and returns E' to the proxy server. Then the proxy server computes

$$B = \prod_{i\in I} \big(e(C_{i,1}, K_1)e(C_{i,2}, K_{\tau,2})e(C_{i,3}, K_{\tau,3})\big)^{\omega_i}$$

$$C' = \frac{e(C_0, K_0)e\big(E', C_j\big)}{e\big(C_0, (K_2)^{sk_{pxy}} \cdot D\big) \cdot B} = e(g, pk_{\text{id}})^{\alpha s}$$

and outputs the partially decrypted ciphertext $CT'_{rl} = \{(\mathcal{M}, \rho), C'\}$.

$Decrypt(CT_{rl}, CT'_{rl}, sk_{\text{id}})$: The data user takes as input the partially decrypted ciphertext CT'_{rl} and parses as the original ciphertext $CT_{rl} = ((\mathcal{M}, \pi), C, C_0, cm, C_{\tau,1}, C_{\tau,2}, C_{\tau,3_{\tau\in[l_1]}})$. Then it computes $(\text{msg}||r_m) = C \oplus H_1(C'^{sk_{\text{id}}^{-1}})$. If the equation $g^{H_2(\text{msg})}h^{H_2(r_m)} = cm$ holds, the data user accepts the message msg. Otherwise, the data user discards the message.

$Audit(CT_{rl}, CT'_{rl}, msk, pk_{\text{id}})$: The central authority takes as input the original ciphertext CT_{rl}, a partially decrypted ciphertext CT'_{rl}, the master secret key msk and the public key pk_{id} of user id, it retrieves C_0 from CT_{rl} and C' from CT'_{rl}. Then it checks whether the equation $e(pk^{\alpha}_{\text{id}}, C_0) = C'$ holds. If yes, it outputs 1 and means the proxy server returns a correct computing result. Otherwise, it outputs 0 and means the computing result is incorrect.

$Revoke(\text{id}, rl)$: When a user's decryption key is leaked or the user maliciously accuses the proxy server of returning an incorrect result, the central authority adds the user identity id to the revocation list and updates it to $rl' = rl \cup \text{id}$.

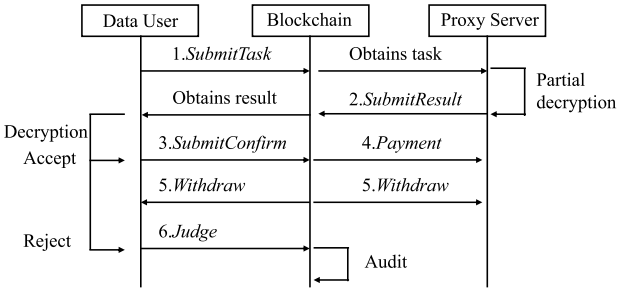


Fig. 3. Workflow of the protocol.

Fair and Efficient Revocable Data Sharing Protocol

- The data user retrieves CT_{rl} from the cloud server and uses its index $i_{CT_{rl}}$ to construct the computing task t_i, which is then published via the *SubmitTask* function.
- The proxy server obtains CT_{rl} from the cloud, then it runs $\mathbf{PxyDec}$ to generate a partially decrypted result, and submits it through *SubmitResult*.
- The data user retrieves the result from the blockchain and runs the $\mathbf{Decrypt}$ algorithm to obtain the plaintext. If correct, he confirms the task via *SubmitConfirm*.
- Upon confirmation, either party can trigger the *Payment* function to settle the transaction.
- Both the data user and proxy server can reclaim their deposits using the *Withdraw* function.
- If the result is incorrect or the user refuses to confirm, either party may invoke the central authority to arbitrate via the *Judge* function.

Fig. 4. Fair and efficient revocable data sharing protocol.

B. Integrating With Blockchain

To achieve fair payment, we integrate the proposed SR-ABE-VD with the blockchain. Since the smart contract is not suitable for performing too many cryptographic operations, most algorithms still need to be executed off the chain. Specifically, the proposed revocable data sharing scheme with fair payment has the following smart contract functions: *SubmitTask*, *SubmitResult*, *SubmitConfirm*, *Payment*, *Withdraw* and *Judge*. The workflow can be found in Fig. 3. And the high-level protocol can be found in Fig. 4. In the figure, the process of ciphertext uploading by the data owner is not depicted.

SubmitTask: After obtaining the ciphertext from the cloud, the data user can generate a computing task for the proxy server. To lower the communication overhead and the storage overhead of the blockchain, the data user can store the computing task in the cloud server and only submit the index of the task. It submits the task to the blockchain by receiving the task index and computation fee. To avoid the data user refusing to pay for the computing service, this function is set to be payable. When the data user executes this function, he needs to pay a certain amount of security deposit. We set a condition for the amount to be no less than the computing fee.

SubmitResult: When the proxy server gets a computing task, he can execute the *PxyDec* algorithm to obtain the task result. Then he runs this contract function to submit the result. It receives the task index and partially decrypted ciphertext as input and submits the result to the blockchain. This is also a payable function, which is used to avoid the proxy server submitting an incorrect result. The condition of the security deposit amount is the same as that of the *SubmitTask* function.

SubmitConfirm: When the data user accepts the computing result, which means the partially decrypted ciphertext can pass the integrity check, the data user can execute this contract function to submit a confirmation of the computing task.

Payment: When the data user submits confirmation of the computing task, the proxy server or the data user can execute this contract function to complete the payment. It takes the task index as input and handles the fee transfer to the proxy server. This function will check whether the data user submits confirmation of the task, then will check whether the payment has not been executed for this task. If all the conditions hold, the function will record the transaction and complete the payment.

Withdraw: When the task has been paid, the data user or proxy server can execute this contract function to retrieve the security deposit.

Judge: When the data user object to the computing result (resp. the data user keeps not submitting the result confirmation), the data user (resp. the proxy server) can request arbitration from the central authority. This contract function enables the central authority to determine whether the result is correct. The task index is passed to this function, and the contract function will execute the *Audit* algorithm and automatically confirm the task status if the computing result is correct. At this point, the proxy server can run the *Payment* function to complete the payment. If the computing result is incorrect, the contract will mark this task as completed and send the deposit paid by the proxy server to the corresponding data user as compensation. And when the task is marked as completed, the *Payment* will not transfer the computing fee to the proxy server.

V. Security Analysis

Security analyses are provided in this section. We develop the analyses from two aspects: 1) the selectively IND-CPA security and 2) the verifiability of decryption defined in Definition 2, Definition 3 and Definition 4. After that, we use the automated tools to verify the security formally.

A. Formal Analysis

Theorem 1: If the CP-ABE scheme in [37] is selectively IND-CPA secure, then the proposed SR-ABE satisfies the selectively IND-CPA security.

We use Π_{sr} to stand for the SR-ABE scheme and use Π_{rw} to denote the underlying CP-ABE scheme, respectively. Assume there exists an adversary can break the selectively IND-CPA security of the proposed scheme Π_{sr} with a non-negligible probability, then we can construct a simulator $\mathcal{B}$ to selectively break the scheme Πrw with a non-negligible probability.

Init: The simulator $\mathcal{B}$ obtains the tuple $(\mathcal{M}^*, \pi^*)$ from the adversary $\mathcal{A}$, where $(\mathcal{M}^*, \pi^*)$ is an access structure.

Setup: $\mathcal{B}$ forwards $(\mathcal{M}^*, \pi^*)$ to the challenger of Π_{rw} to obtain the public parameters $pp^*_{rw} = (\mathcal{G}, g, w, v, u, h, e(g,g)^\alpha)$. Then it chooses $x_{pxy} \in Z_p$ and computes $pk_{pxy} = g^{x_{pxy}}$. The proxy server's public secret key pair is set to be $(pk_{pxy} = g^{x_{pxy}}, sk_{pxy} = x_{pxy})$. Finally, the simulator initializes $rl \in \emptyset$ and a user tree BT, and the tuple $(pp^* = pp^*_{rw}, pk_{pxy}, sk_{pxy})$ is sent to $\mathcal{A}$.

Phase 1: $\mathcal{B}$ generates two empty list L_1, L_2 and responds the adversary's query requests as follows.

1) *UKGen(*id*):* The adversary queries on id to the *UKGen* query, the simulator $\mathcal{B}$ runs the *UKGen* algorithm to generate $(pk_{\mathrm{id}} = g^{x_{\mathrm{id}}}, sk_{\mathrm{id}} = x_{\mathrm{id}})$ if id has not been queried to the *UKGen*. Then the simulator inserts $(\mathrm{id}, pk_{\mathrm{id}}, sk_{\mathrm{id}})$ into L_1.
2) *PxyKGen(*id, U_{id}*):* Upon receiving the *PxyKGen* query on $(\mathrm{id}, U_{\mathrm{id}})$, $\mathcal{B}$ checks whether the id has been queried to *UKGen*(id), if not $\mathcal{B}$ generates $(pk_{\mathrm{id}, sk_{\mathrm{id}}})$ as in the *UKGen*(id) query. Then for the attribute set $U_{\mathrm{id}} \vdash (\mathcal{M}^*, \pi^*)$, where $\vdash$ means the attribute set satisfies the access policy, $\mathcal{B}$ computes the proxy server's decryption key $PxyK_{\mathrm{id}} = (K_0, K_1, K_2, D, E, \{K_{\tau,2}, K_{\tau,3}\}_{\tau\in[k]})$ as in the *PxyKGen* algorithm, i.e., it chooses $\beta, b, r, r_1, r_2, \ldots, r_k \in Z_p$ and computes $K_0 = g^{\beta r} pk_{\mathrm{id}}^{\alpha} pk_{pxy}^{\beta} w^r, K_1 = g^r, K_2 = g^\beta, \{K_{\tau,2} = g^{r_\tau}, K_{\tau,3} = (u^{U_\tau} h)^{r_\tau} v^{-r}\}_{\tau\in\{1,2,\ldots,k\}}$. Then it computes $D = g^{\beta r} g^b_{y_{\theta_j}}, E = g^b$. If the attribute set $U_{\mathrm{id}} \nvdash (\mathcal{M}^*, \pi^*)$, where $\nvdash$ means the attribute set does not satisfy the access policy, $\mathcal{B}$ sends U_{id} to the challenger of Π_{rw} to acquire the attribute related secret key $SK_U = (K^*_0, K^*_1, \{K^*_{i,2}, K^*_{i,3}\}_{i\in[k]})$, where $K^*_0 = g^\alpha w^r, K^*_1 = g^r, \{K^*_{i,2} = g^{r_i}, K^*_{i,3} = (u^{U_\tau} h)^{r_\tau} v^{-r}\}_{i\in[k]}$. Then it chooses $\beta, b, \overline{\alpha}$ and computes the proxy server's decryption key $PxyK_{\mathrm{id}} = (K_0, K_1, K_2, D, E, \{K_{\tau,2}, K_{\tau,3}\}_{\tau\in[k]})$ as $K_0 = K^*_0 g^{\beta r} pk_{\mathrm{id}}^{\overline{\alpha}} pk_{pxy}^{\beta}, K_1 = K^*_1, K_2 = g^\beta, E = g^b, D = g^{\beta r} g^b_{y_{\theta_j}}, \{K_{\tau,2} = K^*_{i,2}, K_{\tau,3} = K^*_{i,3}\}_{\tau\in[k]}$. Finally, the simulator inserts $(\mathrm{id}, U_{\mathrm{id}}, PxyK_{\mathrm{id}})$ into L_2.
3) *Corrupt(*id*):* The simulator $\mathcal{B}$ checks whether the entry $(\mathrm{id}, pk_{\mathrm{id}}, sk_{\mathrm{id}})$ exists in the list L_1, it aborts and returns $\perp$ if no such entry exists. Otherwise, it inserts the entry to the list L_1.
4) *Corrupt(*$PxyK_{\mathrm{id}}$*):* The simulator $\mathcal{B}$ checks whether the entry $(\mathrm{id}, U_{\mathrm{id}}, PxyK_{\mathrm{id}})$ exists in the list L_2, it returns $\perp$ if it not exists. Otherwise, it returns $PxyK_{\mathrm{id}}$.
5) *Revocation(*id*):* Upon receiving a query on identity id, $\mathcal{B}$ inserts id into the revocation list *rl*.

Challenge. The adversary $\mathcal{A}$ chooses a message tuple $(\mathrm{msg}_0, \mathrm{msg}_1)$ with the restriction that $|\mathrm{msg}_0| = |\mathrm{msg}_1|$ and submits it to the simulator. $\mathcal{B}$ forwards the message tuple to the challenger of Π_{rw} to obtain the ciphertext $CT_{rw} = (C, C_0, C_{\tau,1}, C_{\tau,2}, C_{\tau,3\tau\in[l_1]})$, where $C = \mathrm{msg}_b \cdot e(g,g)^{\alpha s}, C_0 = g^s, \{C_{\tau,1} = w^{\lambda_\tau} v^{t_\tau}, C_{\tau,2} = (u^{\pi(\tau)} h)^{-t_\tau}, C_{\tau,3} = g^{t_\tau}\}_{\tau\in[l_1]}$ Then $\mathcal{B}$ computes $C^* = C/\mathrm{msg}_b$ and sets $CT^*_{rl} = (C^*, C_0, cm, C_{\tau,1}, C_{\tau,2}, C_{\tau,3\tau\in[l_1]}, \{C_j = g^s_j\}_{j\in\mathrm{cover}(rl)}, rl)$. Finally, the challenge ciphertext CT^*_{rl} is sent to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ requests queries as in *Phase* 1.

Guess: $\mathcal{A}$ submits $b' \in \{0, 1\}$ and the simulator $\mathcal{B}$ sends it to the challenger of Π_{rw}.

According to the above analysis, we can observe that the distributions of the public parameters, secret keys and the ciphertext in the above IND-CPA game are identical to the real system. Therefore, the adversary that can selectively break the IND-CPA game of Π_{sr} with a non-negligible probability can also break Π_{rw} with the same probability.

Theorem 2: The proposed SR-ABE-VD scheme achieves selective IND-CPA secure, provided that the underlying SR-ABE is selective IND-CPA secure and the commitment scheme is computationally hiding.

Proof: We denote the length of the challenge message of the challenge phase in Theorem 1 to be $|\text{msg}_0| = |\text{msg}_1| = l_1$ and the fixed length of the random value selected in the *Encrypt* algorithm to be $|r_m| = l_2$. And the IND-CPA security of SR-ABE-VD can be proved as follows.

Game$_0$: The selective IND-CPA security game of SR-ABE-VD. Upon receiving the challenge message $(\text{msg}_0, \text{msg}_1)$ and an access structure $(\mathcal{M}^*, \pi^*)$ from the adversary, the simulator chooses $b \in \{0, 1\}$ and r^* and computes $CT^*_{rl} = \text{Encrypt}(\text{msg}_b||r^*_m, (\mathcal{M}^*, \pi^*))$ and the commitment of the message $cm^* = \text{Com}.C(\text{msg}^*_b, r^*_m)$, respectively.

Game$_1$: This game is basically the same as Game_0. When the adversary requests two message $(\text{msg}_0, \text{msg}_1)$, the simulator computes the challenge ciphertext by executes $CT^*_{rl} = \text{Encrypt}(0^{l_1+l_2}, (\mathcal{M}^*, \pi^*))$.

Game$_2$: This game is basically the same as Game_1. When the adversary requests two message $(\text{msg}_0, \text{msg}_1)$, the simulator computes the challenge ciphertext by executes $cm^* = \text{Com}.C(0^{l_1}, r^*_m)$.

To complete the proof we introduce two lemma to prove that $Adv^{\text{Game}_0}_{\mathcal{A}} \approx Adv^{\text{Game}_1}_{\mathcal{A}}$ and $Adv^{\text{Game}_1}_{\mathcal{A}} \approx Adv^{\text{Game}_2}_{\mathcal{A}}$, where $\approx$ means computationally indistinguishability. In Game_2, the challenge messages selected by the adversary is independent of the ciphertext which means the adversary has no advantage. According to the fact that $Adv^{\text{Game}_0}_{\mathcal{A}} \approx Adv^{\text{Game}_1}_{\mathcal{A}}$ and $Adv^{\text{Game}_1}_{\mathcal{A}} \approx Adv^{\text{Game}_2}_{\mathcal{A}}$, we can obtain that $Adv^{\text{Game}_0}_{\mathcal{A}} \approx Adv^{\text{Game}_2}_{\mathcal{A}}$. Thus, we can conclude that the adversary $\mathcal{A}$ has a negligible advantage in the selective IND-CPA security of SR-ABE-VD (Game_0). ■

Lemma 1: Assuming the SR-ABE satisfies selective IND-CPA secure, it follows that no PPT adversary is able to distinguish Game_0 and Game_1 with non-negligible advantage.

Proof: Assume that $\mathcal{A}$ can distinguish Game_0 and Game_1 with non-negligible advantage, then we can construct $\mathcal{B}$ that can break the IND-CPA security of the SR-ABE with non-negligible advantage.

We denote $\mathcal{C}$ the challenger in the IND-CPA security game of SR-ABE. And $\mathcal{B}$ answers $\mathcal{A}$'s requests as follows.

Setup: $\mathcal{C}$ executes *Setup* to obtain *pp* and *msk*. The *pp* is sent to the simulator $\mathcal{B}$ and $\mathcal{B}$ then runs $pp_{cm} \leftarrow \text{Com}.Init(1^\lambda)$ to get the parameters of the commitment scheme. The tuple (pp, pp_{cm}) is sent to $\mathcal{A}$.

Phase 1: Upon receiving the *UKGen* query (resp. *PxyKGen* query) on user identity id (resp. user attributes U_{id}) from the adversary $\mathcal{A}$, $\mathcal{B}$ executes its own *UKGen* oracle (resp. *PxyKGen* oracle) and returns the outputs to $\mathcal{A}$.

Challenge: $\mathcal{A}$ chooses two challenge message $|\text{msg}_0| = |\text{msg}_1| = l_1$ and an access structure $(\mathcal{M}^*, \pi^*)$ satisfies that $U \not\vdash (\mathcal{M}^*, \pi^*)$ for all queried $(\text{id}, U_{\text{id}})$. The challenge messages and the access structure are submitted to the simulator $\mathcal{B}$. $\mathcal{B}$ chooses $r^* \in Z_p$ and computes $cm^* = \text{Com}.C(\text{msg}_b, r^*_m)$ by randomly chooses a bit $b \in \{0, 1\}$. Then it sets $(M_0, M_1) = (\text{msg}_b||r^*_m, 0^{l_1+l_2})$ and forwards it along with an access structure $(\mathcal{M}^*, \pi^*)$ to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ executes $CT^*_{rl} = \text{Encrypt}(M_b, (\mathcal{M}^*, \pi^*))$ and returns it to the simulator. Finally, $\mathcal{B}$ constructs the tuple (CT^*_{rl}, cm^*) and sends it to $\mathcal{A}$.

Phase 2: This phase is basically the same as Phase 1 except that $\mathcal{A}$ cannot query the *UKGen* on the identity id that the attributes U_{id} associated with id satisfy the access structure (i.e., $U_{\text{id}} \vdash (\mathcal{M}^*, \pi^*)$).

Guess: The adversary $\mathcal{A}$ submits $b' \in \{0, 1\}$ to the simulator, $\mathcal{B}$ outputs 0 if $b = b'$ else outputs 1.

We can observe that if the encrypted message is $M_0 = \text{msg}_b||r^*_m$, then it is exactly the same as in Game_0. And when the encrypted message is $M_1 = 0^{l_1+l_2}$, it is exactly the same as in Game_1. Suppose there exists adversary $\mathcal{A}$ can exhibit a non-negligible difference in advantage between the Game_0 and Game_1, then we can construct algorithm $\mathcal{B}$ break the selective IND-CPA security of SR-ABE with a non-negligible advantage. Thus, we can prove the above lemma. ■

Lemma 2: If the underlying commitment scheme of SR-ABE-VD is computationally hiding, then no adversary can distinguish Game_1 and Game_2 with a non-negligible advantage.

Proof: Assume that the PPT adversary $\mathcal{A}$ can distinguish Game_1 and Game_2 with a non-negligible advantage, then we can construct $\mathcal{B}$ that can break the underlying commitment scheme in the hiding property with a non-negligible advantage.

We denote $\mathcal{C}$ the challenger in the underlying commitment hiding property game. The simulator $\mathcal{B}$ responds $\mathcal{A}$'s requests as follows.

Setup: The challenger $\mathcal{C}$ executes $pp_{cm} \leftarrow \text{Com}.Init(1^\lambda)$ to get the public parameters of the commitment scheme. Given the input pp_{cm}, $\mathcal{B}$ runs *Setup* to obtain *pp* and *msk*. The tuple (pp, pp_{cm}) is sent to $\mathcal{A}$.

Phase 1: Upon receiving the *UKGen* queries or the *PxyKGen* queries from the adversary $\mathcal{A}$, the simulator $\mathcal{B}$ executes its own *UKGen* oracle or *PxyKGen* oracle and returns the outputs to the adversary.

Challenge: $\mathcal{A}$ chooses two challenge message $|\text{msg}_0| = |\text{msg}_1| = l_1$ and an access structure $(\mathcal{M}^*, \pi^*)$ satisfies that $U_{\text{id}} \not\vdash (\mathcal{M}^*, \pi^*)$ for all queried $(\text{id}, U_{\text{id}})$. The tuple $(\text{msg}_0, \text{msg}_1, (\mathcal{M}^*, \pi^*))$ will be submitted to the simulator $\mathcal{B}$. $\mathcal{B}$ executes the *Encrypt* algorithm to obtain the ciphertext $CT^*_{rl} = \text{Encrypt}(0^{l_1+l_2}, (\mathcal{M}^*, \pi^*))$ and chooses a random bit $b \in \{0, 1\}$. Then it sends $(M_0, M_1) = (\text{msg}_b, 0^{l_1})$ to $\mathcal{C}$ to obtain the commitment on M_β. The challenger $\mathcal{C}$ chooses a random value $r^*_m \in Z_p$ and computes $cm^* = \text{Com}.C(M_\beta, r^*_m)$. Finally, $\mathcal{B}$ constructs the tuple (CT^*_{rl}, cm^*) and sends it to $\mathcal{A}$.

Phase 2: This phase is basically the same as Phase 1 except that $\mathcal{A}$ cannot query the *UKGen* on the identity id that the attributes U_{id} associated with id satisfy the access structure.

Guess: The adversary $\mathcal{A}$ submits $b' \in \{0, 1\}$ to the simulator, $\mathcal{B}$ outputs 0 if $b = b'$ else outputs 1.

We can observe that the view of adversary $\mathcal{A}$ is basically the same as in Game_1 if $\beta = 0$ and when $\beta = 1$ the view of $\mathcal{A}$ is identical to that of Game_2. If $\mathcal{A}$ can exhibit a non-negligible difference in advantage between the Game_1 and Game_2, then we can construct algorithm $\mathcal{B}$ breaks the commitment scheme's hiding property. Thus, we can prove the above lemma. ■

Theorem 3: If the scheme Π_{cm} is computationally binding, then Π_{sr} is decryption verifiable.

Proof: Assume that $\mathcal{A}$ can break the verifiability game of SR-ABE-VD, then we can utilize $\mathcal{A}$ to construct a simulator algorithm $\mathcal{B}$ that can break the binding property of Π_{cm} with non-negligible advantage.

$\mathcal{B}$ obtains (pp_{cm}, pp, msk) as in the natural way and sends the tuple to the adversary $\mathcal{A}$. $\mathcal{B}$ can answer the *UKGen* and *PxyKGen* queries using the master secret key. Upon receiving the challenge message msg* and an access structure $(\mathcal{M}^*, \pi^*)$ from $\mathcal{A}$, $\mathcal{B}$ constructs (CT^*_{rl}, cm^*) and returns to $\mathcal{A}$. $\mathcal{B}$ continues to response the queries from $\mathcal{A}$ as above. Finally, $\mathcal{A}$ returns $((\mathcal{M}^*, \pi^*), CT^*_T)$, where CT^*_T is a partially decrypted ciphertext. $\mathcal{A}$ wins the game if $\text{Dec}(CT^*_{rl}, CT^*_T, sk_{\text{id}}) \notin \{m^*, \perp\}$, where sk_{id} is associated with the challenge access structure $(\mathcal{M}^*, \pi^*)$.

When $\mathcal{A}$ wins the verifiability game, we can obtain a different message msg′ satisfies $\text{msg}'||r'_m = \text{Dec}(CT^*_{rl}, CT^*_T, sk_{\text{id}})$ and $\text{Com}.V(\text{msg}', r'_m, cm^*) = 1$ where $\text{msg}' \notin \{\text{msg}^*, \perp\}$. This means the simulator $\mathcal{B}$ reveals the commitment and gets two different messages $(\text{msg}^*, \text{msg}')$, which is conflict with the binding property of Π_{cm}. Thus, we can prove the theorem, i.e., the proposed scheme is decryption verifiable. ■

B. Further Analysis

To further analyze the security, we employed the automated verification tool ProVerif [38] to formally verify the protocol's security. As ABE is not natively supported in ProVerif, we modeled it as a probabilistic public key encryption scheme. In addition, we modeled the decryption behavior of the proxy server and the commitment generation performed by the data owner during encryption.

The primary security goal of our scheme is IND-CPA, i.e., the adversary should not be able to distinguish which of two chosen plaintexts was encrypted. This property is captured in ProVerif using observational equivalence. Another goal is to ensure that an adversary without a valid private key cannot decrypt the ciphertext, thus guaranteeing both message confidentiality and key secrecy. This property is expressed in ProVerif through query-based secrecy specifications. Since equivalence properties and adversary queries are incompatible in ProVerif, the two security goals must be verified separately. Fig. 5 illustrates part of the source code used to implement the protocol. Fig. 6 presents a summary of the verification results, combining the outcomes of both analyses into a single overview. The results confirm that our scheme satisfies the intended security goals.

```
(* Process for secrecy analysis *)
free c: channel.
free m:message[private].
free uSkey:skey[private].
query attacker(m).
query attacker(uSkey).
process
    new attrUserA: attrlist;
    let uSkey=ABE_KeyGen(attrUserA) in
    let pubk = pk(uSkey) in
    let privk = psk(uSkey) in
    out(c, pubk);
    new pol: policy;
    let ct = ABE_Enc(m, pubk, pol) in
    out(c, ct);
    new r:randomness;
    let com = commit(m,r) in
    out(c, com);
    let pct = ABE_Pdec(ct, privk) in
    out(c, pct)
(* Process for equivalence analysis *)
free c: channel.
free m0:message[private].
free m1:message[private].
process
    new attrUserA: attrlist;
    let s=ABE_KeyGen(attrUserA) in
    let pubk = pk(s) in
    let privk = psk(s) in
    out(c, pubk);
    new pol: policy;
    let m=choice[m0, m1] in
    let ct = ABE_Enc(m, pubk, pol) in
    out(c, ct);
    new r: randomness;
    let com = commit(m,r) in
    out(c, com);
    let pct = ABE_Pdec(ct, privk) in
    out(c, pct)
```

Fig. 5. Main processes in ProVerif.

```
Verification summary:
Query not attacker(m[]) is true.
Query not attacker(uSkey[]) is true.
Observational equivalence is true.
```

Fig. 6. Output of the verification.

VI. Performance Evaluation

This section presents the comparisons of the proposed scheme and other schemes [4], [26], [30], [35]. The reason for choosing these schemes to analyze is that scheme [4] first introduced server-aided revocable ABE and scheme [26], [30], [35] have the property of data integrity or

TABLE II
BENCHMARK RESULTS OF VARIOUS OPERATIONS (MS)

Groups	mul	exp	h_1	h_r	p	sr
G	0.002	1.036	2.259	0.0008	0.636	0.019
G_T	0.0008	0.091	-	-	-	-

decryption verifiability. In the following, we concentrate on the analysis with respect to computing overhead, communication overhead, and smart contract overhead.

Computing Overhead: We use the Charm (version 0.5) framework and Python 3.7 to implement the proposed scheme and the comparison schemes. Experiments were executed on a desktop computer with an Intel core i5-10500 CPU @ 3.1 GHz and 12 GB RAM running the Windows Subsystem for Linux (Ubuntu 20.04) in a Windows 11 host environment.

In Table II, the benchmark time of basic operations are provided. We used the SS512 curve for testing, and the benchmark time is the average of 100 runs. In the table *mul* is the multiplication operation, exp stands for the exponential operation, h_1 is the hash operation of hashing an attribute to a group element, h_r denotes the hash operation of hashing an attribute to a random integer, p is the pairing operation, sr is the operation of selecting a random integer. From the table we can see that the operations of exp, h_1 and p take more time. Therefore, we only consider these operations in the following theoretical analysis.

In Table III, the efficiency comparisons are provided. For the **Keygen** algorithm, the scheme [35] requires minimal exponential operation and [4] requires a maximal exponential operation. The *Keygen* algorithm here denotes the generation of attribute-related keys (e.g. *PxyKGen* of the prosed scheme). For the *Encrypt* algorithm, the scheme [26] requires more exponential operations than other schemes. Since the underlying attribute encryption algorithm of the proposed scheme and schemes [4], [30] is the same, the exponential operation required in the *Encrypt* algorithm time is basically the same. For the *Decrypt* algorithm, the proposed scheme and schemes [4], [30], [35] both introduce a proxy server to assist the user decryption and both involve a few exponential operations. For the *Transform* algorithm (e.g. *PxyDec* of the proposed scheme), the scheme [30] needs to generate an additional transformation ciphertext for verification, which requires about twice the computation operation compared with the proposed scheme and scheme [4]. For the *Revoke* algorithm, the scheme [4] needs to generate the transformation update key component and the updated transformation key. Scheme [26], [35] must merge the revoked access policy with the original one to form a new access policy and update the ciphertext. Only the ciphertext module relating to the revoked user has to be updated in the proposed scheme.

To further evaluate the computing overhead of the proposed scheme, we implement prototypes of the proposed scheme and other schemes. The access policy follows the form "Att_1 and $\text{Att}_2, \ldots$ and Att_n"s, the attributes number involved in the *Keygen*, *Encrypt*, *Transform*, *Decrypt* and *Revoke* is set from 10 to 50 in increments of 10. The number of revoked attributes in [26], [35] is set equal to the attributes number

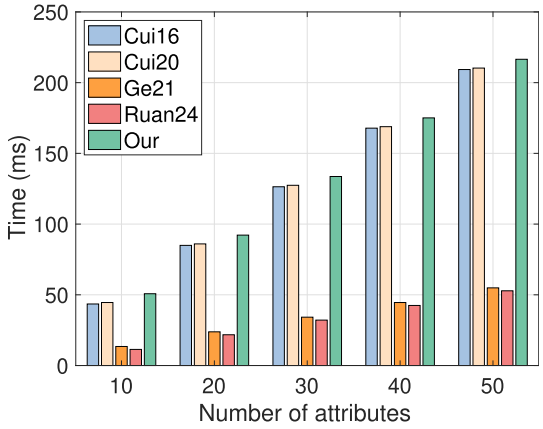


Fig. 7. Keygen.

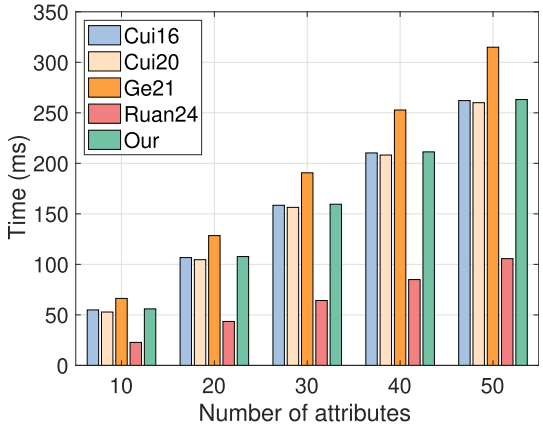


Fig. 8. Encrypt.

used in the encryption phase. The results are the average runtime measured for ten runs of each algorithm.

Fig. 7 shows the correlation between the computing overhead of key generation algorithm and the attributes number. It should be noted, however, that in scheme [4], the generation of attribute keys is related to the tree hierarchy in which the user node is located. A user must generate the corresponding private key for each node on its path. We only show the time required to generate a node key for the sake of analysis.

As illustrated in Fig. 8, encryption overhead in these schemes is attribute-dependent and increases accordingly. The encryption overhead of [4], [30] and the proposed scheme is essentially the same, which is consistent with the theoretical analysis results. Since scheme [26] generates extra ciphertext when implementing integrity, its encryption overhead is roughly double that of the other schemes.

As shown in Fig. 9, there is no significant difference in transformation overhead between the proposed scheme and scheme [4]. Scheme [30] requires the generation of additional outsourcing transformation ciphertext in order to implement outsourced computing verification, and its computing overhead is relatively high.

As shown in Fig. 10, the proposed scheme and schemes [4], [30] have low decryption costs that do not vary with the attributes number because of the adoption

TABLE III
EFFICIENCY COMPARISONS

Scheme	Keygen	Encrypt	Decrypt	Transform	Revoke
Cui16 [4]	$\log N(2+4I)exp$	$(5I+3)exp$	$1exp$	$(3I+2)p+Iexp$	$(4I+8)exp$
Cui20 [30]	$(4I+3)exp$	$(5I+1)exp$	$1exp$	$(6I+2)p+2Iexp$	-
Ge21 [26]	$(3+I)exp$	$(6I+4)exp$	$(4I+2)p+2Iexp$	-	$(6(I+J)+2)exp$
Ruan24 [35]	$(I+1)exp$	$(2I+2)exp$	$1exp$	$(2p+2Iexp)$	$2(I+J+1)exp$
Our scheme	$(4I+9)exp$	$(5I+4)exp$	$3exp$	$(3I+3)p+(I+2)exp$	$2exp$

exp indicates the exponential operation and p indicates the paring operations. I denotes the number of user attributes, J denotes the number of revoked attributes and N denotes the max number of the system user.

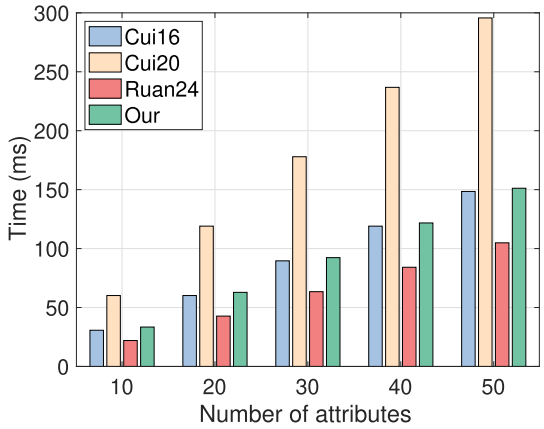


Fig. 9. Transform.

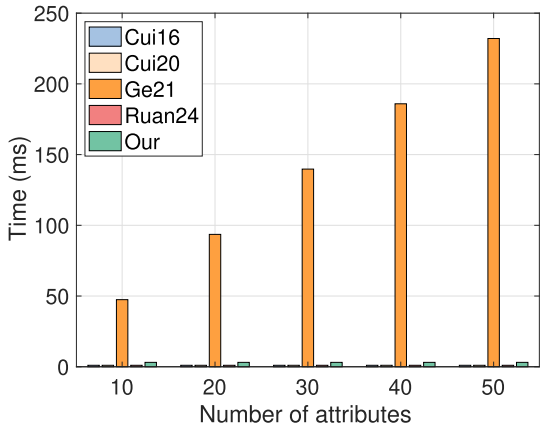


Fig. 10. Decrypt.

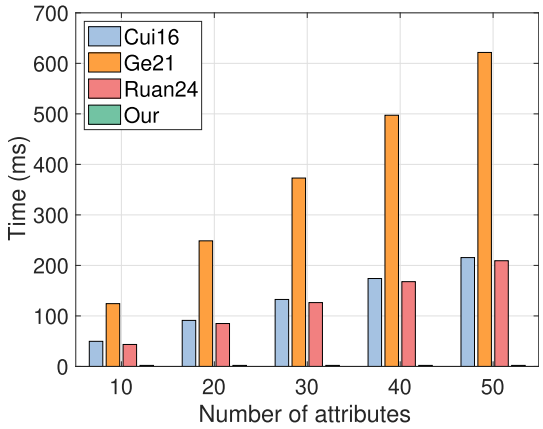


Fig. 11. Revoke.

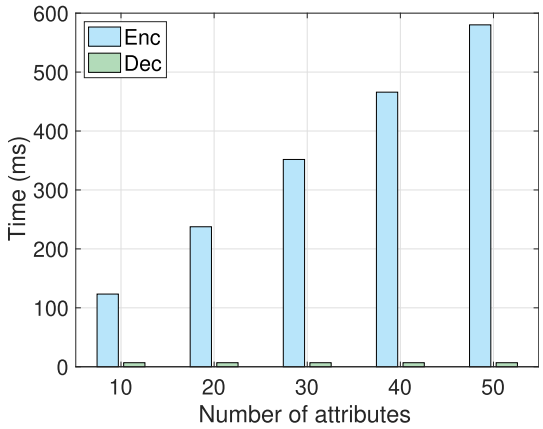


Fig. 12. Computation cost on Raspberrypi.

of server-aided decryption. Scheme [26] has high computational overhead because it does not use server-aided decryption.

It is worth noting that although our scheme performs slightly worse than that of [35] in the key generation, transformation, and encryption, it achieves better performance in user revocation. In addition, [35] attains fairness by involving multiple blockchain nodes to perform partial decryption and comparing their results with those of the outsourced computation provider, thereby ensuring both verifiability and fairness of the computation. In contrast, our scheme enables the verification of outsourced results with significantly lower computational overhead. With respect to revocation, their approach requires updating the ciphertext access policy, resulting in substantial computational costs.

Fig. 11 shows our scheme incurs the minimal computational overhead for user revocation. In contrast, the revocation process in [26], [35] involves operations that are related to both the attributes number used in the encryption phase and attributes number to be updated, resulting in higher overhead.

To validate the computational efficiency of our scheme, we carried out additional experiments using a Raspberry Pi 4B with 4 GB RAM to simulate a resource-constrained device, and a cloud server equipped with a 4 GB RAM Intel Xeon Gold 6133 CPU @ 2.50 GHz to simulate an edge server. We performed encryption and decryption on the Raspberry Pi, while the outsourced transformation and revocation operations were executed on the edge server. Fig. 12 demonstrates that our scheme achieves efficient decryption performance on resource-constrained devices. Although the encryption phase

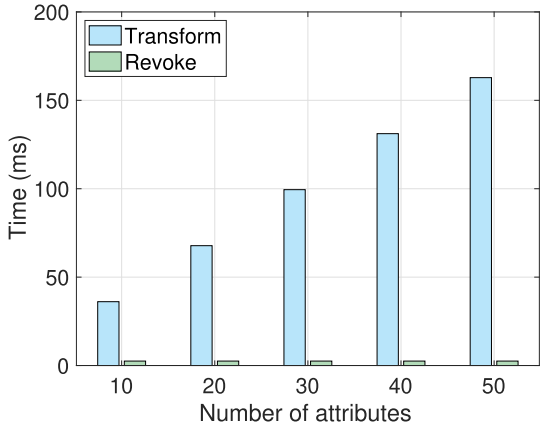


Fig. 13. Computation cost on edge server.

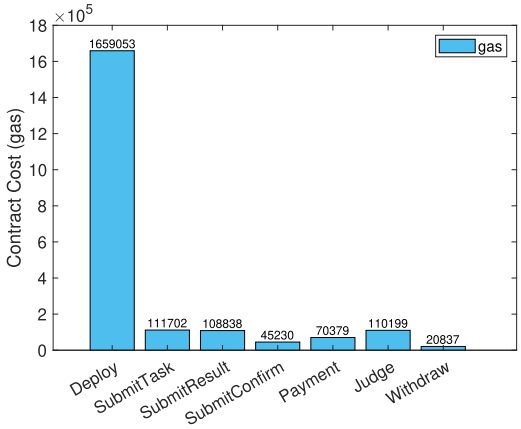


Fig. 14. Contract cost.

TABLE IV
COMMUNICATION COMPARISONS

Scheme	Key	Ciphertext	Transformed ciphertext
Cui16 [4]	$\log N(2+2I)\|G\|$	$(3I+2)\|G\|+\|G_T\|$	$2\|G_T\|$
Cui20 [30]	$(2I+2)\|G\|$	$(3I+1)\|G\|+\|G_T\|$	$3\|G_T\|$
Ge21 [26]	$(I+2)\|G\|$	$(4I+2)\|G\|+2\|G_T\|$	-
Ruan24 [35]	$(I+2)\|G\|$	$(I+1)\|G\|+\|G_T\|$	$\|G_T\|$
Our scheme	$(2I+3)\|G\|$	$(3I+2)\|G\|$	$\|G_T\|$

$|G|, |G_T|$ indicates the group element size, I indicates the attributes number, N is the maximize number of the system users. Note that $|G_T| = 2|G|$ on curve SS512.

incurs a higher overhead, it remains acceptable—with an encryption time of less than 0.6 s when the number of attributes is 50. For the edge server, the main computational cost lies in ciphertext transformation, which takes less than 0.2 s for 50 attributes. The experimental results can be found in Fig. 13.

Communication overhead: In Table IV, we analyze the communication overhead of these schemes in terms of key size, ciphertext size and transformed ciphertext size. From the table we can see that the attribute-related key in [26] has the least group elements of G. The key sizes in the proposed scheme and [30] are comparable. Scheme [4] generates larger keys, as the number of group elements depends on the depth of the user's leaf node. Our scheme maintains a similar ciphertext size to schemes [4] and [30]. Since the proposed scheme, as well as schemes [4], [30] and [35], employ comparable server-aided decryption methods, their transformed ciphertext sizes are roughly identical.

Smart Contract Overhead: To evaluate the smart contract overhead, we use the Solidity language (version 0.8.13) to implement the prototype of our smart contract with Remix online development environment. Due to the implementation of the *Judge* function requires a pair compute and comparison and Ethereum does not provide a pairing check function that can be directly used for the judgment, we created a precompiled smart contract to complete this function, which also limits the contract's testing to the private chain. The gas cost of *Deploy*, *SubmitTask*, *SubmitResult*, *SubmitConfirm*, *Payment*, *Judge*, and *Withdraw* functions were all tested. Fig. 14 shows the gas cost of these functions. When the price of gas is set at 1 Gwei and the exchange rate is 1 eth = 1805.75 USD at the time of writing, the capital cost is U.S. $2.99, U.S. $0.20, U.S. $0.19, U.S. $0.08, U.S. $0.12, U.S. $0.19, and U.S. $0.03,, respectively. The block generation interval on Ethereum is approximately 12 s, which determines the upper bound of response latency for on-chain operations. Assuming a block can accommodate about 200 KB of transaction data and an average transaction size of 300 bytes, a block can theoretically include over 600 transactions. This corresponds to a throughput of approximately 50 transactions per second (TPS), which is sufficient for many IIoT environment, especially considering that most operations happen off-chain.

To address the scalability concern in IIoT scenarios with potentially high-frequency data access and task submission, our design minimizes on-chain workload. All cryptographic operations, such as key generation, encryption, and partial decryption are executed off-chain. Only the minimal metadata and transaction outcomes (e.g., task submission status and final payment) are uploaded on-chain. This ensures that the throughput of our scheme is bounded not by the blockchain's transaction rate. Furthermore, since the payment process is decoupled from data access, and disputes are rare in typical use, the *Judge* function is expected to be triggered infrequently, further reducing the overall chain overhead. In conclusion, by minimizing on-chain cryptographic operations and carefully partitioning tasks between the blockchain and external components, the proposed approach ensures security, fairness, and scalability, demonstrating its practicality for deployment in IIoT contexts.

VII. CONCLUSION

In this study, we constructed a fair and efficient revocable access control to enable fine-grained data sharing for IIoT environment. The key contribution lies in the secure integration of cryptographic primitives with blockchain technology. To balance security and efficiency, the scheme minimizes the cryptographic operations required on-chain. By integrating server-aided revocable ABE with verifiable decryption and leveraging blockchain technology, our scheme ensures not only confidentiality and revocation flexibility, but also fair payment and result verifiability. Formal security analysis proves that our scheme achieves desired security goals. Experimental results further confirmed its practicality and acceptable overhead. As part of future research, we will examine the integration of advanced cryptographic tools with blockchain technology to

enable fair payment among multiple parties, thus extending the scheme's applicability to more complex and practical scenarios. In addition, we aim to design privacy-preserving access control schemes with support for policy hiding, which is particularly desirable in decentralized environments, such as blockchains.

ACKNOWLEDGMENT

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this article.

REFERENCES

[1] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4682–4696, Jun. 2020.

[2] W. Weng, J. Li, Y. Zhang, Y. Lu, J. Shen, and J. Han, "Efficient registered attribute based access control with same sub-policies in mobile cloud computing," *IEEE Trans. Mobile Comput.*, early access, Mar. 31, 2025, doi: 10.1109/TMC.2025.3556279.

[3] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: A survey," *ACM Comput. Surveys*, vol. 53, no. 4, pp. 1–41, 2020.

[4] H. Cui, R. H. Deng, Y. Li, and B. Qin, "Server-aided revocable attribute-based encryption," in *Proc. Eur. Symp. Res. Comput. Security*, 2016, pp. 570–587.

[5] B. Qin, Q. Zhao, D. Zheng, and H. Cui, "(Dual) server-aided revocable attribute-based encryption with decryption key exposure resistance," *Inf. Sci.*, vol. 490, pp. 74–92, Jul. 2019.

[6] C. Lin, D. He, X. Huang, and K.-K. R. Choo, "OBFP: Optimized blockchain-based fair payment for outsourcing computations in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3241–3253, 2021.

[7] X. Chen, J. Li, and W. Susilo, "Efficient fair conditional payments for outsourcing computations," *IEEE Trans. Inf. Forensics Security*, vol. 7, pp. 1687–1694, 2012.

[8] X. Zhang et al., "A data trading scheme with efficient data usage control for industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4456–4465, Jul. 2022.

[9] Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, "Outsourcing service fair payment based on blockchain and its applications in cloud computing," *IEEE Trans. Services Comput.*, vol. 14, no. 4, pp. 1152–1166, Jul./Aug. 2021.

[10] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and W. Susilo, "Blockchain-based public auditing and secure deduplication with fair arbitration," *Inf. Sci.*, vol. 541, pp. 409–425, Dec. 2020.

[11] C. Wang et al., "Smart contract-based integrity audit method for IoT," *Inf. Sci.*, vol. 647, Nov. 2023, Art. no. 119413.

[12] Y. Guo, C. Zhang, C. Wang, and X. Jia, "Towards public verifiable and forward-privacy encrypted search by using blockchain," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 3, pp. 2111–2126, May/Jun. 2023.

[13] L. Chen, S. Xu, H. Zhang, and J. Weng, "Fair-and-exculpable-attribute-based searchable encryption with revocation and verifiable outsourced decryption using smart contract," *IEEE Internet Things J.*, vol. 12, no. 4, pp. 4302–4317, Feb. 2025.

[14] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng, "VeriML: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2524–2540, Oct. 2021.

[15] X. Zhang et al., "Secure collaborative learning in mining pool via robust and efficient verification," in *Proc. IEEE 43rd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2023, pp. 794–805.

[16] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 1343–1354, 2013.

[17] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Security Symp.*, 2011, pp. 1–16.

[18] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, pp. 2119–2130, 2015.

[19] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 94–105, 2017.

[20] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 478–487, May/Jun. 2020.

[21] Y. Miao et al., "Verifiable outsourced attribute-based encryption scheme for cloud-assisted mobile e-health system," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 1845–1862, Jul./Aug. 2024.

[22] L. Sun, C. Xu, and F. Zeng, "Verifiable and hybrid attribute-based proxy re-encryption for flexible data sharing in cloud storage," *J. Parallel Distrib. Comput.*, vol. 193, Nov. 2024, Art. no. 104956.

[23] Y. Guo, Z. Lu, H. Ge, and J. Li, "Revocable blockchain-aided attribute-based encryption with escrow-free in cloud storage," *IEEE Trans. Comput.*, vol. 72, no. 7, pp. 1901–1912, Jul. 2023.

[24] A. Peñuelas-Angulo, C. Feregrino-Uribe, and M. Morales-Sandoval, "A revocable multi-authority attribute-based encryption scheme for fog-enabled IoT," *J. Syst. Archit.*, vol. 155, Oct. 2024, Art. no. 103265.

[25] F. Meng and L. Cheng, "TSR-ABE: Traceable and server-aided revocable ciphertext-policy attribute-based encryption under static assumptions," *IEEE Trans. Inf. Forensics Security*, vol. 20, pp. 955–967, 2024.

[26] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 2864–2872, Sep./Oct. 2022.

[27] S. Chen, J. Li, Y. Zhang, and J. Han, "Efficient revocable attribute-based encryption with verifiable data integrity," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10441–10451, Mar. 2024.

[28] H. Liu, Y. Ming, C. Wang, and Y. Zhao, "Flexible selective data sharing with fine-grained erasure in VANETs," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 9582–9597, 2024.

[29] B. Carbunar and M. V. Tripunitara, "Payments for outsourced computations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 313–320, Feb. 2012.

[30] H. Cui, Z. Wan, X. Wei, S. Nepal, and X. Yi, "Pay as you decrypt: Decryption outsourcing for functional encryption using blockchain," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3227–3238, 2020.

[31] D. Liu, C. Huang, J. Ni, X. Lin, and X. S. Shen, "Blockchain-cloud transparent data marketing: Consortium management and fairness," *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3322–3335, Dec. 2022.

[32] C. Ge, Z. Liu, W. Susilo, L. Fang, and H. Wang, "Attribute-based encryption with reliable outsourced decryption in cloud computing using smart contract," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 2, pp. 937–948, Mar./Apr. 2024.

[33] Z. Feng, Q. Wu, Y. Liu, B. Qin, M. Zhai, and W. Susillo, "Secure and fair data trading based on blockchain with enhanced access control," *IEEE Internet Things J.*, vol. 12, no. 6, pp. 7277–7292, Mar. 2025.

[34] F. Chen, H. Zhang, T. Xiang, and J. K. Liu, "A two-stage approach for fair data trading based on blockchain," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 9835–9849, 2024.

[35] C. Ruan, C. Hu, X. Li, S. Deng, Z. Liu, and J. Yu, "A revocable and fair outsourcing attribute-based access control scheme in metaverse," *IEEE Trans. Consum. Electron.*, Feb. 2024.

[36] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 41–62.

[37] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 463–474.

[38] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, "ProVerif 2.00: Automatic cryptographic protocol verifier, user manual and tutorial," 2018. [Online]. Available: https://publications.bensmyth.com/2010-ProVerif-manual-version-2.00/

Bei Li is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Anhui University, Hefei, China.

His research interests include IoT security, blockchain, and applied cryptography.

Hong Zhong was born in Anhui Province, China, in 1965. She received the Ph.D. degree in computer science from University of Science and Technology of China, Hefei, China, in 2005.

She is currently a Professor and the Ph.D. supervisor with the School of Computer Science and Technology, Anhui University, Hefei. She has over 200 scientific publications in reputable journals, such as IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and IEEE TRANSACTIONS ON BIG DATA, and academic books and international conferences. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking.

Jing Zhang was born in Henan Province, China, in 1990. She received the M.A.Eng. and Ph.D. degrees in computer science from Anhui University, Hefei, China, in 2021.

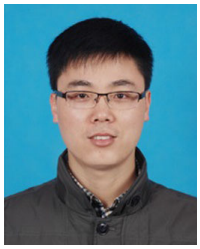
She is currently a Professor with the School of Computer Science and Technology, Anhui University. She has nearly 20 scientific publications in reputable journals, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *Information Sciences*, *Science China Information Sciences*, and *Vehicular Communications*, and international conferences. Her research interests include vehicular ad hoc network, IoT security, and applied cryptography.

Qingyang Zhang was born in Anhui, China, in 1992. He received the B.Eng. and Ph.D. degrees in computer science from Anhui University, Hefei, China, in 2014 and 2021, respectively.

He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. He has over 30 scientific publications in reputable journals, such as PROCEEDINGS OF THE IEEE, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON COMPUTERS, and international conferences. His research interests include edge computing, computer systems, and security.

Jiaxin Li received the master's degree from Anhui University, Hefei, China, in 2020 where he is currently pursuing the Doctoral degree.

He is currently holds the position of Director of Government Affairs with H3C Information Security Technology Company Ltd., Hefei. His research focuses on the security of vehicular ad hoc network, data security, and the security of Industrial Internet of Things.



Jie Cui (Senior Member, IEEE) was born in Henan Province, China, in 1980. He received the Ph.D. degree from University of Science and Technology of China, Hefei, China, in 2012.

He is currently a Professor and the Ph.D. supervisor with the School of Computer Science and Technology, Anhui University. He has over 150 scientific publications in reputable journals, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON MULTIMEDIA), and academic books and international conferences. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking.