Journal of Systems Architecture 102 (2020) 101653

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud

Nabeil Eltayieb[a], Rashad Elhabob[a], Alzubair Hassan[b], Fagen Li[a,*]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[b] School of Computer Science, Guangzhou University, Guangzhou 510006, China

ARTICLE INFO

Keywords:
Cloud computing
Blockchain
Attitude-based signcryption
Data sharing

ABSTRACT

Traditional cloud data sharing schemes have relied on the architecture of the network and large storage providers. However, these storage providers work as trusted third parties to transfer and store data. This kind of cloud storage model has some weak points, such as data availability, centralized data storage, high operational cost, and security concern. In this paper, we combine the concept of blockchain with attribute-based signcryption to provide a secure data sharing in the cloud environment. The proposed scheme satisfies the security requirements of the cloud computing such as confidentiality and unforgeability. Further, the smart contract solves the problem of cloud storage such as returning wrong results as in the traditional cloud server. Finally, the performance comparisons and simulation results show that our proposed scheme is more efficient than others, and it is practical.

1. Introduction

Over the years, the importance of cloud computing has become notable; many individual users and companies resort to the cloud for various services. By moving their data to the cloud storage, the data owners get low cost, scalability, and the availability of the cloud. Besides, the data owners can be liberated from updating the software, periodic maintenance, and maintaining the storage infrastructure. Despite the tremendous benefits, the security and privacy are still the obstacles in the cloud computing usages [1]. For instance, the users don't know how their data are organized in the cloud, store data in centralized format, and limited control that granted from the cloud. Moreover, most of the existing schemes are suffering from data availability and the centralized data storage. Therefore, secure data sharing scheme based on a trusted construction and cryptographic system becomes necessary in the cloud environment. Recently, the emergence of the blockchain technology in the cloud computing has fascinated the attention of a big number of the researchers [2], which can solve the problem of centralized storage and mutual trust. Also, when the data enters the blockchain, all the information about the transactions have to be recorded. Besides, no user will able to change this data. This feature makes the use of the blockchain technology simple and more efficient than other security methods.

In the cloud technology, the data owners outsource their sensitive informations to the cloud to share it with their customers. This feature helps the data owners and authorized users to reach their data from anywhere through the Internet when they require it. The essential issue is what is the warranty that let your sensitive data accessible only by authorized users? (who already are selected by the data owner). Attackers may take illegal data access and modify the data before authorized users. Consequently, the data owners need to prove the genuineness of the outsourced data by using cryptographic methods. So as to fulfill secure access control, data confidentiality, and ciphertext unforgeability, we combine the features of both the blockchain technology and the Attribute-Based Signcryption (ABSC) [3]. Indeed, the signcryption gives a more efficient way by using a signature and an encryption scheme separately. Consequently, the ABS has been widely utilized in the cloud computing [4,5]. Further, the message is signed without exposing the identity of the users.

1.1. Contributions

Based on the technology of blockchain and attribute-based signcryption, we construct a secure data sharing scheme for cloud environment to deal with the problems mentioned above. The main contributions of our work are summarized as below:

1. To ensure efficient access control over the data in the cloud server, we construct a new scheme called Blockchain-based Attribute-Based Signcryption (BABSC).
2. By combining the blockchain with the advantages of signature and encryption, the proposed protocol can achieve confidentiality and unforgeability.

* Corresponding author.
E-mail addresses: nabeil9@yahoo.com (N. Eltayieb), rashaduestc@gmail.com (R. Elhabob), alzubairuofk@gmail.com (A. Hassan), fagenli@uestc.edu.cn (F. Li).

https://doi.org/10.1016/j.sysarc.2019.101653
Received 13 April 2019; Received in revised form 27 August 2019; Accepted 7 October 2019
Available online 5 November 2019

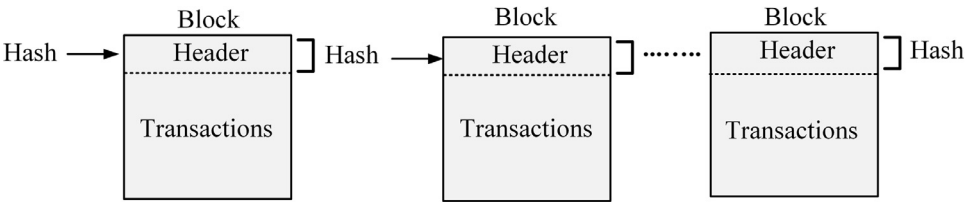

Fig. 1. The structure of blockchain.

3. We further compare BABSC with similar ABSC schemes regarding the storage and the computation costs. Also, the experiments' result determines that BABSC protocol has better performance than others.

1.2. Organization

The remaining sections are organized as follows: In Sect. 2, the details about the attribute-based signcryption and blockchain are given. In Sect. 3, the preliminary knowledge used in BABSC scheme is presented. The framework of BABSC scheme is mentioned in Sect. 4. The security proof and the performance are shown in Sect. 5 and Sect. 6, respectively. The conclusion is drawn in Sect. 7.

2. Related work

2.1. Attribute-based signcryption

Zheng [6] introduced the concept of digital signcryption, which combined the advantage of the encryption and signature in a single phase. The ABSC is a logical mixture of ABE and ABS, which is one of the effective and promising strategies. It provides many security properties such as data confidentiality, ciphertext unforgeability, data authentication, and secure access control. Further, it has less computation cost compared with the traditional encrypt-after-sign method [7]. Recently, several data sharing schemes in cloud computing based on ABSC have been introduced. Liu et al. [5] have suggested a new scheme to secure health records data, and they claimed that their scheme achieve data confidentiality. Unfortunately, Rao [8] showed [5] does not achieve data confidentiality. In addition, the scheme does not realize the public ciphertext verification. Sreenivasa et al. [9,10] introduced two ABSC schemes. The first ABSC in [9] is built on the concept of a key-policy attribute-based signcryption (KP-ABSC) by adopting constant size ciphertext and boolean function. However, both schemes have been proven in random oracle model under decisional bilinear diffie-hellman. Wang et al. [11] suggested another ABSC scheme, which merges ABE and ABS based on access trees. The computation cost of Wang et al.'s scheme is low, and it is proved under the generic group model and the random oracle model. To ensure data integrity and traceability in medical data, Wang et al. [12] introduced a new scheme by combining blockchain techniques with attribute-based/identity-based encryption beside the concept of the signature. However, their scheme suffers from the large computational overhead on user side.

2.2. Blockchain

The concept of blockchain technology is dawned from bitcoin [2]. It has attracted both industries and academia. Blockchain originally developed to support crypto currency services such as digital assets, remittance and online payment [13]. It mainly depends on blocks containing information which cannot be changed, and these blocks are cryptographically linked. For this, no attacker can modify it. The structure of blockchain is strong because each block is connected with the previous block and it is identifiable by a hash, which is created using the SHA-256 algorithm (see Fig. 1). The benefit of using the hash function is to map data of arbitrary size to data with a fixed size.

Nowadays, the blockchain technology is extensively spread. Its use is not limited to digital currency, but it is used in different fields, such as cloud computing [14], personal health records [12,15], electronic voting [16,17], and the Internet of Things [18,19]. However, blockchain could be an efficient solution to solve some of the security issue related to the data in the cloud, by distributing peer-to-peer computing. In this work, we introduce a new protocol using blockchain and ABSC to secure data sharing in the cloud environment.

We can summarize the capabilities of using blockchain to secure data sharing in the cloud storage as follow:

1. Blockchain presents real-time auditing for all data sent to the cloud server. In addition, user anonymity can be ensured and the security of transactions can be increased.
2. The use of blockchain decreases the need for trust. Even the cloud computing is not trusted for keeping the data.
3. The decentralized system in the blockchain ensures data integrity by making a copy of data records with each node [20]. However, it leads to resisting against any distributed denial-of-service (DDoS) attack, and no failure problem since no single node holds all the data record.

3. Preliminaries

3.1. Bilinear map and hard assumptions

- Bilinear map: Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ be multiplicative cyclic bilinear groups of same prime order p. Suppose that g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. A bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ is a map with the following properties:
  1. Bilinear: $\forall g_1 \in \mathbb{G}_1, \forall g_2 \in \mathbb{G}_2$ and $\forall a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ holds.
  2. Non-degeneracy: g_1 and g_2 satisfy $e(g_1, g_2) \neq 1$, where 1 is an identity element in $\mathbb{G}_3$.
  3. Computability: To compute $e(g_1, g_2)$, $\forall\, g_2 \in \mathbb{G}_2$ there will be an efficient algorithm.
- Computational Bilinear Diffie-Hellman problem (CBDH): Given $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ and $a, b, c \in \mathbb{Z}_p$. The CBDH(A, B, C) $:= Z$, where $A = g_1^a$, $B = g_2^b$, $C = g_1^c$ and $Z = e(g_1, g_2)^{abc} = e(g_1, g_2)^z$.

3.2. The secret sharing scheme

Since Shamir [21] introduced the concept of secret sharing, it has been widely used in the Attribute-Based Encryption (ABE) schemes. It is one of the important security mechanisms used by BABSC scheme. In the context, the owner of data desires to share a secret value $s \in \mathbb{Z}_p$ with n users $u_1, u_2, \ldots, u_n$, where $p > n$. If a user wants to discover the secret, he/she cooperates with at least $t-1$ other users. Let $t \leq n$ is a pre-determined parameter. Each user u_i has a secret key k_i (Just known by u_i and the data owner). Next, the data owner should follow two steps:

- In short, the data owner creates a random polynomial $f(z)$ of degree at most $t-1$ shown below:

$$f(z) = s + \sum_{j=1}^{t=1} a_j z^j \tag{1}$$

Each a_j with a uniform distribution from $\mathbb{Z}_p$ is randomly chosen. Two notes about the above equation:

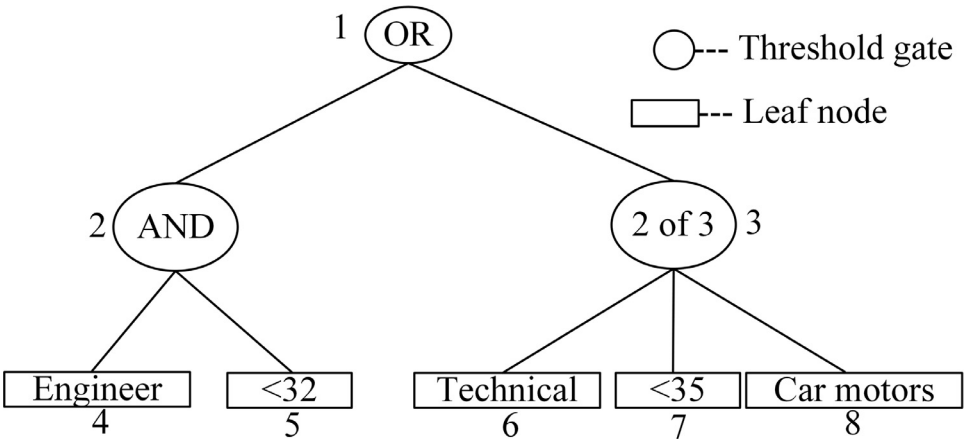

Fig. 2. An example of the access tree structure.

1. All additive and multiplicative operations used in this equation and the rest of this paper are modular arithmetic. (defined over $\mathbb{Z}_p$)
2. s is constant component of $f(z)$.

- The data owner sends to each of his users u_i a shared secret $s_i = f(k_i)$.

Let $u_1, \ldots, u_t$ are t users want to cooperate. They can reconstruct the secret $s = f(0)$ using $s_1 = f(k_1), \ldots, s_t = f(k_t)$ by calculating:

$$s = f(0) = \sum_{j=1}^{t} (s_j \prod_{i \in [1,t], i \neq j} \frac{0 - k_i}{k_i - k_i}) \quad (2)$$

The correctness of equation (2) depends on the value of $f(z)$. The Lagrange coefficient is displayed in equation (1) as a cumulative product.

3.3. The access tree

The BABSC scheme relies on the access structure tree proposed in [22]. The goal of using this access tree is to enforce the user's access policy in a different operation such as: encryption, decryption. The next example declares the main idea behind this access tree. Consider the job conditions for a company are: ("Engineer" and "<32") or (2 of "Technical", "<35", "Car motors"). To represent these conditions in the access tree, we define:

T: tree representing the access structure; parent(x): parent of a node x; att(x): if x is a leaf node then return the attribute associated with x; num(x): the number of children of a node x; $k(x)$ threshold value, then $(0 \leq k(x) \leq \text{num}(x))$. If $k(x) = 1$ the threshold is an OR gate, $k(x) = \text{num}(x)$, it is AND gate; index(x): return node's index. Fig. 2 explains the access control tree for the example.

3.4. The definition of BABSC

Our BABSC scheme comprises the next four algorithms.

1. **Setup(λ, U):** It's run by a trust authority (TA), which takes a security parameter k and generates a master secret key msk and public system parameters pk. The system parameters pk is shared by user while msk is kept secret.
2. **Keygen(msk, S):** Upon input msk and an attribute set S, the algorithm produces the private key SK and the verification key K_v according to user attributes set S. Next, to share the encrypted transaction information in the blockchain, our scheme uses the smart contract (see Fig. 5).
3. **Signcrypt(M, T, SK):** It's run by the data owner, it takes the plaintext M, the access tree T and the private key SK as inputs. At the end, it outputs the signcrypt ciphertext CT_s.
4. **De-signcrypt(SK, CT_s, S):** The De-signcrypt algorithm is run by the users, which takes the receiver's private key SK, the signcrypt ciphertext CT_s, and the attributes set S as inputs. At the end of this stage, it produces the M.

4. The overview of BABSC

This section describes briefly the network model, the security requirements, smart contract, security model, and the construction of BABSC.

4.1. Network model

The proposed blcokchain-based attribute-based signcryption for secure cloud data sharing scheme consists of five entities: a data owner, a data user, a cloud server, a trust authority, and a blockchain. The structure of the proposed scheme is presented in Fig. 3.

1. Cloud Server (CS): It's in charge of storing data owners' outsourced ciphertext data. The cloud is usually untrusted by other entities. The cloud does not engage in the data sharing control.
2. Data Owner (DO): The DO specifies the access policy predicates for his data, he signcrypts the data according to the access structure tree. Then, he sends it to the cloud server.
3. Data User (DU): To access the outsourced data, the user should have enough attributes in the access policy associated with that ciphertext.
4. Trust Authority (TA): It is responsible for generating and distributing keys that will be used by the DO and DU.
5. Blockchain: In BABSC scheme, we used the blockchain for collecting the transaction information. This information is encrypted before uploading to the blockchain. To enforce an agreement on the nodes (users), we use the smart contract which is a part of the blockchain. For auditing purposes the blockchain records all the access requests and access activities.

In Fig. 3, each number describes a process, which is shown as follows:

1. TA generates the keys by running the Setup algorithm. These keys will be used by the DO and DU.
2. DO creates the smart contract on the blockchain. The data on the smart contract must be encrypted. (See Fig. 5).
3. In order to use the data in the cloud. DU sends a registration request to DO.
4. DO calls the signcrypt algorithm and signcrypts the data according to the access structure tree and sends it to the cloud.
5. DO records the Files Location Information returned by the cloud server.
6. DO hashes the Files Location Information (FLI) and embeds it into Blockchain.
7. DO generates hashed FLI index and stores it in the smart contract.

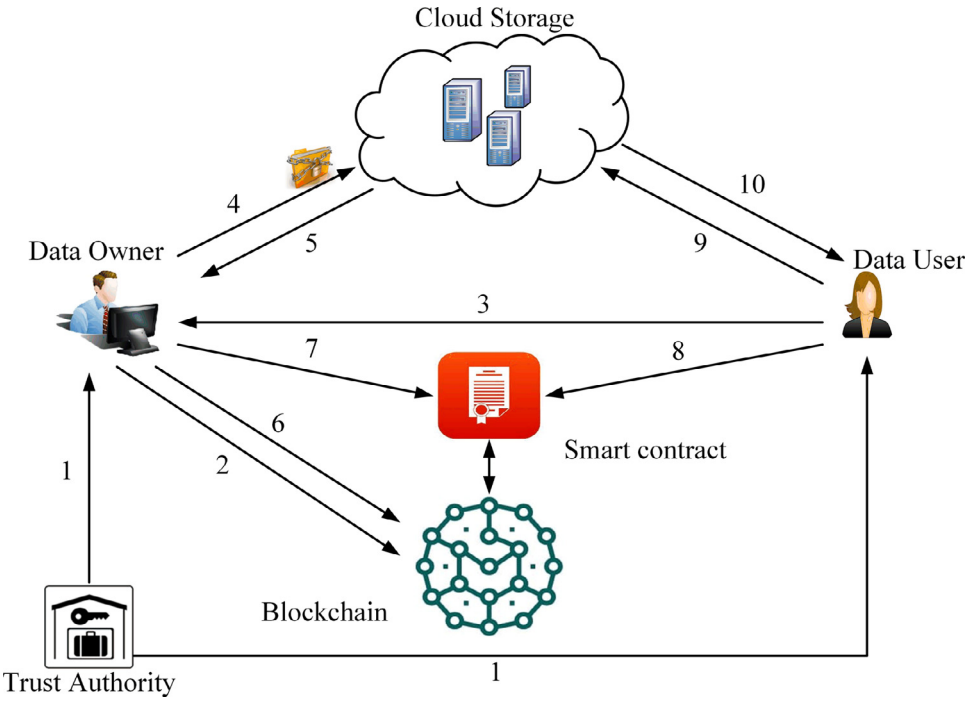

Fig. 3. Network Model.

8. In order to use the data in the cloud. DU accesses the FLI index in the smart contract.
9. To retrieve the data, DU sends a request and downloads the data from the cloud.
10. DU obtains the data by running the De-signcrypt algorithm and verifies whether the data is sent by the data owner or not.

4.2. Security requirements

This subsection outlines the security requirements of BABSC:

1. Confidentiality: It is the biggest challenge in cloud computing. To apply data confidentiality in cloud computing schemes, security mechanisms should be taken to prevent sensitive information from reaching by unauthorized users, and at the same time that the authorized users get it.
2. Access control: The access tree structure offers secure control of the data. Hence, the authorized user should pass the access constraints to get the data from the cloud storage.
3. Unforgeability: An active adversary who wish to signcrypt the data on behalf of the data owner cannot infer the signing key and create a valid ciphertext. There is an effective signing predicates to protect the data against such masquerading attacks.

4.3. Smart contract

A smart contract is a computer protocol, which works inside the blockchain to ensure the scalability of access control [23]. In the proposed scheme, the smart contract is used for secure data sharing abilities between various data owners and data users. It is used to enforce an agreement on the nodes, hense, all participants can exchange data securely. BABSC scheme uses the smart contract proposed by Watanabe et al. [24]. For a reason that the blockchain is public, and other users can view it. The data in the contract and in the transaction (see Fig. 4) needs to be signcrypted to store it in the blockchain.

The question is how/who generate the smart contract? The example in Fig. 5 specifies how the data owner creates the contract and then user 1 followed by user 2 consent to it. The working mechanism of this example is explained as follows.

- The data owner creates the contract and encrypts it using user1ı́s encryption key. Then, he makes transaction data and broadcasts this transaction data into the network.
- Considering the blockchain is synchronized with the network, user 1 receives data owner's transaction data (addressed to user 1) from the blockchain and then obtains the encrypted contract by using his decryption key.
- User 1 reviews the contract, and if he consents to it, he creates a transaction referring to the data owner's transaction. Then, user 1 encrypts the contract using user2ı́s encryption key and broadcasts it.
- In a similar procedure, user 2 also inspects the contract using his key. Then, he transmits the contract to data owner through a transaction.
- At the end,the data owner gets user2ı́s transaction and approves whether the encrypted contract he has accepted is correct or not. Since the records are encrypted, only persons having a decryption key such as data owner, user 1, and user 2 can decrypt the contract.

4.4. Security model of BABSC

A BABSC scheme is required to achieve confidentiality and unforgeability, which are typical security requirements. We hold the next interactive game played between an adversary $\mathcal{A}$ and a challenger C.

Confidentiality. According to scheme in [22], the confidentiality can be proved by the next interactive game.

1. **Init:** The adversary $\mathcal{A}$ outputs an attribute set S that will be used to create the challenge ciphertext during the **Challenge** phase.
2. **Setup:** The C runs **Setup** algorithm to generate the public parameters *pk* (which sends to $\mathcal{A}$) and the *msk* (keep secret).
3. **Query phase 1:** $\mathcal{A}$ requests adaptively a polynomially bounded number of queries as follows:
   - Private key query: In this query, the $\mathcal{A}$ asks for the private key *SK*. For each attribute set S^*, C calls **KeyGen** algorithm and replays with *SK* according to that attribute set.
   - Signcrypt query: The $\mathcal{A}$ asks to signcrypt a message *M*. For each message, C selects attribute set *S* such that $S \in T^*$. Then, C computes the private key *SK* using **KeyGen** algorithm. Next, C

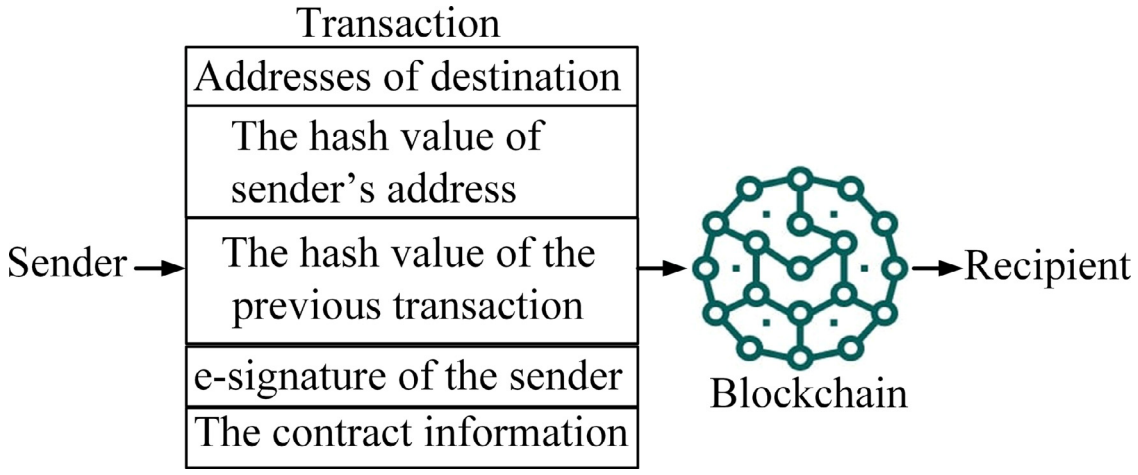

Fig. 4. The transaction information.

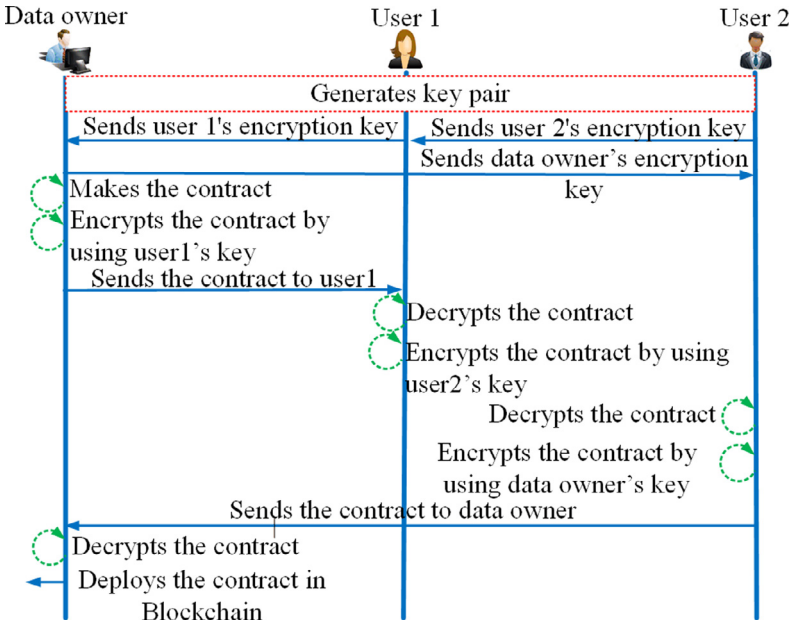

Fig. 5. The processing flow of generating the smart contract.

executes the **signcrypt** algorithm and obtains the CT_s, which forwards to $\mathcal{A}$.

- De-signcrypt query: The signcrypt ciphertext CT_s and the attribute set S are submitted by $\mathcal{A}$. First, C obtains the private key SK by calling **KeyGen** algorithm. Then, C runs the**De-signcrypt** algorithm and sends the output to $\mathcal{A}$.

4. **Challenge:** The $\mathcal{A}$ chooses two messages M_0, M_1 to be challenged. Then, $\mathcal{A}$ chooses a random bit $\int \in \{0,1\}$. To signcrypt the message M with attribute set S^*. C gets the private key SK by running **KeyGen** algorithm. Then, C runs the **signcrypt** algorithm. The CT_s is sent to $\mathcal{A}$ as challenge ciphertext.
5. **Query phase 2:** Simillar to **Query phase 1**. $\mathcal{A}$ can ask a designcryption query on any ciphertext except the challenged ciphertext.
6. **Guess:** A guessing bit $\sigma^{'}$ of σ outputted by $\mathcal{A}$. The game is won by $\mathcal{A}$ if $\sigma^{'} = \sigma$. The advantage of $\mathcal{A}$ can be showed as $Adv(\mathcal{A}) = |\Pr[\sigma^{'} = \sigma] - 1/2|$.

Definition 4.1. A BABSC scheme is judged to be indistinguishable against chosen ciphertext attack (IND-CCA), if no $\epsilon = Adv(\mathcal{A})$ in the above challenge game.

Unforgeability. In this game, the adversary has to forge the signcryption of the message (including all predicates). The formal definition of unforgeability game is built according to the scheme in [25]. The game includes the next steps:

1. **Init:** $\mathcal{A}$ submits an attribute set S to C, that will be used to forge a ciphertext.
2. **Setup:** The C executes **Setup** algorithm to produce pk (which sends to $\mathcal{A}$) and the msk is kept secret.
3. **Query:** The $\mathcal{A}$ requests in each phase as follows:
   - Keygen query: In this step the $\mathcal{A}$ asks for the private key SK and the verification key K_v. For each attribute set S^*, C calls **KeyGen** algorithm and replies with SK and K_v according to that attribute set.
   - Signcrypt query: The $\mathcal{A}$ asks to signcrypt a message M^*. For each message, the C selects attribute set S such that $S \in T^*$. Then, C computes the private key SK using **KeyGen** algorithm. Next, C executes the **signcrypt** algorithm and obtains the CT_s^*, which forwards to $\mathcal{A}$.
   - De-signcrypt query: The signcrypt ciphertext CT_s and the attribute set S are submitted by $\mathcal{A}$. First, obtains the private key

SK by calling **KeyGen** algorithm. Then, C runs the**De-signcrypt** algorithm and sends the output to $\mathcal{A}$.

4. **Forgery:** The adversary $\mathcal{A}$ outputs a forgery the ciphertext CT_s for some message M^* with the attribute set S^*. At the end of this stage, $\mathcal{A}$ wins if the outputs of De-signcrypt(CT_s, *SK*, S^*)= $M^* \neq \perp$, where $T^* = 1$ and the tuple (M^*, T^*) has not been displayed by the signcrypt algorithm before. The advantage of $\mathcal{A}$ in this game is determined as $Adv_{\mathcal{A}}^{\text{Unforgeability}} = \Pr[\mathcal{A} \text{ wins}|$

Definition 4.2. The BABSC is supposed to be secure against existential unforgeability, if no $\epsilon = Adv(\mathcal{A})$ in the above challenge game.

4.5. Concrete construction

Here, we give the precise algorithms of BABSC scheme. We built BABSC scheme based on Bethencourt et al.'s scheme [22] and the signature Scheme in [26]. The construction of the BABSC is described as follows.

1. **Setup(λ, U):** It is run by the TA. It takes as inputs the attributes universe U and the security parameter λ. It chooses cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with generators g_1 and g_2, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$. It chooses two random exponents $\alpha, \beta \in \mathbb{Z}_p$. Also, it selects a random oracle hash function $H : \{0,1\}^* \longrightarrow \mathbb{Z}_p$. At last, TA generates the *pk* (publicly known) and the *msk* (kept secret) as follows:
   $pk = (p, \mathbb{G}_1, \mathbb{G}_2, H, g_1, g_2, h = g_1^{\beta}, t = e(g_1, g_2)^{\alpha})$ and $msk = (\beta, g_2^{\alpha})$
2. **Keygen(*msk, S*):** It selects a random number $r_1 \in \mathbb{Z}_p$. Then, computes $D_1 = g_2^{\frac{(\alpha+r_1)}{\beta}}$ and the verification key: $K_v = g_2^{r_1}$. To produce the private key for an attribute $j \in S$, a random number $r_j \in \mathbb{Z}_p$ chosen. The algorithm computes the private key components $D_j = g_2^{r_1} . g_2^{(H_1(j).r_j)}$ and $D_j^{'} = g_2^{r_j}$. The private key is: $SK = (D_1, \forall j \in S : D_j, D_j^{'})$, which uses for a signing and designcryption process. Finally, the TA sends *SK* for the data owner, and announces K_v for the users to verify them.
3. **Signcrypt(*M, T, SK*):** It takes the message M, the access tree T at node R, and the private key *SK* as inputs. It runs as follows:
   - First, it selects a polynomial q_x and $\forall x \in T$ lets degree $d_x = k_x - 1$.
   - It selects $s, d_R, d_x \in \mathbb{Z}_p$ randomly and sets $q_R(0) = s$;
   - $\forall x \in T$, sets $q_x(0) = q_{parent(x)}(index(x))$.
   - The ciphertext CT is built as follows:
     $C^* = M \oplus t^s$, $C = h^s$, $\forall y \in Y : C_y = g_1^{q_y(0)}$, where Y is the set of leaf nodes in T. $C_y^{'} = g_1^{(H_1(att(y)).q_y(0))}$.
     $$CT = (T, C^*, C, \forall y \in Y : C_y, C_y^{'}) \tag{3}$$
   - To sign the ciphertext CT, the algorithm selects a random $\zeta \in \mathbb{Z}_p$. Then sets $\delta = e(C, g_2)^{\zeta}$, $\pi = H_1(\delta|M)$, and $\psi = g_2^{\zeta} . D_1^{\pi}$.
   - Finally, the algorithm outputs the signcryption ciphertext:
     $CT_s = (T, C^*, C, \forall y \in Y : C_y, C_y^{'}, W = g_1^{s}, \pi, \psi)$
4. **De-signcrypt(CT_s, *SK, S*):** It takes the signcryption CT_s, *SK*, and attributes set S as inputs. It computes $B =$ **DecryptNode**(CT_s, SK, R), if $B \neq \perp$ calculates $B^{'} = e(C, D_1)/B$. Also, the algorithm computes:
   $$\delta^{'} = \frac{e(C, \psi)}{(e(W, K_v).B^{'})^{\pi}} \tag{4}$$
   Then, if $H_1(\delta^{'}|M^{'}) = \pi$, $M = M^{'}$. Otherwise, the algorithm De-signcrypt outputs $\perp$.
   Function DecryptNode (CT_s, *SK, x*): If x is a leaf node of T then Let i= $att(x)$, if $i \in S$ computes F_x as follows:
   $$F_x = \frac{e(C_i, D_i)}{e(C_i^{'}, D_i^{'})} = e(g_1, g_2)^{r_1 q_x(0)} \tag{5}$$
   $\forall z \in x$ computes F_z= DecryptNode (CT, *SK*, z). To compute F_x sets $F_z \neq \perp$ for $\forall z \in S_x$, where S_x is an arbitrary k_x−sized set of child nodes of x. Let $i_z = \text{index}(z)$, $S_z^{'} = \text{index}(z)||z \in S_x$, and also computes $\Delta_{i_z, S_z^{'}}(y) = \prod_{j \in S_z^{'}, j \neq i_z} \frac{y-j}{i_z - j}$.

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i_z, S_z^{'}}(0)} = \prod_{z \in S_x} (e(g_1, g_2)^{r_1 . q_z(0)})^{\Delta_{i_z, S_z^{'}}(0)}$$
$$= \prod_{z \in S_x} e(g_1, g_2)^{r_1 . q_x(i_z) . \Delta_{i_z, S_z^{'}}(0)} = e(g_1, g_2)^{r_1 . q_x(0)}$$

Correctness: We display the correctness of BABSC scheme, which is done in two steps:

- The decryption procedure can be by following equations:

$$M^{'} = C^* \oplus B^{'} = C^* \oplus \left(\frac{e(C, D_1)}{B} \right)$$
$$= C^* \oplus \left(\frac{e(e(h^s, g_2^{(\alpha+r_1)/\beta}))}{e(g_1, g_2)^{r_1 s}} \right)$$
$$= M \oplus e(g_1, g_2)^{\alpha s} \oplus \left(\frac{e(e(g_1^{\beta s}, g_2^{(\alpha+r_1)/\beta}))}{e(g_1, g_2)^{r_1 s}} \right)$$
$$= M \oplus e(g_1, g_2)^{\alpha s} \oplus \left(\frac{e(e(g_1, g_2)^{\beta s . (\alpha+r_1)/\beta})}{e(g_1, g_2)^{r_1 s}} \right)$$
$$= M \oplus e(g_1, g_2)^{\alpha s} \oplus \left(\frac{e(e(g_1, g_2)^{(\alpha s + r_1 s)})}{e(g_1, g_2)^{r_1 s}} \right)$$
$$= M \oplus e(g_1, g_2)^{\alpha s} \oplus e(g_1, g_2)^{\alpha s} = M$$

- When the authorized user receives the message $M^{'}$, he/she verifies whether M is sent by data owner or not. Then, the user calculates $\delta^{'}$:

$$\delta^{'} = \frac{e(C, \psi)}{(e(W, K_v).B^{'})^{\pi}} = \frac{e(g_1^{\beta s}, g_2^{\zeta}) \times g_2^{\left(\frac{\alpha+r_1}{\beta}\right)\pi}}{(e(g_1^{s}, g_2^{r_1}).e(g_1, g_2)^{\alpha s})^{\pi}}$$
$$= e(g_1, g_2)^{\beta s \left(\zeta + \frac{(\alpha+r_1)}{\beta}\pi\right) - s r_1 \pi - \alpha s \pi}$$
$$= e(g_1, g_2)^{\beta s \zeta + (\alpha + r_1)\pi - s r_1 \pi - \alpha s \pi}$$
$$= e(g_1, g_2)^{\beta s \zeta} = e(C, g_2)^{\zeta} = \delta$$

If $H_1(\delta^{'}|M^{'}) = \pi$, $M^{'}$ is valid and not modified, otherwise $M^{'}$ is invalid.

5. Security proof and discussion

In this part, we provide the security proof of the proposed scheme via two theorems. The security proof builds similar to the scheme in [10]. Moreover, we show the features of blockchain that support data sharing.

5.1. Security proof

Theorem 5.1. *The BABSC is secure under the IND-CCA model if CBDH assumption exists.*

Proof. The challenger C is given (A, B, C) as the CBDH assumption instance as presented in Fig. 6. The $\mathcal{A}$ tries to guess $e(g_1, g_2)^{abc}$, where $a, b, c \in \mathbb{Z}_p$. CT_s represents the challenge ciphertext, it has a component C^* which is randomly both $M_0 e(g_1, g_2)^{\alpha s}$ or $M_1 e(g_1, g_2)^{\alpha s}$. We set $\theta = \alpha s$, where θ is random from $\mathbb{Z}_p$, CT_s is either $e(g_1, g_2)^{\alpha s}$ or $e(g_1, g_2)^{\theta}$. $\mathcal{A}$ must distinguish between $M_0 e(g_1, g_2)^{\alpha s}$ and $M_1 e(g_1, g_2)^{\theta}$ extra in which it needs to differentiate between $e(g_1, g_2)^{\alpha s}$ and $e(g_1, g_2)^{\theta}$. In this game the challenger C is associated with algorithm $\mathcal{B}$. The $\mathcal{B}$ calls $\mathcal{A}$ to run the following steps. □

1. **Init:** The adversary $\mathcal{A}$ submits the target attribute set S^* to $\mathcal{B}$.

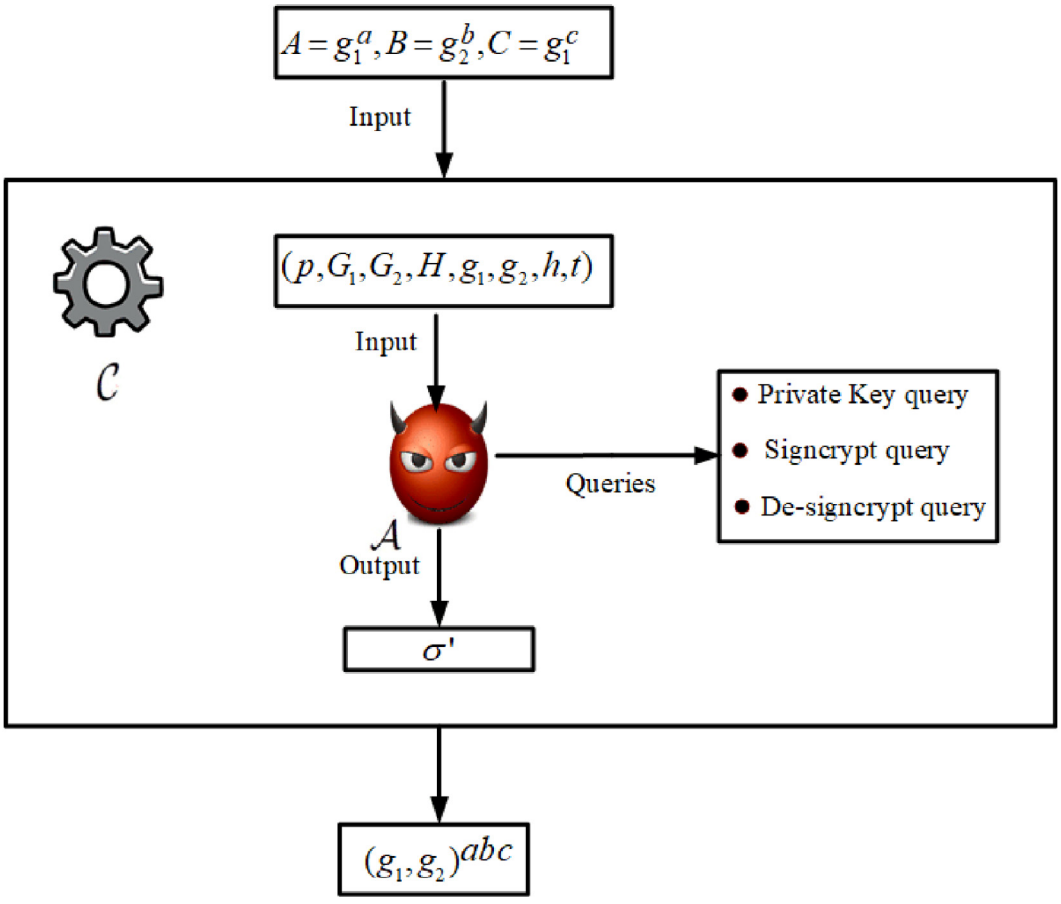

Fig. 6. The structure of IND-CCA security proof.

2. **Setup:** Two random $\alpha, \beta \in \mathbb{Z}_p$ were chosen by $\mathcal{B}$. If $\beta = 0$ then **Setup** aborted. Otherwise, $\mathcal{B}$ runs the **Setup** algorithm to get the public parameters. Next, $\mathcal{B}$ sends $h = g_1^{\beta}$ and $t = e(g_1, g_2)^{\alpha}$ to $\mathcal{A}$. When $\mathcal{A}$ asks to evaluate H, $\mathcal{B}$ picks a random $t_j \in \mathbb{Z}_p$ and provides g^{t_j} as the answer to $H(j)$.
3. **Query phase 1:** In this phase, the $\mathcal{A}$ asks for numbers of inquiries as follows:
   - Private key query: If the $\mathcal{A}$ makes query asking for private key SK for a set of attributes S^*. $\mathcal{B}$ selects random numbers $r_1 \in \mathbb{Z}_p$ and sets $D_1 = g_2^{\frac{(\alpha+r_1)}{\beta}}$. Therefore, for an attribute $j \in S$, $\mathcal{B}$ selects a random number $r_j \in \mathbb{Z}_p$. It computes $D_j = g_2^{r_1} . g_2^{(H_1(j).r_j)}$ and $D_j' = g_2^{r_j}$. The private key SK is: $SK = (D_1, \forall j \in S : D_j, D_j')$. Then, $\mathcal{B}$ returns SK to $\mathcal{A}$. When $\mathcal{A}$ sends i'th key generation query for the attributes set S_i. $\mathcal{B}$ selects a new random value $r^{(i)} \in \mathbb{Z}_p$. Then, $\mathcal{B}$ computes $D_1 = g_2^{\alpha + r^{(i)}/\beta}$, $\forall j \in S_i$ we have $D_j = g_2^{r^{(i)} + t_j r_j^{(i)}}$ and $D_j' = g_2^{r_j^{(i)}}$. These values are sent to $\mathcal{A}$.
   - Signcrypt query: The $\mathcal{A}$ asks for signcrypt a message M^*. For each message, $\mathcal{B}$ executes the **Keygen** algorithm to get the private key. Then, it runs the **signcrypt** (M, T, SK) algorithm and obtains the CT_s^*, which forwards to $\mathcal{A}$.
   - De-signcrypt query: In this step, the $\mathcal{A}$ sends de-signcryption requests for ciphertext $CT_s = (T, C^*, C, \forall y \in Y : C_y, C_y', W = g_1^s, \pi, \psi)$. First, $\mathcal{B}$ verifies whether $C = C^*$ If yes, $\mathcal{B}$ terminates (Since $C = g_1^{\theta}$ is random in $\mathcal{A}'$s view, the probability of this type of ciphertext presented by the $\mathcal{A}$ is at most $1/p$). Otherwise, $\mathcal{B}$ progresses as follows.

If $S^* = 0$ (does not satisfies the de-signcryption access tree T). $\mathcal{B}$ executes the **Keygen** to get the private key. Then runs the **De-signcrypt** (CT_s, SK, S) and returns the outputs to $\mathcal{A}$.
If $S^* = 1$ (satisfies the de-signcryption access tree T). $\mathcal{B}$ first verifies the validity of the ciphertext using Eq. (4). If it is incorrect, outputs $\perp$. Otherwise, it determines $e(g_1, g_2)^{r^{(i)}.q_x(0)}$ using the following calculation

$$\frac{e(C_j, D_j)}{e(C_j', D_j')} \tag{6}$$

If the tree satisfied by S, we set B= DecryptNode (CT, SK, R)= $e(g_1, g_2)^{r^{(i)}.q_x(0)} = e(g_1, g_2)^{rs}$. Then the algorithm decrypts by calculating:

$$C^*/(e(C, D_1)/B) = M \oplus e(g_1, g_2)^{\alpha s}/(e(h^s, g_2^{\frac{(\alpha+r_1)}{\beta}})/e(g_1, g_2)^{rs}) = M$$

Finally, the message M sends to adversary $\mathcal{A}$. Since Eq. (4). is correct, all the ciphertext components are consistent and hence $B' = e(C, D_1)/B$. Therefore,

$$B' = \left(\frac{e(C, D_1)}{B}\right) = \left(\frac{e(e(h^s, g_2^{(\alpha+r_1)/\beta}))}{e(g_1, g_2)^{r_1 s}}\right) = \left(\frac{e(e(g_1^{\beta s}, g_2^{(\alpha+r_1)/\beta}))}{e(g_1, g_2)^{r_1 s}}\right)$$

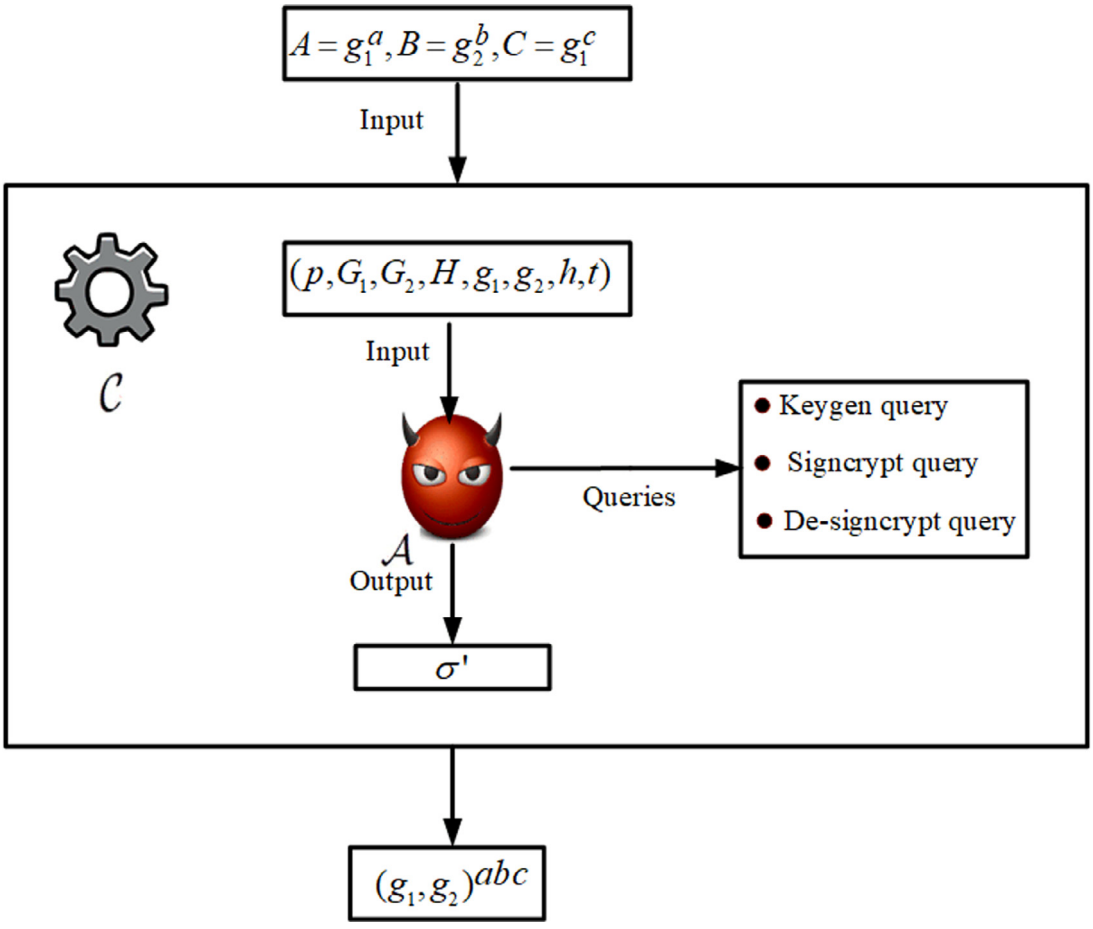

Fig. 7. The structure of unforgeability security proof.

$$= \left(\frac{e(e(g_1, g_2)^{\beta s.(\alpha + r_1)/\beta})}{e(g_1, g_2)^{r_1 s}} \right)$$
$$= \left(\frac{e(e(g_1, g_2)^{(\alpha s + r_1 s)})}{e(g_1, g_2)^{r_1 s}} \right)$$
$$= e(g_1, g_2)^{\alpha s} = e(g_1, g_2)^{\theta}$$

4. **Challenge:** The adversary $\mathcal{A}$ chooses two equal-length messages M_0^*, M_1^* with attribute set S^* to be challenged. $\mathcal{B}$ chooses a random bit $\sigma \in \{0, 1\}$ and signcrypts M_σ^* under the challenge attribute set S^*. The components of challenge ciphertext CT_s^* are simulated as follows:
   - First, $\mathcal{B}$ selects a random $s \in \mathbb{Z}_p$ and uses f_j to construct shares s or attributes j.
   - It calculates $C^* = M_\sigma^* \cdot Z$, $C_j = g_1^{ab}$, $C_j' = g_1^{t_j f_j}$.
   - The algorithm selects a random $\mu \in \mathbb{Z}_p$. Then, computes $\delta^* = e(C, g_2)^{\mu}$, $\pi^* = H_1(\delta^* | M_\sigma^*)$, and $\psi^* = g_2^{\mu} . D_j^{\pi^*}$.
   - $\mathcal{B}$ transfers the challenge ciphertext $CT_s^* = (T^*, C^*, C_j, C_j', W^* = g_1^s, \pi^*, \psi^*)$ to $\mathcal{A}$.
5. **Query phase 2:** For the second times the $\mathcal{A}$ requests for queries. The $\mathcal{B}$ answers these queries similar to **Query phase 1**. There is no designcryption query.
6. **Guess:** A guessing bit σ' of σ is outputted by $\mathcal{A}$. The game is won by $\mathcal{A}$ if $\sigma' = \sigma$. If $Z = e(g_1, g_2)^{abc}$, then CT_s^* is a valid ciphertext, in which case the advantage is ϵ. Hence,

$$Adv_{\mathcal{B}}^{\text{CBDH}} = \Pr[Z = e(g_1, g_2)^{abc}]$$
$$= \Pr[\sigma' = \sigma | Z = e(g_1, g_2)^{abc}] = \frac{1}{2} + \epsilon$$

The advantage of $\mathcal{A}$ in the CBDH game is $\frac{\epsilon}{2}$

Theorem 5.2. *The BABSC is unforgeable under the selective predicate attack based on the CBDH assumption.*

Proof. Assume that an adversary $\mathcal{A}$ has an advantage ϵ in breaking BABSC under the selective predicate. The challenger $\mathcal{B}$ is given (*A, B, C*) as the CBDH assumption instance as presented in Fig. 7. The $\mathcal{A}$ tries to guess $e(g_1, g_2)^{abc}$, where $a, b, c \in \mathbb{Z}_p$. We set $\theta = \alpha s$, where θ is random from $\mathbb{Z}_p$. An algorithm $\mathcal{B}$ was developed to solve the next game. □

1. **Init:** The adversary $\mathcal{A}$ submits the target attribute set S^* to $\mathcal{B}$.
2. **Setup:** Two random $\alpha, \beta \in \mathbb{Z}_p$ were chosen by $\mathcal{B}$. If $\beta = 0$ then **Setup** aborted. Otherwise, $\mathcal{B}$ runs the **Setup** algorithm to get the public parameters. Next, $\mathcal{B}$ sends $h = g_1^{\beta}$ and $t = e(g_1, g_2)^{\alpha}$ to $\mathcal{A}$. When $\mathcal{A}$ asks to evaluate H, $\mathcal{B}$ picks a random $t_j \in \mathbb{Z}_p$ and provides g^{t_j} as the answer to $H(j)$.
3. **Query:** The $\mathcal{A}$ requests in each phase as following:
   - Keygen query: The $\mathcal{A}$ asks for the private key SK and the verification key K_v. For private key query, it similar to the IND-CCA in **Theorem 5.1.** When $\mathcal{A}$ asks for the verification key K_v, $\mathcal{B}$ chooses random numbers $r_1 \in \mathbb{Z}_p$. It runs the **KeyGen** algorithm, then, sends $K_v = g_2^{r_1}$ to $\mathcal{A}$. Otherwise, it selects random $r^{(i)} \in \mathbb{Z}_p$ and simulates $K_v = g_2^{r^{(i)}}$. At the end, $\mathcal{B}$ sends K_v to $\mathcal{A}$.
   - Signcrypt query: The challenger $\mathcal{B}$ formulates an access tree T^* with S^* an authorized attribute set. If the challenge attribute set S^* does not satisfy T^*, the challenger $\mathcal{B}$ can obtain the private key from **Keygen** algorithm. Then, it runs **signcrypt** algorithm and returns CT_s to $\mathcal{A}$. Suppose S^* satisfies T^*. In this case, $\mathcal{B}$ performs as follows.
     - $\mathcal{B}$ selects a random $b \in \mathbb{Z}_p$. Then, it uses b to construct shares s or attributes j.

- Moreover, $\mathcal{B}$ chooses a random bit $\theta \in \mathbb{Z}_p$ and runs the **signcrypt** algorithm.
- It calculates $C^* = M^* \cdot e(g_1, g_2)^{\theta'}$, $C_j = g_1^{ab}$, $C_j' = g_1^{t_j f_j}$.
- The algorithm select a random $c \in \mathbb{Z}_p$. Then, it computes $\delta = e(C, g_2)^c$, $\pi = H_1(\delta|M)$, and $\psi = g_2^c.D_1^{\pi}$.
- At the end, $\mathcal{B}$ transfers the $CT_s = (T, C^*, C_j, C_j', W = g_1^b, \pi, \psi)$ to $\mathcal{A}$.

- De-signcrypt query: Here, $\mathcal{A}$ sends de-signcryption requests for ciphertext CT_s using the attribute set S^*. The challenger $\mathcal{B}$ calls **Keygen** algorithm to create the corresponding private key SK. Next, it runs the **De-signcrypt** (CT_s, SK, S) to designcrypt CT_s and transfers the outputs M or $\perp$ to $\mathcal{A}$.

4. **Forgery:** The adversary $\mathcal{A}$ outputs a valid forgery $CT_s^* = (T^*, C^*, C_j, C_j', W^* = g_1^s, \pi^*, \psi^*)$ for some message M^* and attribute set S^*.Then, the challenger solves the CBDH problem as follows.
Since $CT_s^* = (T^*, C^*, C_j, C_j', W^* = g_1^s, \pi^*, \psi^*)$ is a valid signcryption of M^*, it must pass the verification test stated in Eq. (4), which means that: $C_j = g_1^{ab}$, $W = g_1^b, \pi, \psi$, $K_v = g_2^{r_1}$ $\psi = g_2^c.D_1^{\pi}$, $\delta = e(C, g_2)^c$, $B' = e(g_1, g_2)^{\theta b}$.
At this step, the adversary $\mathcal{A}$ outputs a forged verification $\delta^* = e(C, \psi)/(e(W, K_v).B')^{\pi}$ using the attribute set S^*, where $S^* \in T^*$. If $S^* \neq 0$, $\mathcal{B}$ aborts. The verification δ^* is valid. Considering:

$$= \frac{e(C, \psi)}{(e(W, K_v).B')^{\pi}} = \frac{e(g_1^{ab}, g_2^c \times g_2^{(\frac{\theta+r_1}{a})\pi})}{(e(g_1^b, g_2^{r_1}).e(g_1, g_2)^{\theta b})^{\pi}}$$

$$= e(g_1, g_2)^{ab(c+\frac{(\theta+r_1)}{a}\pi)-br_1\pi-\theta b\pi}$$

$$= e(g_1, g_2)^{abc+(\theta+r_1)\pi-br_1\pi-\theta b\pi}$$

$$= e(g_1, g_2)^{abc} = e(g_1, g_2)^z$$

The $\mathcal{A}$ wins this game if the algorithm **De-signcrypt**(CT_s, SK, S)= $M^* \neq \perp$, where $T^* = 1$ and the tuple (M^*, T^*) has not been displayed by the **signcrypt** algorithm before. The advantage of $\mathcal{B}$ in solving the CBDH problem is

$$Adv_{\mathcal{B}}^{\text{CBDH}} = \Pr[\mathcal{B}(g_1^a, g_2^b, g_1^c) = e(g_1, g_2)^{abc}]$$
$$= \Pr[\mathcal{A} \text{ wins the Unforgeability game}]$$
$$= Adv_{\mathcal{A}}^{\text{Unforgeability}} > \epsilon$$

5.2. Discussion

The data sharing mechanism in ABE is associated with dynamic attributes. Utilizing a blockchain technology in data sharing gives additional restrictions and unchanging log of all significant security events. These benefits are made possible by the following features:

1. Decentralization: The information is equally distributed between the nodes. The public validation of each transaction allows anyone to verify if the system is working correctly, using the distributed ledger records. Furthermore, the decentralization protects the scheme from a single point of failure. For any change in one block; one needs to change every subsequent block before any new block could be mined.
2. Cryptography: The structure of blockchain is strong due to the cryptographic hash techniques applied. Hash values are used to hide true identities. Moreover, this hashing value is created using the SHA-256 algorithm to map data of arbitrary size to data with a fixed size.
3. Consensus: It determines which node can add a block after that node is the winner of the cryptographic race. This kind of consensus is defined as proof of work. It assures each block has passed complex mathematical operations before becoming an immutable part of the blockchain.

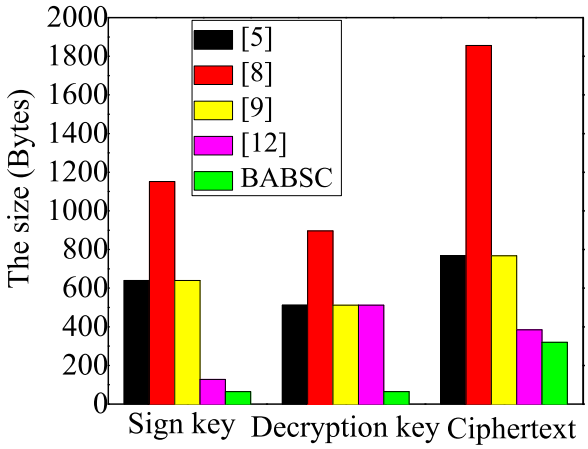

Fig. 8. The communication cost comparison.

6. Performance

This part is consecrated to examine the performance of the BABSC protocol versus existing relevant ABSC schemes proposed in [5,8,9], [12]. In Table 1, the comparison is divided into two terms: the communication cost which includes the size of (signing key, decryption key, and ciphertext); the computation cost, which shows the time used for signcryption and designcryption. The results in Table 1 validate that BABSC is an effective and powerful scheme which supports secure data sharing in the cloud. The sizes of signing key and decryption key have a direct proportion with the number of attributes. In BABSC, we use the private key SK for signing and decryption process. Both of these keys and the ciphertext have the lowest size in BABSC scheme compared with other schemes. Furthermore, the signcryption cost of the proposed scheme is less than those in [5,8,9], and it is almost the same to the scheme in [12]. While the designcryption cost of the proposed scheme is less than others. The simulation experiment was done on Intel i5-7400 computer, which has 3.00GHz CPU with ram 4GB and Windows 10- 64-bit installed. The implementation is done in VC++ 6.0 using PBC library [27]. The experiment uses type A bilinear pair which is constructed on the curve $y^2 = x^3 + x \bmod q$ over the field F_q for some prime $q = 3 \bmod 4$. Both $\mathbb{G}_1$ and $\mathbb{G}_2$ have order p and are subgroups of $E(F_q)$, where p nd q are 160-bit and 512-bit, respectively. Over all algorithms in BABSC scheme, we applied SHA-3 as a hash function. Beside, we assign the size of $\mathbb{G}_1$ and $\mathbb{Z}_p$ to 64 bytes. On the other hand, the size of $\mathbb{G}_2$ is considered 128 Bytes. More concretely, the running times for a bilinear pairing operation, exponentiation in group $\mathbb{G}_1$, exponentiation in group $\mathbb{G}_2$ are $\mathbb{P}$=13.455 ms, $\mathbb{E}_1$=6.441 ms, and $\mathbb{E}_2$=1.489 ms, respectively. These times are the average of 20 trials.

In order to precisely evaluate the performance of BABSC, we set n_s, n_d, k_s, k_e equal to 8, 6, 5, 3, respectively. As shown in Fig. 8, the BABSC scheme has the lowest size in signing key, decryption key, and ciphertext. The detailed analysis is given as follows:

- The signing key size in BABSC is about 64 byte. While, the sign key size in [5,8,9,12] are 640 bytes, 1152 bytes, 640 bytes, 128 bytes, respectively.
- In the proposed scheme the decryption key has cost 64 bytes. While, the decryption key size in [5,8,9,12] are 512 bytes, 896 bytes, 512 bytes, 512 bytes, respectively.
- For the ciphertext size, BABSC scheme has the lowest size, which is equal to 320 bytes. While, the ciphertext size in [5,8,9,12] are 768 bytes, 1856 bytes, 768 bytes, 384 bytes, respectively.

To sum up, we conclude that the size of the signing key and the decryption key in BABSC are constant. So, the signing key and the de-

Table 1
The comparison of computational complexity.

Scheme	Communication cost			Computational cost		Method
	Sign key	Decryption key	Ciphertext	Signcryption	Designcryption	
[5]	$(n_s+2)L_{\mathbb{G}_1}$	$(n_d+2)L_{\mathbb{G}_1}$	$(k_s+k_e+4)L_{\mathbb{G}_1}$	$(3+2k_s+k_e)\mathbb{E}_1$	$(5+k_s)\mathbb{P}+(k_e^2+2k_e+k_s+3)\mathbb{E}_1$	ABSC
[8]	$(2n_s+2)L_{\mathbb{G}_1}$	$(2n_d+2)L_{\mathbb{G}_1}$	$(3k_s+3k_e+5)L_{\mathbb{G}_1}$	$(3k_e+3k_s+4)\mathbb{E}_1$	$(5+k_s)\mathbb{P}+2k_e\cdot\mathbb{E}_1$	ABSC
[9]	$(n_s+2)L_{\mathbb{G}_1}$	$(n_d+2)L_{\mathbb{G}_1}$	$(k_s+k_e+4)L_{\mathbb{G}_1}$	$\mathbb{P}+(3+2k_s+k_e)\mathbb{E}_1$	$(3k_e+2k_s+4)\mathbb{P}$	ABSC
[12]	$2L_{\mathbb{G}_1}$	$(\lvert n_d\rvert+2)L_{\mathbb{G}_1}$	$(k_e+3)L_{\mathbb{G}_1}$	$(k_s+3)\mathbb{E}_1$	$3\mathbb{P}$	ABE+IBE+IBS+$\mathbb{B}$
Ours	$L_{\mathbb{G}_1}$	$L_{\mathbb{G}_1}$	$(k_e+2)L_{\mathbb{G}_1}$	$(2k_e+2)\mathbb{E}_1+\mathbb{E}_2$	$\mathbb{P}$	ABSC+$\mathbb{B}$

Legends: $L_{\mathbb{G}_1}$ and $L_{\mathbb{G}_2}$: the length of an element in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively; n_s, n_d, k_s, k_e: the number of attributes in a signing, encryption, the number of signing, decryption key attributes, respectively; IBE: identity-based encryption; IBS: identity-based signature; $\mathbb{B}$: the scheme uses blockchain. We suppose the signcryption schemes hold up to 20 attributes.

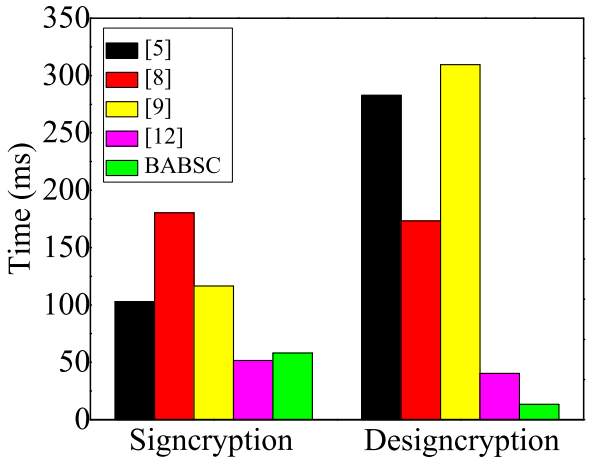

Fig. 9. The signcryption and designcryption cost.

cryption key are independent of the number of attributes. Hence, the communication and storage overhead are overcome.

As indicated in Fig. 9, the time of sigcryption algorithm in BABSC scheme is about 53.017 ms. However, the sigcryption time consumes 103.056 ms, 180.348 ms, 116.511 ms and 51.528 ms in [5,8,9,12], respectively. For desigcryption algorithm, the BABSC has the lowest time cost, which is equal to one bilinear pairing operation ($\mathbb{P}$=13.455 ms). While, the schemes in [5,8,9,12] require 282.693 ms, 173.196 ms, 309.465 ms and 40.365 ms to desicrypt operation respectively.

In the end, the experimental results show that BABSC is an effective and powerful scheme which support secure data sharing in cloud computing.

7. Conclusion

This work presented a new blockchain-based attribute-based signcryption scheme (BABSC) to secure data sharing in the cloud environment. The proposed BABSC has the advantage of using both blockchain and attribute-based signcryption. It provides secure data confidentiality and unforgeability. The performance analysis reveals that the BABSC does not only minimize the communication overhead, but also gives fast designcryption in the user side. Furthermore, BABSC enforces access control of the users and is suitable for the cloud computing. Future work will focus on the deployment of smart contracts on Ethereum.

Declaration of competing interest

The authors declared that they have no conflicts of interest to this work.

Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant No. 61872058) and Fundamental Research Funds for the Central Universities (Grant No. ZYGX2016J081).

References

[1] C. Modi, D. Patel, B. Borisaniya, A. Patel, M. Rajarajan, A survey on security issues and solutions at different layers of cloud computing, J. Supercomput. 63 (2) (2013) 561–592.
[2] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf.
[3] H.K. Maji, M. Prabhakaran, M. Rosulek, Attribute-based signatures: achieving attribute-privacy and collusion-resistance, IACR Cryptology ePrint Archive 2008 (2008) 328.
[4] M. Gagné, S. Narayan, R. Safavi-Naini, Threshold attribute-based signcryption, in: J.A. Garay, R. De Prisco (Eds.), Security and Cryptography for Networks, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 154–171
[5] J. Liu, X. Huang, J.K. Liu, Secure sharing of personal health records in cloud computing: ciphertext-policy attribute-based signcryption, Future Generat. Comput. Syst. 52 (2015) 67–76.
[6] Y. Zheng, Digital Signcryption or How to Achieve Cost (Signature & Encryption) Cost (Signature)+ Cost (Encryption), in: Annual International Cryptology Conference, Springer, 1997, pp. 165–179.
[7] A. Yin, H. Liang, On security of a certificateless hybrid signcryption scheme, Wireless Pers. Commun. 85 (4) (2015) 1727–1739.
[8] Y.S. Rao, A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing, Future Generat. Comput. Syst. 67 (2017) 133–151.
[9] Y.S. Rao, R. Dutta, Expressive attribute based signcryption with constant-size ciphertext, in: Progress in cryptology – AFRICACRYPT, International Publishing, Cham, Springer, 2014. 398–419
[10] Y.S. Rao, R. Dutta, Expressive bandwidth-efficient attribute based signature and signcryption in standard model, Information Security and Privacy, Springer International Publishing, Cham, 2014. 209–225
[11] C. Wang, J. Huang, Attribute-based signcryption with ciphertext-policy and claim-predicate mechanism, in: Computational Intelligence and Security (CIS), 2011 Seventh International Conference on, IEEE, 2011, pp. 905–909.
[12] H. Wang, Y. Song, Secure cloud-based ehr system using attribute-based cryptosystem and blockchain, J. Med. Syst. 42 (8) (2018) 152.
[13] G.W. Peters, E. Panayi, Understanding modern banking ledgers through blockchain technologies: future of transaction processing and smart contracts on the internet of money, Springer International Publishing, Cham, 2016. 239–278
[14] C.C Dong, Y. Wang, A. Aldweesh, P. Mc, A. van Moorsel, Betrayal, distrust, and rationality: smart counter-collusion contracts for verifiable cloud computing, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, 2017. 211–227
[15] X. Yue, H. Wang, D. Jin, M. Li, W. Jiang, Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control, J. Med. Syst. 40 (10) (2016) 218.
[16] P.C Mc, S.F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy, in: International Conference on Financial Cryptography and Data Security, Springer, 2017. 357–375
[17] Z. Wang, H. Zhang, X. Song, H. Zhang, Consensus problems for discrete-time agents with communication delay, Int. J. Control Autom. Syst. 15 (4) (2017) 1515–1523.
[18] J. Yang, Z. Lu, J. Wu, Smart-toy-edge-computing-oriented data exchange based on blockchain, J. Syst. Archit. 87 (2018) 36–48. http://www.sciencedirect.com/science/article/pii/S1383762118300638.
[19] H. Huang, X. Chen, Q. Wu, X. Huang, J. Shen, Bitcoin-based fair payments for outsourcing computations of fog devices, Future Generat. Comput. Syst. 78 (2018) 850–858.
[20] W. Tang, X. Zhao, W. Rafique, L. Qi, W. Dou, Q. Ni, An offloading method using decentralized p2p-enabled mobile edge servers in edge computing, J. Syst. Archit. 94 (2019) 1–13. http://www.sciencedirect.com/science/article/pii/S138376211830448X.

[21] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.
[22] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy (SP'07), IEEE, 2007, pp. 321–334.
[23] W. Ethereum, A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper 151, 2014, 1–32
[24] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, J.J. Kishigami, Blockchain contract: a complete consensus using blockchain, in: 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE), IEEE, 2015, pp. 577–578.
[25] T. Pandit, S.K. Pandey, R. Barua, Attribute-based signcryption: Signer privacy, strong unforgeability and ind-cca2 security in adaptive-predicates attack, in: International Conference on Provable Security Springer, 2014, pp. 274–290.
[26] G. Shanqing, Z. Yingpei, Attribute-based signature scheme, in: 2008 International Conference on Information Security and Assurance (ISA (2008), IEEE, 2008, 509–511
[27] B. Lynn, et al., pbc: the pairing-based cryptography library, http://crypto.stanford.edu/pbc.

Nabeil Eltayieb received the B.Sc. degree, in 2008 from the Faculty of Computer Science and Information Technology, University of Karary, Khartoum, Sudan. In 2016 received his Master egree from School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests in the areas of applied cryptography network security, with current focus on data security in cloud computing.

Rashad Elhabob received the B.Sc. degree, in 2010 from the Faculty of Computer Science and Information Technology, University of Karary, Khartoum, Sudan. In 2014 received his Master degree from Faculty of Mathematical Science, University of Khartoum, Khartoum, Sudan. He is currently pursuing the Ph.D. degree with the School of Software Engineering, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include cryptography and network security

Alzubair Hassan received the B.Sc. degree in Computer Science from University of Kassala in 2010. He received M.Sc. degree in Mathematical Science from University of Khartoum in 2013. He received his Ph.D. degree in computer science and technology from University of Electronic Science and Technology of China in 2018. Currently, he is a postdoc researcher at School of Computer Science and Educational Software, Guangzhou University. His current research interests include cryptography and network security.

Fagen Li is a professor in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China. He received his Ph.D. degree in Cryptography from Xidian University, Xi'an, P.R. China in 2007. From 2008 to 2009, he was a postdoctoral fellow in Future University-Hakodate, Hokkaido, Japan, which is supported by the Japan Society for the Promotion of Science (JSPS). He worked as a research fellow in the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan from 2010 to 2012. His recent research interests include cryptography and network security. He has published more than 100 papers in international journals and conferences. He is a member of the IEEE.