Attribute-Based Encryption With Reliable Outsourced Decryption in Cloud Computing Using Smart Contract

Chunpeng Ge, *Member, IEEE*, Zhe Liu, *Senior Member, IEEE*, Willy Susilo, *Fellow, IEEE*, Liming Fang, *Member, IEEE*, and Hao Wang, *Member, IEEE*

***Abstract*—Outsourcing the heavy decryption computation to a cloud service provider has been a promising solution for a resource-constrained mobile device to deploy an attribute-based encryption scheme. However, the current attribute based encryption with outsourced decryption schemes only enable the mobile device to verify whether the cloud service provider has returned a correct decryption result, they lack a mechanism to enable the cloud service provider to escape from a mobile device's wrong claim if it has returned a correct decryption result. This article, for the first time, proposes an attribute based encryption with reliable outsourced decryption scheme using the blockchain smart contract. In the proposed scheme, not only can the mobile device verify whether the cloud service provider has returned a correct decryption result, but also the cloud service provider can escape from a wrong claim if the returned decryption result is correct. Moreover, our system achieves the fairness property, which means the cloud service provider can get the reward from the mobile device if and only if it has returned a correct decryption result. Finally, we conduct an implementation to demonstrate that the proposed scheme is practical and efficient.**

***Index Terms*—Attribute-based encryption, reliable outsourced decryption, smart contract.**

I. Introduction

With the development of cloud computing, more and more individuals and enterprises prefer to buy a cloud service due to its massive storage and vast computing capability. With a cloud service, users can outsource their data to a cloud server and thus eliminate local data maintenance. The security problem is one of the critical obstacles that hinder the wide deployment of cloud computing. Encryption is a basic method to protect data confidentiality and attribute-based encryption is a promising representative for flexible access control on outsourced encrypted data [1]. In an attribute-based encryption system [2], [3], a user's data is encrypted with an access policy/attribute set, and the user's private key is generated under an attribute set/access policy. A ciphertext can be decrypted by a private key if and only if the attribute set satisfies the access policy.

Manuscript received 27 November 2021; revised 14 January 2023; accepted 3 April 2023. Date of publication 10 April 2023; date of current version 14 March 2024. This work was supported in part by the Australian Research Council Discovery Project under Grant DP180100665, in part by the National Natural Science Foundation of China under Grants 62032025, 62076125, 61702236, in part by the Natural Science Foundation of Jiangsu Province under Grants BK20200418, BE2020106, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2021A1515012650, and in part by the Shenzhen Science and Technology Program under Grant JCYJ20210324134810028. *(Corresponding author: Willy Susilo.)*

Chunpeng Ge is with the Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR) & School of Software, Shandong University, Jinan 250100, China (e-mail: gechunpeng2022@163.com).

Zhe Liu is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China, and also with Zhejiang Laboratory, Hangzhou 311121, China (e-mail: zheliu@nuaa.edu.cn).

Willy Susilo is with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: wsusilo@uow.edu.au).

Liming Fang and Hao Wang are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China (e-mail: fangliming@nuaa.edu.cn; wangh24@nuaa.edu.cn).

Digital Object Identifier 10.1109/TDSC.2023.3265932

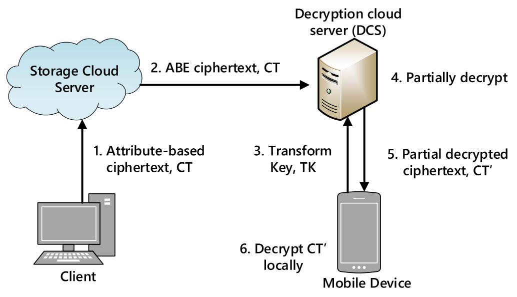


Fig. 1. Current OABE architecture.

One of the main drawbacks of attribute-based encryption is the complex decryption cost which is in proportion to the size of the access policy [2], [3], [4]. This limitation makes the attribute-based encryption scheme infeasible for the mobile application in which a resource constrained mobile device (e.g., smartphone) is used to retrieve data from the encrypted ciphertext stored in the cloud. To address this problem, Green et al. [5] introduced the notion of attribute-based encryption with outsourced decryption (OABE) which eliminates the decryption overhead from a mobile device. In an attribute-based encryption with outsourced decryption scheme, depicted in Fig. 1, a mobile device leverages a decryption cloud server (DCS) to execute most of the decryption work for it. When a mobile device wants to outsource the decryption task to the decryption cloud server, it first generates a transformation key TK and a retrieve key RK using its private key SK. Then it sends the transformation key TK to the decryption cloud server and keeps the retrieve key RK privately. Using the transformation key TK, the decryption cloud server can transform a user's ABE ciphertext to an ElGamal ciphertext [6]. Finally, the mobile device can decrypt the ElGamal ciphertext with the retrieve key on the mobile device only at a very little computation cost.

Though OABE enables a mobile device to decrypt the ciphertext efficiently, it cannot prevent a malicious decryption cloud server from returning an incorrect transformed ciphertext. The decryption cloud server has a strong incentive to return an incorrect result since the transformation is computation consuming [5].

The decryption cloud server may return an incorrect transformed ciphertext or even a random value in order to save its computation resource. In order to detect such malicious behave of the decryption cloud server, many verifiable attribute-based encryption schemes with outsourced decryption schemes [7], [8], [9], [10] were proposed. In these schemes [7], [8], [9], [10], redundant information was added into the original ABE ciphertext and the transformed ciphertext to verify the correctness of the decryption cloud server's output. Whenever the decryption cloud server returned an invalid transformed ciphertext, it will be detected by the mobile device.

However, such a process will increase the decryption cloud server's computational cost of the transformation. Moreover, the previous schemes [7], [8], [9], [10] only achieve verifiability which means it only enables the mobile device to verify whether the transformed ciphertext is correct. A malicious mobile device may claim that the decryption cloud server has returned an incorrect transformed ciphertext even if the decryption cloud server has returned a correct one in order to refuse to pay for the decryption service provided by the decryption cloud server. They cannot achieve exemptibility which ensures the decryption cloud server to escape from a malicious accusation if it has returned a correct transformed ciphertext. The exemptibility is a critical property in outsourced computation cloud service model, without which a customer may make a malicious claim that the decryption cloud server has behaved unhonestly, and thus refuse to pay for the decryption cloud service. In the previous schemes, as the transformed ciphertext can only be privately verified by the mobile device, it may claim that the decryption cloud server has returned an incorrect transformed ciphertext to get compensation from the decryption cloud server even if the decryption cloud server has returned a correct transformed ciphertext. What's more, the current cloud service works in a prepaid paradigm model [11], [12], which means even if the decryption cloud server returned an incorrect transformed ciphertext, the user has already paid the cloud server for the decryption service. It lacks a mechanism to achieves the fairness property which means that the decryption cloud server will get paid if and only if it has returned a correct transformed ciphertext.

A. Motivation

Although the existing verifiable attribute-based encryption with outsourced decryption schemes can detect the malicious decryption cloud server from cheating, they relay on the extra redundant information which increases the computation cost of the decryption cloud server. Moreover, the current OABE schemes cannot achieve the exemptibility which enables the decryption cloud server escape from a wrong accusation if it has returned a correct transformed ciphertext. Additionally, it lacks a mechanism to achieve the fairness property which means the decryption cloud server will get the agreed reward from the user if and only if it has returned the correct transformed ciphertext. However, efficient detection, exemptibility and fairness are very important for the current cloud service paradigm. All of these concerns motivate us to design an attribute-based encryption mechanism with outsourced decryption that:

1) achieves the fairness property, which ensures that the decryption cloud server will get the reward if and only if it has returned the correct transformed ciphertext.
2) does not introduce redundant information which will bring extra burden for the decryption cloud server and the user to check the validity of the transformed ciphertext;
3) achieves reliable outsource decryption that means the verifiability and exemptibility are both guaranteed. It should not only enables the mobile device to detect the misbehave of the decryption cloud server but also prevents the decryption cloud server from wrong accusation;

With the motivation, we propose an attribute based encryption with reliable outsourced decryption scheme using smart contract in the Ethereum blockchain platform [13].

B. Related Work

The concept of attribute-based encryption was first introduced by Sahai et al. [1] in which a user's identity is generalized as a set of descriptive attributes. Since attribute-based encryption enables fine-grained access control on the encrypted data, it has been widely adopted in many scenarios such as the electronic personal health records system [14] and cloud storage system [15]. Attribute-based encryption can be categorized into two types, key-policy attribute-based encryption (KP-ABE) [2] and ciphertext-policy attribute-based encryption (CP-ABE) [3]. In a KP-ABE scheme [2], a user's private key is associated with an access policy and the ciphertext is generated with an attribute set. While in a CP-ABE scheme [3], a user's private key is associated with an attribute set and the ciphertext is generated with an access policy. A ciphertext can be decrypted by a user's private key if and only if the attribute set satisfies the access policy. In traditional attribute-based encryption schemes [2], [3], the decryption cost is linear with the size of the access policy/attribute set, which is not acceptable for a user with resource constrained mobile device to decrypt an ABE ciphertext. To enable a user with a mobile device to retrieve his data from the outsourced encrypted ABE ciphertext, Green et al. [5] introduced the notion of attribute-based encryption with outsourced decryption where a mobile device outsourced most of its decryption work to a decryption cloud server without revealing the plaintext. In OABE [5], the decryption cloud server returns a transformed (partially decrypted) ciphertext to the mobile device and finally the mobile device can decrypt the transformed ciphertext at a very low computation cost.

However, this pioneering work [5] lacks a mechanism to verify whether the decryption cloud server has returned a correct transformed ciphertext. Recently, Lai et al. [7] proposed an attribute-based encryption with verifiable decryption scheme. In their scheme [7], a ciphertext of a random message and a commitment are added to the original ciphertext which leads the original ciphertext size, transformed ciphertext size and computation cost of the decryption cloud server almost double of that in Green et al.'s work [5]. Following their idea, Qin et al. [9] and Lin et al. [10] presented two OABE schemes with more efficient verification. Subsequently, Li et al. [8] proposed the notion of attribute-based encryption with both outsourced encryption and decryption. In their scheme, the extra redundant information was also needed to achieve the verifiability of the outsourced decryption. Moreover, their work only supports the threshold access policy while the schemes [5], [7], [9], [10] achieve any monotonic access policy. Following their work, Ma et al. [16] proposed an attribute-based encryption with both outsourced encryption and decryption that supports any monotonic access policy. In their scheme, the authors pointed out that their achieves the verifiability and exemptibility. However, the receiver needs to reveal his private key to decrypt the transformed ciphertext and then anyone can check whether the decryption server has returned the correct transformed ciphertext. The scheme [16] relays on the same verifiable mechanism with [7] and cannot achieve the exemptibility since the receiver may reveal an incorrect private and claim that the decryption cloud server has returned incorrect transformed ciphertext. All of these previous works rely on the extra redundant information to check the validity of the transformed ciphertext. However, this will increase the computation work of the decryption cloud server. Additionally, none of the previous schemes can achieve exemptibility and fairness. To address this problem, Cui et al. [17] introduced an attribute-based encryption scheme with outsourcing the decryption to a smart contract. In their scheme, the recipient issues a smart contract that contains the decryption task. Any miner in the blockchain platform can issue a transaction that calls the smart contract and get the related reward. However, outsourcing heavy computation to a smart contract may lead to the verifier's dilemma problem [18] which will be shown in detail in Section V-B.

To achieve reliable outsourced computation, various method based on verifiable computation [19], [20], audit technique [21], replication verification [22] and zero-knowledge proof [23] have been proposed. However, the verifiable computation method relies on the secure multi party computation [24] or fully homomorphic encryption [25] techniques which often introduce large computation cost. The audit technique requires the user to recompute the outsourced computation cost which is infeasible for mobile devices. Additionally, the zero-knowledge proof method also introduces large computation and communication cost.

C. *Our Contribution*

In this work, for the first time, we propose a reliable attribute-based encryption with outsourced decryption scheme based on smart contracts in the Ethereum platform [13]. In the proposed scheme, no extra redundant information is needed in the original ciphertext and the transformed ciphertext, which makes our scheme much more efficient than the previous OABE schemes. Additionally, our scheme achieves the verifiability and exemptibility property. Finally, the proposed scheme achieves the fairness property which means the decryption cloud server will get the preset reward if and only if it has returned a correct transformed ciphertext. At the first glance, the approach seems straightforward by using smart contract. However, there are some technical challenges to adopt smart contracts to OABE schemes. We will illustrate these challenges in Section V.

II. Background

Ethereum. Ethereum is an emerging decentralized blockchain platform [13]. In the Ethereum network, the nodes, called miners, verify the transactions and compete to solve the mining puzzle. Whenever a miner finds the answer to the mining puzzle, he broadcasts a new block that contains the answer of the mining puzzle and transactions he collected to the whole network. All the nodes receiving the broadcasted block validate the answer to the mining puzzle and the transactions contained in the block. Once the block is added to the Ethereum network, it cannot be retrospectively removed or modified, and the successful miner will receive a reward of cryptocurrency. In the Ethereum, transactions and states are transparent to all nodes. The consensus protocol of the Ethereum makes sure that the transactions were verified according to the predefined rules and all nodes will agree on a final state.

Smart Contracts. A smart contract is an automatically executable program based on its input and inner logic. It consists of a piece of script code, balance and storage. Once deployed, the codes of a smart contract cannot be modified. A smart contract is created by a node issuing a transaction to the Ethereum network. Once created, the smart contract will be bound with a unique address. Any node can call the smart contract by sending a transaction to the smart contract's address and thus change the state of the smart contract.

Gas System. In Ethereum, a smart contract is compiled into opcodes and stored in the Ethereum platform. Each opcode will consume a predefined amount of *gas* [13]. The smart contract is initiated and called by a node issuing a transaction. When issuing a transaction that initiates or calls a smart contract, the sender should specify an upper bound gas, *gasLimit*, for executing the transaction, and *gasPrice* for each unit of gas. The sender can issues such a transaction successfully only when his balance is larger than $gasLimit * gasPrice$. The executing of the transaction will halt if the gas is exhausted. In such a manner, executing of the smart contract will finally halt and the Ethereum system is prevented from the denial of service (DoS) attacks.

III. System Overview

A. *System Architecture*

Fig. 2 outlines our system architecture. Our system architecture consists of three entities, ie. the data sender, cloud server and receiver. The data sender (not shown in Fig. 2) generates ABE ciphertexts and sends them to the cloud server. The cloud server stores the receiver's ciphertexts and uploads them to a

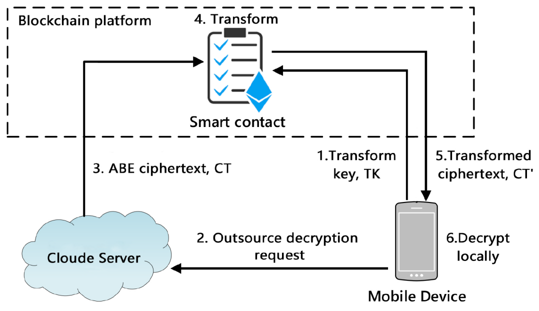


Fig. 2. A system overview of the proposed scheme.

smart contract at the receiver's request. The receiver (equipped with a mobile device) installs a smart contract, generates a transformation key and uploads it to the smart contract, and decrypts the transformed ciphertext locally.

In an attribute-based encryption scheme, we denote (I_{key}, I_{enc}) as the input of to the key generation and encryption algorithms respectively. In a key-policy attribute-based encryption scheme, $(I_{key}, I_{enc}) = (\mathbb{AS}, S)$, where $\mathbb{AS}$ represents an access structure and S is an attribute set. While in a ciphertext-policy setting, $(I_{key}, I_{enc}) = (S, \mathbb{AS})$. Similar to the previous OABE scheme, our scheme consists of the following six algorithms.

- $Setup(\lambda, U)$ is executed by a trusted authority. On input a security parameter λ and the universal attribute description U, it outputs a master secret key msk and the common public parameters PP.
- $KeyGen(PP, msk, I_{key})$ is also executed by the trusted authority. On input the public parameters PP, mast secret key msk and an attribute set (resp. access structure) I_{key}, outputs a private key sk.
- $Encrypt(PP, K, I_{enc})$ is executed by a sender to generate an ABE ciphertext which is then outsourced in the cloud server. On input the public parameters PP, an encapsulated key K and an access structure (resp. an attribute set) I_{enc} as input. It outputs a ciphertext CT.
- $GenTK_{out}(PP, sk)$ is executed by the receiver who is equipped with a mobile device. On input the public parameters PP and his private key sk, outputs a transformation key TK and the corresponding retrieve key RK.
- $Transform_{out}(PP, CT, TK)$ is executed by nodes in the Ethereum network. On input the public parameters PP, a ciphertext CT and a transformation key TK, a node computes a meta element T_i of the transformed ciphertext CT and issues a transaction that uploads the meta element T_i to the smart contract. Once all the meta pieces have been uploaded, the smart contract computes the transformed ciphertext CT'.
- $Decrypt_{out}(PP, CT', RK)$ is executed by the receiver locally. On input the public parameters PP, a transformed ciphertext CT' and a retrieving key RK, outputs the encapsulated key K.

In our scheme, we adopt the notion of key encapsulated mechanism (KEM), where a symmetric encryption key K is encrypted using the attribute-based encryption and the plaintext is encrypted under the symmetric key K with a symmetric key encryption algorithm (e.g., AES). In our architecture, the receiver leveraged the OABE technique to retrieve the symmetric key K from the KEM ciphertext. Then, the receiver downloads the symmetric key encryption ciphertext from a cloud server and decrypts the ciphertext with the symmetric key K. In our paper, we only consider the OABE part.

B. System Requirements

Confidentiality. The confidentiality property indicates that an outside attack even with the transformation key cannot disclose the encapsulated key K. In this article, we adopted the security notion of against chosen plaintext attack (CPA) secure which is described in [5]. In our KEM setting, it requires an attacker cannot reveal the encapsulated key from the attribute-based ciphertext.

Soundness. In our design, we integrate the verifiability and exemptibility properties into the notion of soundness. The verifiability indicates that the entity who returned an invalid transformed ciphertext will be caught, and the exemptibility means that the the entity cannot be accused when it returned a correct transformed ciphertext. All the previous works [7], [8], [9], [10] resort to adding redundant information to the ABE ciphertext and the transformed ciphertext to achieve the verifiability, and lacks a mechanism to achieve the exemptibility. In this work, we extend the notions by claiming that the correctness of the transformed ciphertext is guaranteed by the Ethereum consensus protocol. In our mechanism, the extra redundant information for verifying the transformed ciphertext is unneeded.

Note that, the security definition of exculpability in [16] is quite similar to exemptibility in our work. Both of them ensure that the entity who has returned a correct partial decrypted ciphertext cannot be accused. The difference is that the underlying methodologies are different. In [16] the key derivation function and Pedersen commitment are leveraged while the consensus protocol is adopted in our scheme to achieve this property.

Fairness. The fairness property is critical for the outsourcing cloud service system. Generally, the notion of fairness in our scheme guarantees that

- the client gets the correct transformed ciphertext as long as he pays for the transformation work.
- the users who contribute to the computation work of the meta elements of the transformed ciphertext for the receiver get the reward as long as they returned the valid meta elements that are used to compose the transformed ciphertext.

The fairness property incentives each party to perform the protocol honestly. Otherwise, they will get nothing.

IV. Preliminaries

In this section, we present some basic definitions and the security assumption for our scheme.

A. Bilinear Map

Let G and G_T be two multiplicative cyclic groups of prime order p. A bilinear pairing $e : G \times G \longrightarrow G_T$ is a map, which satisfies the following properties [26]:

1) $e(u^x, v^y) = e(u, v)^{xy}$ for all $x, y \xleftarrow{R} Z_p^*$ and $u, v \in G$;
2) $e(u, v) \neq 1$.
3) $e(u, v)$ can be computed in polynomial time for all $u, v \in G$.

B. Access Structure

Definition 1 (Access Structure [3]). Let $\mathcal{P} = (P_1, P_2, \ldots, P_n)$ be a set of parties. A collection $\mathbb{AS} \subseteq 2^{\mathcal{P}}$ is monotone if $\forall B, C$: if $B \in \mathbb{AS}$ and $B \subseteq C$ then $C \in \mathbb{AS}$. An monotone access structure is a collection $\mathbb{AS}$ of non-empty subsets of $\mathcal{P}$, $\mathbb{AS} \subseteq 2^{\mathcal{P}} \setminus \{\emptyset\}$. The sets in $\mathbb{AS}$ are called the authorized sets, and the sets not in $\mathbb{AS}$ are called unauthorized sets.

In this article, we focus on monotone access structure. As described in [27], any monotone access structure can be represented by a linear secret-sharing scheme (LSSS).

C. Linear Secret-Sharing Schemes

Definition 2 (Linear Secret-Sharing Schemes [27]). A secret-sharing scheme $\prod$ over a set of parties $\mathcal{P}$ is called linear (over Z_p) if

1) The shares for each party form a vector over Z_p.
2) There exists a matrix M with l rows and n columns called the share-generating matrix for $\prod$. For all $i = 1, 2, \ldots, l$ the i^{th} row of M is labeled by a party $\rho(i)$ (ρ is a function from $1, 2, \ldots, l$ to $\mathcal{P}$). When we consider the column vector $\mu = (s, r_2, \ldots, r_n)$, where $s \in Z_p$ is the secret to be shared and $r_2, \ldots, r_n \in Z_p$ are randomly chosen, then $M\mu$ is the vector of l shares of the secret s according to $\prod$. The share $(M\mu)_i$ belongs to party $\rho(i)$.

Note that every LSSS according to the above definition achieves the linear reconstruction property [27]. Suppose that $\prod$ is an LSSS for the access structure $\mathbb{AS}$. Let $S \in \mathbb{AS}$, which means S satisfies the access structure $\mathbb{AS}$ denoted as $S \models (M, \rho)$, be any authorized set, and let $I \subset \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. There will exist constants $\{\omega_i \in Z_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \cdot \lambda_i = s$, if $\{\lambda_i\}$ are valid shares of any secret s according to $\prod$. As showed in [27], $\{\omega_i\}$ can be found in polynomial time in the size of the share-generating matrix M using M and I.

V. Challenges

In this section, we will first discuss some challenges and our idea of using the Ethereum smart contract to realize reliable outsourced decryption of attribute-based encryption scheme. Intuitively, the existing OABE schemes [7], [8], [9], [10] can be directly adapted to the Ethereum environment by replacing the decryption cloud server with a smart contract. Unfortunately, this trivial approach will be infeasible due to some innovative features of smart contracts in the Ethereum. The main challenges are as follows.

A. Gas Limitations

In Ethereum, each transaction that calls a function of a smart contract is restricted with an upper bound amount of consumed gas, called *gasLimit*. Each operation in the calling, including sending and storing data and executing computations will consume a fixed amount of gas. Under this restriction, the designated transaction has extremely limited storage and computation steps. Therefore, since the size of the attribute-based ciphertext and the transformation key grow linear with the size of the access structure (resp. attribute set) and the attribute set (resp. access structure), we are motivated to divide the attribute-based ciphertext and transformation key into smaller ones in order not to exceed the gas limit. Our general idea is that we partition the attribute-based ciphertext and the transformation key into several blocks and upload them to smart contracts with sufficient transactions such that each transaction will consume less than *gasLimit* gas.

Moreover, when calling a smart contract, the sender issues a transaction associated with *gasPrice* which specifies the cryptocurrency he is willing to purchase the gas. It is required that the account balance of the sender who issues the transaction is larger than the gas cost for executing the transaction. Otherwise, the execution will abort immediately while the consumed gas cannot be refunded. Thus, it is a challenge to design smart contracts regarding gas cost. We should enable the partial decryption functionality in smart contracts to consume less gas than the sender's account balance.

B. Verifier's Dilemma

The second challenge is the verifier's dilemma problem [18]. In Ethereum, when a transaction that call the function of a smart contract is broadcast to the network, the miner needs to verify the correctness of the transaction. Since the verification of the partial decryption result is significant expensive especially when the access structure size is large, a rational miner may skip the verification of these transactions so as to stay ahead in the competition of mining a new block. Thus, the transaction that calls the smart contract to generate a transformed ciphertext will not be included into the blocks by the rational miners. To mitigate this problem, our main idea is to decompose the transformation task to pieces of meta computations that can be efficiently verified. Thus, the verification work of each piece meta transformation result will not exceed the critical value and rational miners will include transactions that computing a meta result into their blocks.

C. Elliptic Curve Cryptography in Ethereum

The existing ABE with outsourced decryption schemes rely on the bilinear pairing. However, the current Ethereum platform does not support the bilinear pairing computation directly. To overcome this problem, we designed a blockchain platform based on the Ethereum and integrated the PBC cryptography library [28]. In the designed blockchain platform, we integrated the bilinear pairing computation in the built-in functions. Moreover, we construct a precompiled contract that can be called to enable pairing computing, and open source it publicly.[1]

[1]http://82.156.224.174:7070/tenon

Note that, in the previous work, the smart contract is leveraged as a trusted third party. The receiver can establish a smart contract in which the transformation computing task is published. However, this will face the gas limitation, verifier's dilemma and pairing computation problems as described. In this work, to overcome these problems. We first proposed the meta-computing technique that decomposes the transformation computing task into several meta-computing tasks. By doing so, the gas limitation and verifier's dilemma problems are overcome. Additionally, we designed a blockchain platform to enable the pairing computation in smart contract.

VI. Proposed Construction

In this section, we will present our CP-ABE with outsourced decryption scheme based on the smart contracts in Ethereum. Our construction is built on Green et al.'s CP-ABE with outsourced decryption scheme [5]. We will first present our modified attribute-based encryption with outsourced decryption scheme based on the smart contract, and then we present the concrete smart contract protocol for outsourcing decryption CP-ABE scheme.

A. Proposed CP-ABE With Outsourced Decryption Scheme

Our proposed CP-ABE with outsourced decryption scheme is as follows.

- $Setup(\lambda, U)$. The public key generator (PKG) generates the a bilinear pairing parameters (G, G_T, e, g), where g is a generator of cyclic multiple group G of prime order p. The PKG initializes a hash function $F: \{0,1\}^* \to G$ that maps an attribute to an element in G. Then it chooses random elements $a, \alpha \in Z_p$. Finally, it sets the master secret key as $msk = g^{\alpha}$ and the public parameters $PP = (g, e(g,g)^{\alpha}, g^a, F)$.
- $KeyGen(PP, msk, S)$. On input the master secret key $msk = g^{\alpha}$ and an attribute set S, the PKG chooses a random element $s \in Z_p^*$ and computes
$$R = g^{\alpha} g^{as}, \quad L = g^s, \quad \forall x \in S, R_x = F(x)^s.$$
Output the private key $sk = (S, R, L, \{R_x\}_{x\in S})$.
- $Encrypt(PP, K, (M, \rho))$. On input a symmetric key K that needed to be encapsulated, and an access structure (M, ρ) where M is an $l \times n$ matrix and ρ associates each row of M to an attribute. The algorithm selects a random vector $\vec{v} = (r, m_2, \ldots, m_n) \in Z_p^n$. For $1 \le i \le l$, computes $\lambda_i = \vec{v} \cdot M_i$, where M_i is the ith row of M. Then it chooses $t_1, \ldots, t_l \in Z_p$, and outputs the ciphertext $CT = ((M, \rho), C, C', \{C_i, D_i\}_{i \in \{[1,\ldots,l]\}})$, where
$$C = K \cdot e(g,g)^{\alpha r}, \qquad C' = g^r,$$
$$C_i = g^{a\lambda_i} \cdot F(\rho(i))^{-t_i}, \qquad D_i = g^{t_i}.$$
- $GenTK_{out}(PP, sk)$. On input a private key sk, the algorithm selects a random element $\theta \in Z_p^*$ and sets $RK = \theta$, $TK = (S, R', L', \{R'_x\}_{x\in S})$ where
$$R' = R^{1/\theta}, \quad L' = L^{1/\theta}, \quad \forall x \in S, R'_x = R_x^{1/\theta}.$$
- $Transform_{out}(PP, CT, TK)$. On input a ciphertext $CT = ((M, \rho), C, C', \{C_i, D_i\} \quad i \in \{[1,\ldots,l]\})$ and a transformation key $TK = (S, R', L', \{R'_x\}_{x\in S})$. The receiver first finds a set $W = \{\omega_i \in Z_p\}_{i\in I}$ such that $\sum_{i\in I} \omega_i \lambda_i = r$, where I is the set $I = \{i : \rho(i) \in S\}$ if S satisfies (M, ρ), and abort if S doesn't satisfy (M, ρ). The receiver establishes a smart contract and then partitions the ciphertext CT, the transformation key TK and W into n parts and uploads each part to the smart contract through transactions such that each transaction fee will not exceed the gasLimit.
Miners in the Ethereum can execute the procedure $Meta\_compute(i, CT, TK, W)$ which is defined as:
  - The $Meta\_compute$ procedure outputs an element $T_i = e(C_i^{\omega_i}, L') \cdot e(D_i^{\omega_i}, R'_{\rho(i)})$.

  Next, miners can send their meta results T_i to the smart contract by issuing transactions that call the smart contract. When all $T_i, i \in I$ has been uploaded to the smart contract, the state of the smart contract is set as $Meta\_Computed$, which indicates that the receiver can retrieve the plaintext by composing these meta elements. Then, the miners compute $T = e(C', R') / \prod_{i\in I} T_i$. Finally, the transformed ciphertext is set as $CT' = (C, T)$.
- $Decrypt_{out}(PP, CT', RK)$. On input a transformed ciphertext $CT' = (C, T)$, and a retrieve key $RK = \theta$, the receiver computes $K = C/T^{\theta}$.

Discussion. In the proposed scheme, we set the computation of each T_i as a meta computation. Every node in the Ethereum network can contribute to a meta computation and will receive a predefined reward the receiver deposit in the smart contract as long as the returned meta computation result is correct.

B. Smart Contract for CP-ABE With Outsourced Decryption

Fig. 3 presents the design of smart contract of our CP-ABE with outsourced decryption scheme. In the contract, we use the $SetState$ function to enable the receiver to outsource the decryption for multiple CP-ABE ciphertexts and to update the transformation key.

Note that \$denotes the cryptocurrency, $Creat.gasLimit$, $UploadTK.gasLimit$, $UploadCT.gasLimit$, $UploadShare.gasLimit$, $ComputeMe\text{-}ta.gasLimit$ and $SetState.gas Limit$ denote the $gasLimit$ for each function.

C. Security Proof

Confidentiality. The confidentiality of our proposed CP-ABE with outsourced decryption scheme is based on the security of the CP-ABE with outsourced decryption scheme in [5].

Theorem 1. Suppose that the CP-ABE scheme in [5] is IND-CPA secure, then our proposed CP-OABE scheme is CPA secure in the KEM setting.

Proof. Suppose there exists a probability polynomial-time (PPT) adversary $\mathcal{A}$ that can break the CPA security of the proposed CP-OABE scheme using smart contract, then we can

Contract **CP-OABE**

Init: Set the initial state $State := Init$. Call the $Setup$, $Encrypt$, $KeyGen$,and $GenTK_{out}$ algorithms to get the public parameters PP, ciphertext CT and transformation key TK. Also set $\widetilde{S} := \emptyset$, $\widetilde{Z} := \emptyset$, $\widetilde{W} := \emptyset$ and $\widetilde{I} := \emptyset$.

Create: Upon receiving ("Create", $\$deposit$, $\$reward$, T_{end}, pk_C) from the receiver pk_R that creates the contract. Note that, pk_C indicates the address of the cloud server that will upload the ciphertext later. The smart contract:

1) Assert $state = Init$.
2) Assert $\$deposit \geq \$reward$.
3) Assert $balance[R] \geq \$deposit + Create.gasLimit * gasPric$.
4) $balance[R] := balance[R] - \$deposit - Creat.gas * gasPrice$.
5) $State := Created$.
6) send $Creat.gas * gasPrice$ to the miner.

UploadTK: Upon receiving ("UploadTK", $TK_{S'}$, TK_{index}) from the receiver pk_R that uploads the transformation key TK to the contract. Note that, TK may need to be partitioned into n parts for uploading. If $TK_{index} = 1$ indicates it is first part of TK and $TK_{S'} = (S, R', L', \{R'_x\}_{x\in S'})$ where $S' \subseteq S$. Otherwise for $TK_{index} > 1$, $TK = \{R'_x\}_{x\in S'}$.

1) Assert $state = Created$.
2) Assert that the transaction sender is pk_R.
3) Assert $balance[R] \geq UploadTk.gasLimit * gasPrice$.
4) If $TK_{index} = 1$, add S, R', L'.
5) For each $x \in S'$ add R'_x, and set $\widetilde{S} := \widetilde{S} + x$.
6) If $\widetilde{S} = S$, set $State := TK\_Uploaded$.
7) $balance[R] := balance[R] - UploadTK.gas * gasPrice$.
8) send $UploadTK.gas * gasPrice$ to the miner.

UploadCT: Upon receiving ("UploadCT", $CT_{Z'}$, CT_{index}) from the cloud pk_C that uploads the ciphertext CT to the contract. Note that, CT may need to be partitioned into n' parts for uploading. If $CT_{index} = 1$ indicates it is first part of CT and $CT_{Z'} = ((M,\rho), C, C', \{C_i, D_i\}_{i\in Z'})$, where $Z' \subseteq \{1, \cdots, l\}$. Otherwise for $CT_{index} > 1$, $CT = \{C_i, D_i\}_{i\in Z'}$.

1) Assert $State = TK\_Uploaded$.
2) Assert that the transaction sender is pk_C.
3) Assert $balance[C] \geq UploadCT.gasLimit * gasPrice$.
4) If $CT_{index} = 1$, add $(M,\rho), C, C'$.
5) For each $i \in Z'$ add C_i, D_i, and set $\widetilde{Z} := \widetilde{Z} + i$.
6) If $\widetilde{Z} = \{0, \cdots, l\}$, set $State := CT\_Uploaded$.
7) $balance[C] := balance[C] - UploadCT.gas * gasPrice$.
8) send $UploadCT.gas * gasPrice$ to the miner.

UploadShare: Upon receiving ("UploadCT", W') from the receiver pk_R that uploads the shares $W' \subseteq W = \{\omega_i \in Z_p\}_{i\in I}$ to the contract. Note that, W may need to be partitioned into n'' parts for uploading.

1) Assert $State := CT\_Uploaded$.
2) Assert that the transaction sender is pk_R.
3) Assert $balance[R] \geq UploadShare.gasLimit * gasPrice$.
4) For each $\omega_i \in W'$ add ω_i, and set $\widetilde{W} := \widetilde{W} + \omega_i$.
5) If $\widetilde{W} = W$, set $State := Share\_Uploaded$.
6) $balance[R] := balance[R] - UploadShare.gas * gasPrice$.
7) send $UploadShare.gas * gasPrice$ to the miner.

ComputeMeta: Upon receiving ("ComputeMeta", i ,T_i), where $i \in (I - \widetilde{I})$, from a node N.

1) Assert $State = Share\_Uploaded$.
2) Assert $balance[N] \geq ComputeMeta.gasLimit * gasPrice$.
3) Assert $\$deposit \geq \$reward$.
4) If $T_i = Meta\_compute(i, CT, TK, W)$, then
   - $\widetilde{I} := \widetilde{I} + i$;
   - send $\$reward$ to node N;
   - $\$deposit := \$deposit - \$reward$;
5) If $\widetilde{I} = I$, then
   - compute $T = e(C', R')/\prod_{i\in I} T_i$;
   - set $State := Meta\_Computed$;
6) $balance[N] := balance[N] - ComputeMeta.gas * gasPrice$.
7) send $ComputeMeta.gas * gasPrice$ to the miner.

SetState: Upon receiving ("SetState",$Type$):

1) Assert $State \neq Aborted$.
2) Assert that the transaction sender is pk_R.
3) Assert $balance[R] \geq SetState.gasLimit * gasPrice$.
4) Case "Type" =
$$\begin{cases} \text{“}UpdateShare\text{”}, & \text{set } State := CT\_Uploaded, \\ & \text{and } \widetilde{I} := \emptyset; \\ \text{“}UpdateCT\text{”}, & \text{set } State := TK\_Uploaded, \\ & \widetilde{Z} := \emptyset \text{ and } \widetilde{I} := \emptyset; \\ \text{“}UpdateTK\text{”}, & \text{set } State := Created, \widetilde{S} := \emptyset, \\ & \widetilde{Z} := \emptyset \text{ and } \widetilde{I} := \emptyset; \\ \text{“}Abort\text{”}, & \text{set } State := Abort; \end{cases}$$
5) $balance[R] := balance[R] - SetState.gas * gasPrice$.
6) send $SetState.gas * gasPrice$ to the miner.

Timer: If the current $T > T_{end}$:

- $balance[R] := balance[R] + \$deposit$.
- $State := Abort$.

Fig. 3. Contract design for CP-OABE.

build a simulator $\mathcal{B}$ that can break the CPA security of the CP-OABE scheme in [5].

Init. The simulator $\mathcal{B}$ calls $\mathcal{A}$ to get the challenge access structure (M^*, ρ^*), and passes it to the challenger $\mathcal{C}$ in the OABE scheme in [5] as the challenge structure it wishes to be challenged.

Setup. $\mathcal{B}$ returns the public parameters $PP = (g, e(g,g)^\alpha, g^a, F)$ that got from the challenger $\mathcal{C}$ to the adversary $\mathcal{A}$.

Query Phase 1. The adversary $\mathcal{A}$ makes the following queries:

- $KeyGen(S)$ query: Whenever $\mathcal{A}$ issues $KeyGen$ queries with the restriction that S doesn't satisfies (M^*, ρ^*). The simulator $\mathcal{B}$ passes it to the challenger $\mathcal{C}$ and returns the result that answered by $\mathcal{C}$ to $\mathcal{A}$.

Challenge. $\mathcal{B}$ calls $\mathcal{A}$ to get two equal length encapsulated key (K_0, K_1) and sends to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ generates the challenge ciphertext CT^*_b, where $b \in \{0,1\}$ and

TABLE I
FUNCTIONALITY COMPARISON WITH [7], [8], [9], [10], [16], [29]

Schemes	confidentiality?	verifiability?	exemptibility?	fairness?	Access Policy
Scheme [7]	✓	✓	✗	✗	any monotonic policy
Scheme [8]	✓	✓	✗	✗	threshold policy
Scheme [9]	✓	✓	✗	✗	any monotonic policy
Scheme [10]	✓	✓	✗	✗	any monotonic policy
Scheme [16]	✓	✓	✗	✗	any monotonic policy
Scheme [29]	✓	✓	✗	✓	any monotonic policy
Our Scheme	✓	✓	✓	✓	any monotonic policy

CT_b^* is an CP-ABE KEM ciphertext of K_b, and sends CT_b^* to $\mathcal{B}$.

Query Phase 2. $\mathcal{A}$ continues to make queries as in phase 1.

Guess. $\mathcal{B}$ passes CT_b^* to $\mathcal{A}$ and gets the response b. $\mathcal{B}$ outputs b as its guess.

Note that, the simulation of B is perfect if the above game does not abort. The advantage of the simulator that wins the CPA game is the same as the adversary $\mathcal{A}$ that breaks the CPA security of the CP-OABE in [5]. Thus completes the proof of Theorem 1.

Soundness. The soundness of our scheme is guaranteed as long as the underlying Ethereum consensus protocol is secure. All transactions that contain the meta element are public and verified by all nodes in the Ethereum. The transformation of each meta element is stored as a state in the smart contract. When a miner executes transactions that call the smart contract and changes the states, all nodes in Ethereum will verify the operation of this transaction.

Fairness. The fairness of our scheme is twofold. From the receiver's perspective, he must deposit enough cryptocurrency first for the executing of the outsourced transformation. If a node returns a correct meta element of the transformation ciphertext, the specified amount of cryptocurrency will be transferred to him from the deposit according to the smart contract automatically. Thus, the receiver cannot cheat by refusing to pay the node if the node has returned a correct meta element of transformation ciphertext. On the other hand, the validity of a transaction that contains the meta element of transformation ciphertext will be verified by all nodes in Ethereum. Only if all the transactions of a new block are correct, the new block can be added to the Ethereum network. Thus the proposed smart contract based OABE scheme achieves the fairness property as long as the Ethereum consensus protocol is guaranteed.

VII. PERFORMANCE AND EVALUATION

A. Functionality Comparison

We compare our scheme with the previous OABE schemes [7], [8], [9], [10], [16], [29] in terms of the access policy, confidentiality, verifiability, exemptibility and fairness properties. Table I shows that all the schemes achieve confidentiality and verifiability. And most of the schemes support any monotonic policy express, except scheme [8] that only supports the threshold policy express. However, only the scheme [29] and our scheme achieve fairness properties. Moreover, only our scheme achieves exemptibility properties.

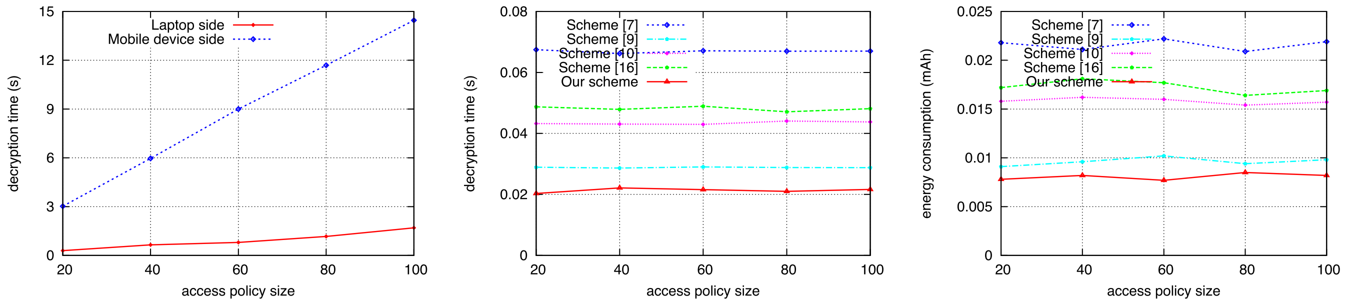
B. Computation and Communication Cost

We leverage a laptop with Intel(R) Core(TM) i5-8265 U CPU @1.60GHZ 1.80GHZ 16 GB RAM to initialize the cloud server. The operating system of the cloud server is Linux Mint 18.1 Serena and the programming language is Java 1.8. The receiver is equipped with a Huawei HMA-AL00 mobile phone with kirin980 CPU, 6 GB RAM, 4000mAh Battery and HarmonyOS operating system. We use the JPBC library [28] to execute the bilinear pairing computation. We use the BLS curve and the security parameter is set 1,024 b. In the experiment, the access policy is set as an AND gate of attributes and its size varies from 20 to 100 step 20. We implemente $SHA-256$ as the hash function in [9] and Pedersen commitment as the commitment algorithm in [10]. For each experiment, we repeat 1,000 times to get the average time and energy consumption. The implementation is open sourced in the github.[2]

1) Decryption Cost Comparison: We first implement the basic ABE scheme [3] on the mobile device and laptop to compare the decryption time on the laptop and mobile device. Then, we compare our scheme with previous OABE schemes [7], [9], [10], [16] that work in a monotonic access policy manner in terms of the execution time of the decryption algorithm which is executed on the mobile phone.

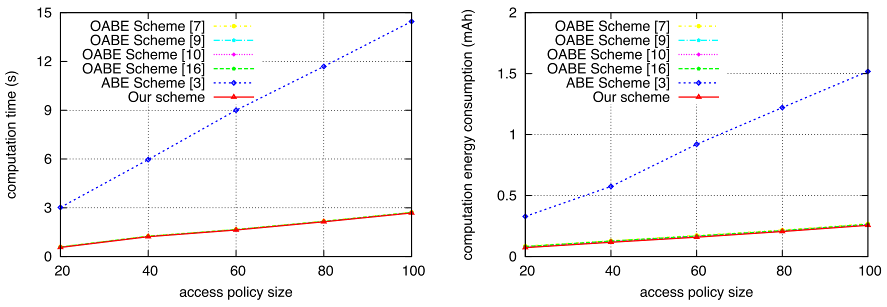
Fig. 4(a) shows that the decryption time of the ABE scheme on both laptop and mobile device increase linearly with the access policy size. The decryption time on the mobile device varies from about 3.03 s to 14.46 s. This means that, when the ABE scheme [3] is deployed on a mobile device, decrypting a ciphertext takes as large as 14.46 s when the access policy size is 100. However, as shown in Fig. 4(b), the decryption time on the mobile device is constant and does not linear with the access policy size when OABE schemes are deployed. The decryption time is only about 0.02 s in our scheme and

[2]https://gitee.com/neo0627/abe



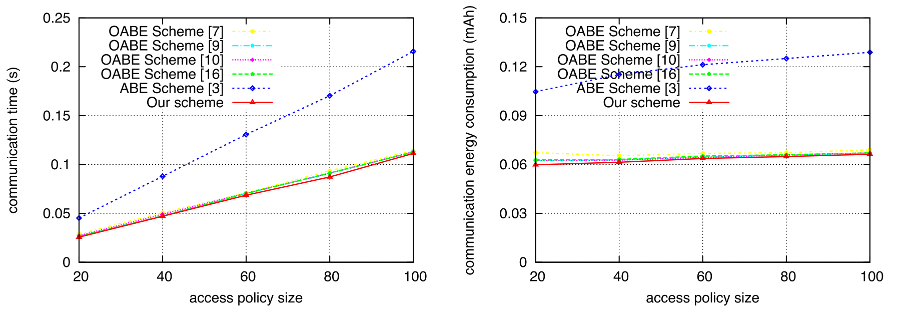
(a) ABE decryption time on both devices (b) OABE decryption time on mobile device (c) OABE decryption energy consumption on mobile device

Fig. 4. ABE and OABE decryption time and energy consumption.



(a) Mobile device computation time (b) Mobile device computation energy consumption

Fig. 5. Computation cost on mobile device.

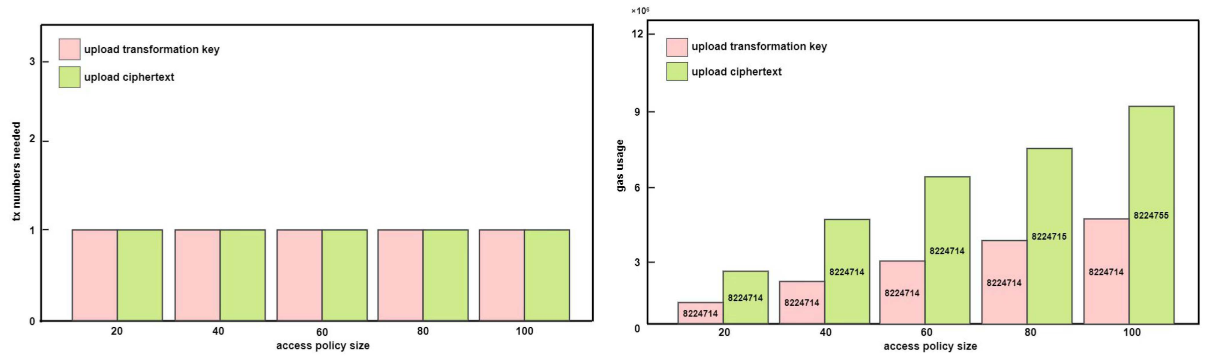


(a) Mobile device communication time (b) Mobile device communicaiton energy consumption

Fig. 6. Communication cost on mobile device.

0.07 s in the scheme [7]. Additionally, as shown in Fig. 4(c), the mobile device energy consumption is constant in OABE schemes [7], [9], [10], [16] and our scheme. The mobile device energy consumption in our scheme is about 0.008mAh which is only about 0.002% of the battery capacity. Moreover, our scheme outperforms the previous OABE schemes [7], [9], [10], [16] in terms of decryption time. This is because no redundant elements are needed in our scheme. While in the previous OABE schemes [7], [9], [10], [16], redundant elements are added in the ciphertext to achieve the verifiability and exemptibility properties.

2) Computation Cost Comparison on Mobile Device: Fig. 5 shows the computation time and energy consumption comparison of the ABE scheme [3], OABE schemes [7], [9], [10], [16] and our scheme on the mobile device. In the ABE scheme [3], the mobile device is in charge of executing the decryption algorithms. While in OABE schemes [7], [9], [10], [16] and our scheme, the mobile device is in charge of the transformation key generation and local decryption. Fig. 6 shows that the computation time and energy consumption on the mobile device side of OABE schemes [7], [9], [10], [16] and our scheme is much smaller than that of the ABE scheme [3]. Even when the access policy size is 100, the total computation time of the mobile device in our scheme is only 2.68 s which is much smaller than that in the ABE scheme [3]. Additionally, the total energy consumption of the mobile device in our scheme is only 0.26mAh when the access policy size is 100.



(a) Transactions needed for uploading

(b) Gas usage for uploading

Fig. 7. Transactions and gas usage of uploading transformation key, ciphertext and shares.

3) Communication Cost Comparison on Mobile Device: To evaluate the communication cost of the mobile device, we set up a local area network with 125 Mbps bandwidth between the mobile device and the laptop. Fig. 6(a) shows the communication time and energy consumption comparison on the mobile device side. In the basic ABE scheme [3], the mobile device's communication mainly relies on downloading the ciphertext from the cloud server. However, in OABE schemes [7], [9], [10], [16] and our scheme, the communication time of the mobile device relies on uploading the transformation key and downloading the transformed ciphertext. As shown in Fig. 6(a), the communication time of OABE schemes [7], [9], [10], [16] and our scheme is much smaller than that of the basic ABE scheme [3]. Additionally, Fig. 6(b) shows that the energy consumption of our scheme and OABE schemes [7], [9], [10], [16] is also much smaller than that of the basic ABE scheme [3]. Moreover, the communication time and energy consumption of our scheme are a bit lesser than that of the previous OABE schemes [7], [9], [10], [16]. This is because that no redundant information is needed to achieve the verifiability and exemptibility properties in our scheme.

C. Implementation of Smart Contract

1) Transactions Needed: In this subsection we implemented our designed CP-OABE smart contract which is described in Section VI-B. To evaluate the number of transactions and gas usage needed to upload the transformation key TK and the ciphertext CT, we implemented our contract on the Ethereum official test network *Goerli Testnet* network. The receiver's account address and the smart contract address in the *Goerli Testnet* network are

- 0xd6e9eecbde35036cf86b23eb5373a3c8ab89694e
- 0x857ff4dfcd6b3e8cc125b545485623088a8e5091.

Detailed transactions information of the experiment is published on the official *Etherscan* website: https://goerli.etherscan.io/address/0xd6e9EEcbde35036cF86B23EB5373a3C8Ab89694e.

Fig. 7(a) shows the number of transactions needed to upload the transformation key and the ciphertext to the smart contract. Fig. 7(b) presents the total gas usage of transactions for uploading the transformation key and the ciphertext, and block numbers (presented in the bars) that transactions were involved. It is shown, in Fig. 7(a), that only one transaction is needed even when the access policy size is 100 for uploading the transformation key and ciphertext that is executed on the mobile device and cloud server respectively. The block numbers involving these transactions are 8224714,8224715 and 8224755.

Note that, in some applications where end-to-end delay matters, the delay of our scheme mainly relies on the consensus time of the underlying blockchain platform. In our scheme, the mobile device is charge of uploading the transformation key. Even when the access policy size is 100, it takes only one round of consensus time. The current consensus time for a new block is about 12 s. It means the average delay for uploading the transformation key when the access policy size 100 is 12 s/2 = 6 s. This is much smaller than the original decryption time of ABE scheme [3] on the mobile device which is about 14.46 s (shown in Fig. 4(a)).

Note that, by adopting the blockchain smart contract, the transformation work is omitted from the cloud server. Thus, the cloud server's computation time is saved.

2) Gas Usage: Moreover, to evaluate the gas usage of the meta computing in our smart contract, we build a blockchain platform based on the Ethereum, and integrate the PBC cryptography library. Additionally, we construct a precompile contract that can be called to enable the pairing computing. Moreover, in order to define a reasonable gas usage for the pairing computation in the smart contract, we adopt the method in EIP-1108 [30]. We install a smart contract in the designed blockchain platform and evaluated the executing time of the basic *ecrecover* precompile and the pairing precompile. To make the execution time more precise, we conduct each experiment 1,000 times to get the average time. The experiment result shows that the performance of the *ecrecover* precompile was measured at 44 microseconds per *ecrecover* invocation, and the *ecrecover* gas usage is officially set as 3,000. Thus, we can compute that for a precompile algorithm's runtime, the price is 68 gas per microsecond. The computation time for pairing precompile consists of a fixed base-time and computing-time per pairing because of the structure of the algorithm. We conduct an experiment and get the base-time is 2,473 microseconds and computing-time is

681 microseconds. Thus, the gas usage for pairing computation is defined as $2473 * 68 + k * 681 * 68 = 168164 + k * 46308$, where k is the number of pairing computation. Our designed blockchain platform can be publicly accessed at http://82.156.224.174:7070/tenon.

The experiment results show that, when the access policy size is 20, the gas usage of uploading the transformation key and the ciphertext is 1022262 and 2497431 respectively (Fig. 7(b)). Additionally, the experiment shows that the gas usage of each meta computing is about 651304. In the current official Ethereum blockchain, the transactions[3] whose gas usage is similar to the meta computing were involved by the miners soon after their issuing. This ensures the meta computing transactions will be verified and involved by the miners in blockchain the network, and thus the verifier's dilemma problem was overcome.

VIII. Conclusion

In this article, we proposed an attribute based encryption with reliable outsourced decryption scheme based on the smart contract in Ethereum and then presented a formal security proof of its confidentiality. We also analyzed the verifiability and exemptibility of the proposed scheme. Additionally, we presented a detailed explanation to show that the proposed scheme achieves the fairness property. Finally, we conducted an implementation to illustrate the practicality and efficiency of the proposed scheme. Many interesting problems remain, such as designing a smart contract platform that supports outsourcing different kinds of cryptographic computation.

References

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

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

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

[4] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," *Lecture Notes Comput. Sci.*, vol. 2008, pp. 321–334, 2011.

[5] M. Green et al., "Outsourcing the decryption of ABE ciphertexts," in *Proc. 20th USENIX Secur. Symp.*, 2011, Art. no. 34.

[6] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[7] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.

[8] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.

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

[10] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 10, pp. 2119–2130, Oct. 2015.

[11] S. Yi, A. Andrzejak, and D. Kondo, "Monetary cost-aware checkpointing and migration on amazon cloud spot instances," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 512–524, Fourth Quarter 2012.

[12] N. Kshetri, "Cloud computing in developing economies," *Computer*, vol. 43, no. 10, pp. 47–55, Oct. 2010.

[13] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[14] A. Akinyele, C. Lehmann, M. Green, M. Pagano, Z. Peterson, and A. Rubin, "Self-protecting electronic medical records using attribute-based encryption on mobile device," Technical report. Cryptology ePrint Archive, Tech. Rep. 2010/565, 2010.

[15] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 735–737.

[16] H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 6, pp. 679–692, Nov./Dec. 2017.

[17] 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 Secur.*, vol. 15, no. 1, pp. 3227–3238, Feb. 2020.

[18] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 706–719.

[19] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Annu. Cryptol. Conf.*, Springer, 2010, pp. 465–482.

[20] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proc. Theory Cryptogr. Conf.*, Springer, 2012, pp. 422–439.

[21] F. Monrose, P. Wyckoff, and A. D. Rubin, "Distributed execution with remote audit," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 1999, pp. 3–5.

[22] A. Küpçü, "Incentivized outsourced computation resistant to malicious contractors," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 6, pp. 633–649, Nov./Dec. 2017.

[23] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for C: Verifying program executions succinctly and in zero knowledge," in *Proc. Annu. Cryptol. Conf.*, Springer, 2013, pp. 90–108.

[24] A. C. Yao, "Protocols for secure computations," in *Proc. IEEE 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164.

[25] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[26] B. Dan and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. Int. Cryptol. Conf.*, 2001, pp. 213–229.

[27] Beimel and Amos, Secure Schemes for Secret Sharing and Key Distribution, Technion-Israel Institute of Technology, Faculty of Computer Science, 1996.

[28] B. Lynn et al., "PBC library," 2006. [Online]. Available: http://crypto.stanford.edu/pbc

[29] T. Wang, H. Ma, Y. Zhou, R. Zhang, and Z. Song, "Fully accountable data sharing for pay-as-you-go cloud scenes," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 2005–2016, Jul./Aug. 2021.

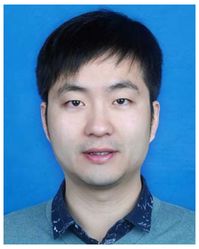
[30] S. C. Antonio and W. Zachary, "EIP-1108: Reduce alt_bn128 precompile gas costs," 2018. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1108

[3]For example, the ERC-20 transactions whose transactions hash are 0x4f423f372a6f468aa62543580e3ee4a3946c5787f04aaeae61731db15d4e0e79 and 0x624b0d4263f5cbd096aaa87b1db268424cfc119ed7110324061a7f75ddf7f9fb.

Chunpeng Ge (Member, IEEE) received the PhD degree in computer science and technology from the Nanjing University of Aeronautics and Astronautics, in 2016. He was a research fellow with the Singapore University of Technology and Design and the University of Wollongong. His current research interests include information security and privacy-preserving for cloud computing, blockchain, security and privacy of AI systems. He has published more than 60 papers in prestigious journal and conferences. He served as the editor of the journal of CSI and program committee for more than 30 international conferences.

Zhe Liu (Senior Member, IEEE) received the BS and MS degrees from Shandong University, China, in 2008 and 2011, respectively, and the PhD degree from the University of Luxembourg, Luxembourg, in 2015. He is a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include security, privacy and cryptography solutions for the Internet of Things. He has co-authored more than 80 research peer-reviewed journal and conference papers. He was a recipient of the prestigious FNR Awards-Outstanding PhD Thesis Award, in 2016, ACM CHINA SIGSAC Rising Star Award, in 2017 as well as DAMO Academy Young Fellow, in 2019. He serves as program committee member in more than 60 international conferences, including program chairs in INSCRYPT 2019, CRYPTOIC 2019 and ACM CHINA SIGSAC 2020.

Willy Susilo (Fellow, IEEE) received the PhD degree in computer science from the University of Wollongong, Australia. He is currently a distinguished professor and the head of the School of Computing and Information Technology and the director of the Institute of Cybersecurity and Cryptology, University of Wollongong. He has published more than 400 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. He was a recipient of the Australian Research Council (ARC) future fellow by the ARC and the researcher of the Year Award by the University of Wollongong, in 2016. He is the editor-in-chief of the *Elsevier's Computer Standards and Interfaces* and *MDPI's Information* journals. He has served as a program committee member in dozens of international conferences. He is currently serving as an associate editor in several international journals, including *ACM Computing Survey* and the *International Journal of Information Security* (Springer). His work has been cited more than 24,000 times in Google Scholar.



Liming Fang (Member, IEEE) is a professor with the College of Computer Science, Nanjing University of Aeronautics and Astronautics. He has published more than 100 papers in prestigious journals and conferences. His current research interests include cryptography and information security. His recent work has focused on the topics of public key encryption with keyword search, proxy re-encryption, identity-based encryption.

Hao Wang (Member, IEEE) is currently working toward the PhD degree in computer science from the Nanjing University of Aeronautics and Astronautics. His current research interests include data security, cryptography and information security. His recent work has focused on the topics of public key encryption with proxy re-encryption and identity-based encryption.