Annals of Telecommunications
https://doi.org/10.1007/s12243-018-00699-y

Efficient and privacy-preserving traceable attribute-based encryption in blockchain

Axin Wu[1,2] · Yinghui Zhang[1,2] · Xiaokun Zheng[1,2] · Rui Guo[1,2] · Qinglan Zhao[1,2] · Dong Zheng[1,2]

Received: 23 July 2018 / Accepted: 12 December 2018

Abstract

Attribute-based encryption, especially ciphertext-policy attribute-based encryption, plays an important role in the data sharing. In the process of data sharing, the secret key does not contain the specific information of users, who may share his secret key with other users for benefits without being discovered. In addition, the attribute authority can generate the secret key from any attribute set. If the secret key is abused, it is difficult to judge whether the abused private key comes from users or the attribute authority. Besides, the access control structure usually leaks sensitive information in a distributed network, and the efficiency of attribute-based encryption is a bottleneck of its applications. Fortunately, blockchain technology can guarantee the integrity and non-repudiation of data. In view of the above issues, an efficient and privacy-preserving traceable attribute-based encryption scheme is proposed. In the proposed scheme, blockchain technologies are used to guarantee both integrity and non-repudiation of data, and the ciphertext can be quickly generated by using the pre-encryption technology. Moreover, attributes are hidden in anonymous access control structures by using the attribute bloom filter. When a secret key is abused, the source of the abused secret key can be audited. Security and performance analysis show that the proposed scheme is secure and efficient.

Keywords CP-ABE · Fast ciphertext generation · Hidden policies · Public traceability

1 Introduction

Attribute-based encryption (ABE) [1] can implement the fine-grained access control [2] and data sharing [3]. In the scene of data sharing, users often use the ciphertext-policy attribute-based encryption (CP-ABE) scheme [4]. In the CP-ABE system, users can specify the access control structure in the encryption phase. In the decryption phase, users with secret keys satisfying the access structure can decrypt the ciphertext correctly. Because the secret key does not contain the specific information of the user, it is difficult to audit who deciphers the ciphertext. If a malicious user shares his secret key, which will be a threat to the security of the system. In addition, the attribute authority can generate the secret of any attribute set. When the secret key is abused, it is difficult to judge whether the abused private key comes from users or the attribute authority.

When data is stored in a distributed network [5], the access control structure usually leaks sensitive information. If these sensitive information is used by malicious users to make profits, which will be a great threat to enterprises or individuals. Besides, the efficiency of ABE is also a bottleneck of its applications.

Nowadays, more and more enterprises and individuals store data on third party servers, which reduces the local storage and management costs. However, the integrity and non-repudiation of data will not be guaranteed. Fortunately, blockchain technology [6, 7] can solve this problem very

✉ Dong Zheng
zhengdong_xupt@sina.com

Axin Wu
waxinsec@163.com

Yinghui Zhang
yhzhaang@163.com

Xiaokun Zheng
xiaokzheng@163.com

Rui Guo
guorui@xupt.edu.cn

Qinglan Zhao
zhaoqinglan@foxmail.com

1 National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications, Xi'an, 710121, People's Republic of China

2 Westone Cryptologic Research Center, Beijing, 100070, China

Published online: 03 January 2019

well. Blockchain is a distributed storage system that ensures the integrity and non-repudiation of data. If data is stored on the blockchain, data can not be tampered with. When the secret key is abused, it is more believable to audit the ownership of the secret key.

1.1 Our contribution

In view of the above issues, an efficient and privacy-preserving traceable attribute-based encryption in blockchain is proposed. Its advantages are as follows:

- Fast ciphertext generation: Before knowing the message to be encrypted, the user does a lot of precomputation based on public parameters. When the message to be encrypted is known, the ciphertext can be generated fast.
- Hidden policy: Hidden policy is achieved through the attribute bloom filter, which removes the mapping function between access control structures and attributes. Attributes are hidden in an anonymous access control structure.
- Publicly verifiable white-box traceability: When a private key is abused, any third party can judge whether the private key comes from the user or the attribute authority.

1.2 Related work

Since Sahai and Waters [1] proposed a scheme of fuzzy identity-based encryption, ABE has become a very hot research direction due to its wide application scenarios. For more flexible application of ABE, ABE is classified into the key-policy attribute-based encryption (KP-ABE) [8] and the ciphertext-policy attribute-based encryption (CP-ABE) [9, 10] in the scheme [11]. In the scene of data sharing, the CP-ABE scheme [4] is often used. In the CP-ABE system, the user can specify the access control structure in the encryption phase, and users whose attributes meet the access control structure can decrypt the ciphertext.

The efficiency of ABE is also the bottleneck of its application. In order to reduce the computation of the client, some computation can be outsourced to the server. Green et al. [12, 13] applied outsourced computing to ABE, which outsources a large number of linear pairs operations to the server and reduces the amount of client computing. In order to improve the efficiency of encryption phase, the online/offline technology is proposed. Before knowing the message to be encrypted, the user does do a lot of precomputation based on public parameters. When the message to be encrypted is knew, the ciphertext can be generated fast. The online/offline technology was first used in the signature scheme [14]. The online/offline technology was first used in ABE scheme in [15].

In practical applications, privacy protection [16, 17], data signature [18, 19], data storage [20], secret key management [21, 22] and ciphertext search [23, 24] are also important for data security. In the practical application of ABE, the access control structure will also be uploaded, which may leak some sensitive information. If some information is used by malicious users, it will be a great threat to individuals and enterprises. Some work is done in schemes [25–27]. Furthermore, the concept of ABE with partially hidden policy was proposed in the scheme [25], in which the values of each attribute is hidden. The privacy protection was achieved in scheme [26] by inner product encryption. Zhang et al. [27] achieved the attribute privacy protection by hiding access policy in ciphertexts.

In an distributed and dynamic network environment [28], because the secret key in ABE scheme does not contain the unique information of the user, the user can share their secret key without being discovered [29]. To solve the traceability problem of the secret key. A number of accountable ABE schemes have been proposed. An accountable ABE with wildcards access policy supporting AND gata was proposed in the scheme [30]. Unfortunately, a malicious user can transform a secret key into another effective secret key which cannot be traced. The scheme [31] and the scheme [32] proposed the white-box traceable CP-ABE and the black-box traceable CP-ABE respectively. Although the above scheme can trace the identity of a secret key, it can not audit the source of a secret key. An accountable ABE scheme with public verifiability was proposed in the scheme [33]. Because the authority knows the valid key of users, the scheme can not achieve accountability.

On the other hand, the ciphertext stored in third party may also be tampered [34–36]. Blockchain can ensure the integrity and non-repudiation of data, and properties of blockchains make them popular in application scenarios. For example, Zhang et al. proposed two blockchain-based fair payment protocols called BPay [37] and BCPay [38] for outsourcing services in cloud computing. The protocol BPay [37] is compatible with the Bitcoin blockchain based on an iterative all-or-nothing checking-proof protocol and a top-down checking method. However, the performance remains to be improved. At the cost of losing the compatibility with the Bitcoin blockchain, the protocol BCPay [38] realizes robust fair payment based on a one round all-or-nothing checking-proof protocol and hence is very efficient in terms of the computation cost and the number of transactions. If data is stored on the blockchain, the reliability of data will be enhanced. When the secret key is abused, the source of the secret key will be more credible.

The rest of the paper is organized as follows: the preliminary of the paper is introduced in Section 2; then, the relevant definitions are given in Section 3; after that,

the scheme and security analysis are presented in Sections 4 and 5, respectively. Finally, performance analysis and conclusions and future work are presented in Sections 6 and 7.

2 Preliminary

We will introduce the cryptographic primitives that are used in this section.

2.1 Bilinear pairing

For two cyclic groups G, G_T with the same prime number p, a linear map $e : G \times G \rightarrow G_T$ has the following properties:

- Bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for all $g, h \in G$ and $a, b \in Z_p$.
- Non-Degeneracy: the order of $e(g_1, g_1)$ in G_T is p, where $\exists g \in G$.
- Computability: e can be effectively computed.

2.2 Linear secret sharing schemes

Linear Secret Sharing Scheme (LSSS) over a set of parties $\mathbb{P}$ is composed of the secret sharing and the secret reconstruction. LSSS over Z_p, where p is a prime, is described as follows:

- The secret sharing: Each share for parties can be expressed as a vector over Z_p. The share-generating matrix M has l rows and n columns. For $j \in [1, ..., l]$, the j-th row of M is labeled by a party $\rho(i)$, where ρ is a function from $[1, ..., l]$ to $\mathbb{P}$. When a secret s is shared, random numbers $r_2, ..., r_n \in Z_p$ are selected, then the vector $\vec{v} = (s, r_2, ..., r_n)^\top$ can be obtained. $(M\vec{v})^\top$ is the shared vector of the secret s.
- The secret reconstruction: For the access control structure $\mathbb{M}$ in the LSSS. $S \in \mathbb{M}$ is defined as the a set. $I \subset \{1, ..., l\}$ is defined as $I = \{j : \rho(j) \in S\}$. If $\{\lambda_i\}$ are valid shares of the secret s, then constants $\{c_i\}_{i \in I}$ can be found in polynomial time, so that equality $\Sigma_{i \in I} c_i \lambda_i = s$ holds. Otherwise, you can't find such constants $\{c_i\}_{i \in I}$.

2.3 Attribute bloom filter

Bloom proposed a more efficient space data structure called the Bloom Filter (BF) [39] in 1970, which is used to determine whether an element exists in a set. The concrete process is as follows: an array of w bits and k independent hash functions $h_1, ..., h_k$ are selected. And there is such a relationship $h_i : \{0, 1\}^* \rightarrow [1, ..., w]$ for $1 \leq i \leq k$. At the initial state, the values in all positions of the array are set to 0. When an element x is added to the set, the value of the position $\{h_i(x)\}_{i \in [1,...,k]}$ in the array will be set to 1. If there is 0 in the position $\{h_i(x)\}_{i \in [1,...,k]}$, the element is not in the set, and if all positions $\{h_i(x)\}_{i \in [1,...,k]}$ are 1, the element may exist in the set. In such a situation, an element does not exist in the set, but all locations are 1, which is called misjudgement.

The attribute bloom filter (ABF) [40] is based on the garbled BF [41], which lowers false positive. In ABF, a string of length of λ bits cascaded by two fixed strings of the row number with L_r bits and the attribute with L_a bits will be constructed, where λ is the security parameter in the garbled BF. ABF can be described from two parts: the attribute bloom filter construction (ABFBulid) and the attribute bloom filter query (ABFQuery). They are described as follows:

- $ABFBuild(M, \rho) \rightarrow ABF$: This algorithm takes access control matrix (M, ρ) as input. Attributes related to the access policy and the corresponding row number in the access matrix M are cascaded. Then, the set of elements $S_e = \{i||at_e\}_{i \in [1,...,l]}$, where there is such a relationship $at_e = \rho(i)$ between the i-th row of M and the attribute at_e, can be obtained. If the row number i and the attribute at_e are not the largest bit lengths, the maximum bit length can be achieved by adding zeros on the left of the bits strings. ABF can be obtained by calling the garbled BF Building algorithm in [41] with S_e as an input.

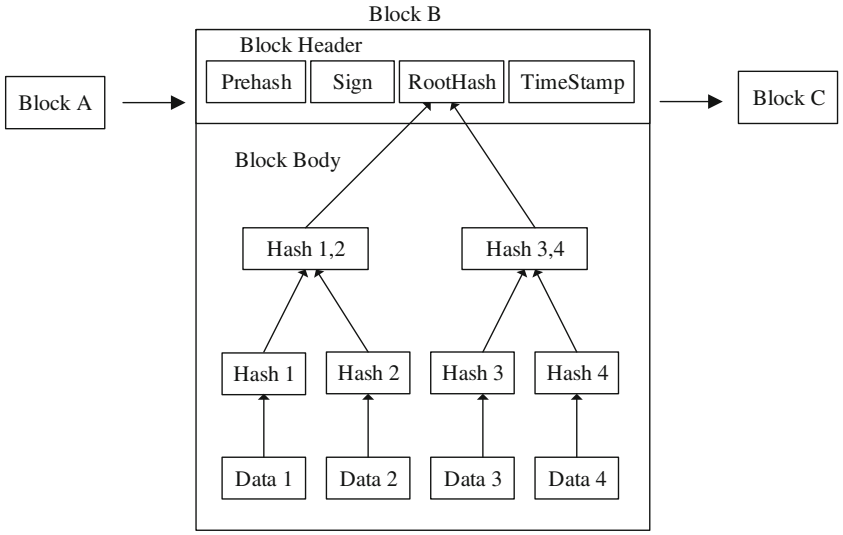
  In order to add the element e in the set S_e to the ABF, the algorithm first generates $k - 1$ λ-bit strings $r_1, r_2, ..., r_{k-1}$ randomly, and then calls the secret sharing scheme (k, k) to share e, and sets

  $$r_k = r_1 \oplus r_2 ... \oplus r_{k-1} \oplus e.$$

  After that, k independent and identically distributed hash functions h_i are used for hashing $h_i(at_e)$ for $i \in [1, ..., k]$, where $h_i(at_e)$ for $i \in [1, ..., k]$ expresses the position in ABF. Then the i-th element share r_i is stored in the position $h_i(at_e)$ in ABF. If an element x is added to ABF and the location $h_i(at_x)$ is occupied, this existing share can be used a share of the new elements.
- $ABFQury(S, ABF, h_i) \rightarrow \rho'$: The algorithm takes the attribute set S, ABF and some public hash functions h_i as input.

  For each attribute $at \in S$, the algorithm calls k hash functions h_i to get position index $h_i(at)$. Then the value r_i of the position $h_i(at)$ in ABF can be obtained.

Fig. 1 Structure of block



After that, e can be reconstructed through the following formula:

$$\begin{aligned} e &= r_1 \oplus r_2 \oplus ... \oplus r_{k-1} \oplus r_k \\ &= r_1 \oplus r_2 \oplus ... \oplus r_{k-1} \oplus r_1 \\ &\quad \oplus r_2 \oplus ... \oplus r_{k-1} \oplus e. \end{aligned}$$

where e exists in the form of $e = i||at_e$.

The string at_e can be obtained from the L_a bits on the right side of the string e. If there is zero on the left side of the string at_e, the zero will be removed. If at_e and at are the same, the attributes at exists in the access control structure. The row number r_w can be obtained from the L_r bits on the left side of the string e. If there is zero on the left side of the row number, the zero will be removed. Finally, the reconstructed attribute mapping as follows:

$$\rho' = \{(r_w, at)\}_{at \in S}.$$

In summary, in the algorithm $ABFBuild$, the attribute mapping is hidden. In the algorithm $ABFQury$, the attribute mapping is reconstructed. Furthermore, attributes can be completely hidden in the access control structure.

2.4 CDH assumption

Let g be a generator of a bilinear group G with the prime order p. Select randomly $u, v \in Z_p$ and $g_1 \in G$.

The advantage in breaking CDH assumption of the adversary A is defined as $Adv_{g,A}(\lambda) = |Pr[A(p, G, g_1^u, g_1^v) = g_1^{uv}]|$. We say that the CDH assumption [42] holds if $Adv_{g,A}(\lambda)$ can be ignored for any polynomial time A in the security parameter λ.

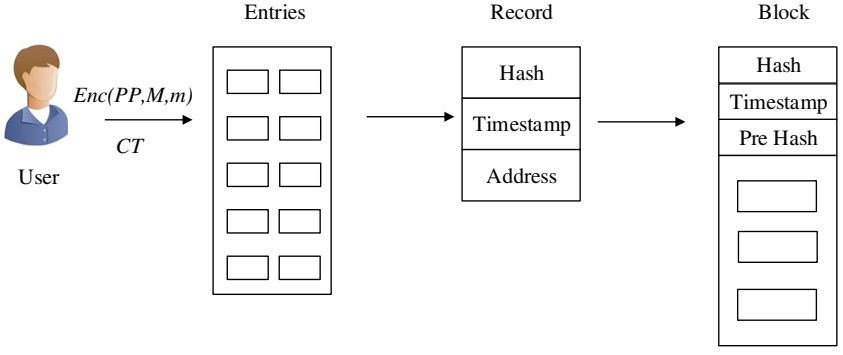
2.5 Reading and writing of files on blockchain

Blockchain is a distributed storage system, which can guarantee the integrity and non-repudiation of the data. The structure of blockchain is shown in Fig. 1.

When a user I wants to write files in the system, I encrypts the file m into a ciphertext CT. Then, the hash value $H(m)$ of the file m, the timestamp t_p a file is written and the address a_I of I are hashed, which is computed as $H_w = Hash(H(m), t_p, a_I)$. This process is described in Fig. 2.

When a user I wants to read files in the system, I decrypts the ciphertext CT into a file m'. Then, the hash

Fig. 2 Write on the blockchain



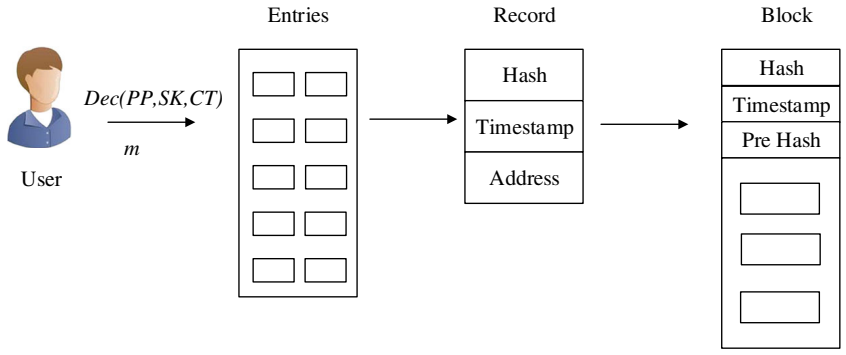


Fig. 3 Read on the blockchain

value $H(m')$ of the file m', the timestamp t_p a file is read and the address a_I of I are hashed, which is computed as $H_r = Hash(H(m), t_p, a_I)$. This process is described in Fig. 3.

3 Definition

We will describe the scheme and definition of the system in this section.

3.1 Definition of model

The system structure is shown in Fig. 4, which involves five entities.

- Data Owner (DO): DO is a data sharer, who encrypts the data to be shared and stores it on the blockchain.
- Data User (DU): DU is a consumer who wants to get data from the blockchain. Only legitimate users can decrypt data.
- Attribute Authority (AA): AA managers users in the system and publishes the system parameters. AA issues a secret key for the user based on the user's properties.
- Blockchain : Blockchain is used to store data and guarantees the integrity and non-repudiation of the data.
- Verification Center (VC): When a secret key is used illegally, VC can judge whether the secret key comes from the user or AA.

3.2 Definition of scheme

Our scheme is composed of the following algorithms:

- AASetup(λ) $\rightarrow$ PP, MSK: The algorithm inputs the security parameters of the system λ, and then outputs the public parameters of the system PP and the main private key of the system MSK.
- KeyGen(PP, MSK, I, S)$\rightarrow$ $x_I, P_I, SK_{I,S}$: This algorithm is composed of the following two sub-algorithms:
  - sExtract: The user inputs the public parameters PP, and then generates a secret key pair (x_I, P_I) according to his own identity I. x_I is kept secretly and P_I is public.
  - dExtract: The interaction is needed between AA and users about the algorithm. The user sends his identity I and the attribute S to AA. AA generates the partial secret key K, then sends K to the user secretly. I generates the signature σ of K. σ is sent to AA. Finally, AA generates the full secret key $SK_{I,S}$ of the user I.
- Enc(PP, (M, ρ), m) $\rightarrow$ CT_M, ABF: This algorithm is composed of the following three sub-algorithms:
  - Offline(PP)$\rightarrow$ IT: The algorithm inputs the system public parameters PP, and then outputs the intermediate ciphertext IT.

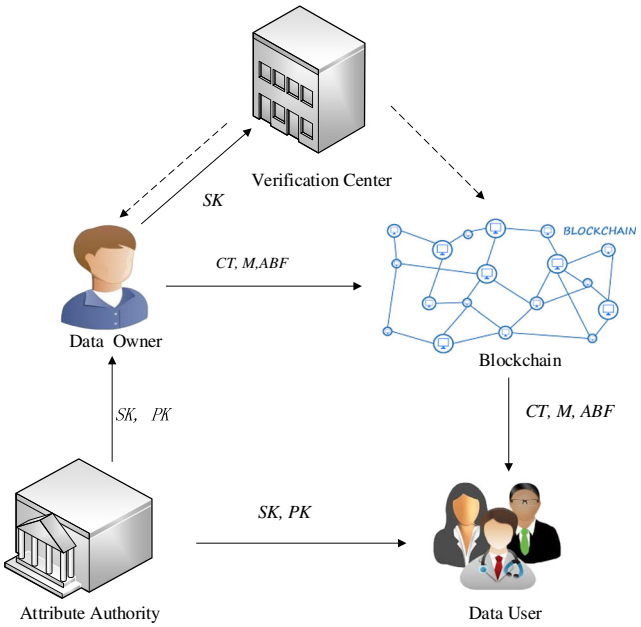


Fig. 4 System architecture

  - Online(IT, $(M, \rho)) \rightarrow CT$: The algorithm inputs intermediate ciphertext IT and the access control structure (M, ρ), and then outputs the ciphertext CT.
  - ABFBulid$(M, \rho) \rightarrow ABF$: The algorithm inputs the access control structure (M, ρ) and outputs ABF.

- Dec(PP, M, ABF, $SK_{I,S}) \rightarrow m$ or $\perp$: This algorithm is composed of the following two sub-algorithms:
  - ABFquery(S, ABF) $\rightarrow \rho'$: The algorithm inputs the attribute S and ABF, and then outputs the reconstructed attribute mapping $\rho' = \{r_w, at\}_S$.
  - Dec(CT,(M, ρ'), $SK_{I,S}) \rightarrow m$ or $\perp$: The algorithm inputs the ciphertext CT, the access control structure (M, ρ') and private key $SK_{I,S}$. If the attribute satisfies the access control structure, it outputs the message m; else, the algorithm terminates.

- Verify(PP, $SK_{I,S}$,S) $\rightarrow I$ or $\perp$: The algorithm inputs a secret key $SK_{I,S}$ related to the attribute set S. It outputs the identity I associated with $SK_{I,S}$ or an invalid symbol $\perp$.
- Audit(PP, $SK_{I,S}$, $S\bar{K}_{I,S}$, S) $\rightarrow I$ or CA: The algorithm inputs a secret key $SK_{I,S}$ related to the attribute set S and another decryption key $S\bar{K}_{I,S}$ from user I. It outputs the identity I or CA.

3.3 Definition of security model

The security of our scheme is determined by the following two security models.

Confidentiality: The security game between the adversary A and the simulator B is as follows:

- Init: A delivers an access control structure (M^*, ρ^*) that will be challenged.
- Setup: B runs the algorithm AASetup(λ) $\rightarrow$ PP, MSK, and then sends PP to A.
- Phase 1: A can query for secret keys connected with S_{at}. If the attribute S_{at} satisfies the access control structure (M^*, ρ^*), the query is terminated. Otherwise, the following inquiry can be carried out.
  - sExtract query: A submits an identity I to B adaptively, and a signature private key x_I is returned to A.
  - dExtract query: A submits an identity I and an attribute set S to B adaptively, and a decryption key $SK_{I,S}$ is returned to A.
- Challenge: A submits two new messages m_0 and m_1 to B, B then randomly selects $b \in \{0, 1\}$, finally generates the ciphertext CT_{M*} of m_b under (M^*, ρ^*), and returns CT_{M*} to A.
- Phase 2: The queries at this stage are the same as in phase 1.
- Guess: Finally, the guess b' is outputted.

The advantage of the adversary A is defined as

$$Adv(A = \left| Pr[b' = b] - \frac{1}{2} \right|.$$

Publicly verifiable white-box traceability: The security game between the adversary A and the simulator B is as follows:

- Setup: B runs the algorithm AASetup(λ) $\rightarrow$ PP, MSK, and then sends PP to A.
- Query phase: A can access sExtract query and dExtract query oracles simulated by B.
- Forge: Finally, a decryption private key $SK_{I,S}$ is outputted by A.

If the private key $SK_{I,S}$ about (I, S), where $SK_{I,S}$ can't come from the dExtract query, can pass the algorithm Verify, A wins the game. The advantage of the adversary A is defined as

$$Adv(A) = Pr[A(wins)].$$

4 Efficient and privacy-preserving traceable ABE

Assume that the matrix in the LSSS access control structure has at most P rows and ABF has an array of w bits. Our scheme is composed of the following several algorithms:

The scheme is composed of the following six algorithms: AASetup, KeyGen, Encrypt, Decrypt, Verify, and Audit. In the setup phase, AA runs the algorithm AASetup, gets the main secret private key of the system and publishes the system's public parameters PP simultaneously. In the secret key generation phase, AA performs the algorithm KeyGen according to attributes of the user, generates the corresponding secret key, and then returns the secret key to the user. In the encryption phase, DO first performs pre-encryption calculation according to the system parameters published by AA. Then DO can quickly generate ciphertext when knowing the message to be encrypted. After that, ABF is constructed, which removes the mapping function between the access control structure and attributes. In the decryption stage, DU first recovers the mapping function between the access control structure and attributes. Then, the ciphertext can be decrypted in the traditional way. In the verification and auditing stage, when the secret key

is illegally used, the source of the illegal secret key can be determined through these two algorithms by VC. Our scheme is as follows:

– AASetup(λ) $\rightarrow$ PP, MSK:

This algorithm takes a security parameter λ as input, and then randomly instantiates two cyclic groups G, G_T of prime order $p > 2^{\lambda}$. Then, a generator g of G, a linear map $e : G \times G \rightarrow G_T$ and $k + 2$ security hash functions $f_0 : \{0, 1\}^* \times G \rightarrow G$, $f_1 : \{0, 1\}^* \rightarrow Z_p$ and $h_i : \{0, 1\}^* \rightarrow [1, w]$ are randomly selected. AA randomly selects $y_i \in Z_p$ for each attribute $i \in U$, and calculates $H_i = g^{y_i}$. After that, AA randomly selects $a, \alpha \in Z_p$ and $g_1 \in G$. Finally, the system's public parameters PP and master key MSK are as follows:

$$PP = (G, G_T, p, e, g_1, g^a, e(g, g_1)^{\alpha}, \{H_i, \forall i \in U\}, f_0, f_1, h_1, h_2, ..., h_k)$$

and

$$MSK = (g_1^{\alpha})$$

– KeyGen(PP, MSK, I, S)$\rightarrow x_I, P_I, SK_{I,S}$:

The user's secret key $SK_{I,S}$ can be generated by the following interaction between AA and the user I.

- sExtract: A random number $x_I \in Z_p$ is randomly selected and $P_I = g^{x_I}$ is calculated by the user I. x_I is then used as the signature key of the user I and P_I is used as the public key of the user I.
- dExtract: AA randomly selects the element $r \in Z_p$, calculates $K = g^r$, and then sends K secretly to the user I. After the user I receives K, the user I signs a short signature $\sigma = f_0(K, P_I)^{x_I}$ with the signature key x_I and sends σ to AA. After AA receives the signature σ of the user I, AA then verifies the signature σ by $e(\sigma, g) = e(f_0(K, P_I), P_I)$. If the verification passes, then $h = f_1(\sigma, K, U)$, $K_1 = g_1^{\alpha h} g^{ar}$, and $K_i = H_i^r$, $\forall i \in S$ are calculated. The secret key $SK_{I,S}$ of the user I with attributes S is $SK_{I,S} = (U, \sigma, K, K_1, \{K_i : \forall i \in S\})$. Otherwise, the algorithm terminates.

– Encrypt(PP, (M, ρ), m) $\rightarrow CT_M, ABF$:

This algorithm is composed of the following three sub-algorithms:

- Offline: A random element $s \in Z_p$ is selected and then calculated: $key = e(g, g_1)^{\alpha s}$ and $C_2 = g^s$.

  After that, for 1 to P, random elements $\lambda_i', r_i \in Z_p$ are selected and then calculated: $C_{1,i} = g^{a\lambda_i'} \cdot H_i^{-r_i}$ and $C_{2,i} = g^{r_i}$. Finally, the intermediate ciphertext is $IT = (key, C_2, s, \{\lambda_i', r_i, C_{1,i}, C_{2,i}\}_{i \in [1,...,P]})$.
- Online: When the encrypted information m is known, DO makes the linear secret sharing of the secret s according to the specified access control structure. And the shared vectors $\lambda_1, \lambda_2, ..., \lambda_l$ can be obtained. Then IT is inputted and the ciphertext can be generated.

  $C = m \cdot key$ and $C_{3,i} = \lambda_i - \lambda_i'$.

  $$CT = (C, C_2, \{C_{1,i}, C_{2,i}, C_{3,i}\}_{i \in [1,...,l]}).$$
- ABFbuild: The data owner calls the algorithm *ABFBuild* with the access control structure (M, ρ), after which the attribute mapping function is removed. Finally, the ciphertext CT, the access control structure M and ABF are uploaded to the cloud server.

– Decrypt(PP, M, ABF, $SK_{I,S}$) $\rightarrow m$ or $\perp$:
This algorithm is composed of the following two sub-algorithms:

- ABFQuery: The data users calls the algorithm *ABFQuery* with the attribute S, ABF, and PP. If the attribute mapping function cannot be reconstructed, the algorithm terminates. If it can be reconstructed, then traditional decryption operation is performed.
- Decrypt: DU then decrypts the ciphertext CT based on his secret key SK, and the reconstructed attribute mapping function. If the user's attributes satisfy the access control structure, the user can decrypt correctly, otherwise the algorithm will terminate.

$$key = \left(\frac{e(C_2, K_1)}{(\prod_{i \in S} e(C_{1,i} \cdot g^{aC_{3,i}}, K) \cdot e(C_{2,i}, K_i))^{w_i}} \right)^{\frac{1}{h}} = e(g, g_1)^{\alpha s}.$$

Finally, the plaintext can be obtained:

$$m = \frac{C}{key}.$$

– Verify (PP, $SK_{I,S}$,S) $\rightarrow I$ or $\perp$:

The secret key $SK_{I,S}$ has the form $SK_{I,S} = (U, \sigma, K, K_1, \{K_i : \forall i \in S\})$. U, σ and K have been signed by user and CA, so U, σ and K can not be masked. But the adversary can share K, K_1 and $\{K_i : \forall i \in S\}$ to hide U and σ. Moreover, the adversary can randomly select $d_1, d_2 \in Z_p$, then calculate $D_1 = (K_1)^{d_1}$, $\{D_i = (K_i)^{d_2} : i \in S\}$, and finally share $SK_{I,S} = (U, \sigma, K, D_1, \{D_i : \forall i \in S\}, d_1, d_2)$. Therefore,

- *Case 1*: If the exposed secret key is the form $SK_{I,S} = (U, \sigma, K, K_1, \{K_i : \forall i \in S\})$, the following operations are done.

  First check whether $e(K_1, g) = e(g_1, g)^{\alpha h} \cdot e(g^a, K)$ holds? if it doesn't hold, the algorithm terminates, else let $S' \in S$ satisfy the equation $e(K_i, g) = e(K, H_i)$, if S' is an empty set, the algorithm terminates, else the algorithm outputs the identity I related to $SK_{I,S}$.
- *Case 2*: If the exposed secret key is the form $SK_{I,S} = (U, \sigma, K, D_1, \{D_i : \forall i \in S\}, d_1, d_2)$, the following operations are done.

  First check whether $e(B, g) = (e(g_1, g)^{\alpha h} \cdot e(g^a, K))^{b_1}$ holds? if it doesn't hold, the algorithm terminates, else let $S' \in S$ satisfy the equation $e(B_i, g) = e(K, H_i)^{b_2}$, if S' is an empty set, the algorithm terminates, else the algorithm outputs the identity I related to $SK_{I,S}$.

– Audit(PP, $SK_{I,S}$, $S\bar{K}_{I,S}$, S) $\rightarrow I$ or CA:

  This algorithm will be able to judge whether the abused secret key is related to AA or the user. The concrete process is as follows:

- *Case 1*: Give a secret key that can be verified by the algorithm *Verify*, then check whether the equation $e(\sigma, g) = e(f_0(K, P_I), P_I)$ holds. If it does not hold, the algorithm is terminated.
- *Case 2*:Else, if the identity I denies the ownership of the secret key $SK_{I,S} = (U, \sigma, K, K_1, \{K_i : \forall i \in S\})$. The auditor will require I to submit the secret key. In order to prove his innocence, I submits his secret key $SK_{I,S} = (U, \sigma, K, K_1, \{K_i : \forall i \in S\})$ to the auditor. Then the auditor verifies whether the secret key $SK_{I,S} = (U, \sigma, K, K_1, \{K_i : \forall i \in S\})$ can pass the verification algorithm. If it passes, the auditor judges that the secret key is illegally used by CA, otherwise, the identity I is outputted.

5 Analysis of security

Theorem 1 *If the adversary A can break through our scheme with adversary ε, there will be an adversary A_1 to break through the scheme* [43].

Proof For the convenience of proof. We define our scheme as $\prod_O = (Setup_0, Keygen_0, Encrypt_0, Decrypt_0, Verify_0, Audit_0)$ and denote the scheme in [43] as $\prod_C = (Setup_C, sExtract_C, dExtract_C, Encrypt_C, Decrypt_C, Verify_C, Audit_C)$. The interactive game between the simulator B and the adversary A as well as challenger C of the scheme [43] is as follows:

– Init: A delivers an access control structure M^* that will be challenged to B. B then forward M^* to C.
– Setup: When C receives M^*, C runs the algorithm $Setup_C$ to obtain the system's public parameters $PP = (G, G_T, p, e, g_1, g^a, e(g, g_1)^\alpha, \{H_i, \forall i \in U\}, f_0, f_1)$, then returns $PP_C = (G, G_T, p, e, g_1, g^a, e(g, g_1)^\alpha, \{H_i, \forall i \in U\}, f_0, f_1)$ to B. B selects k hash functions $\{h_i\}_{i\in[1,\ldots,k]}$, and then sends $PP_C = (G, G_T, p, e, g_1, g^a, e(g, g_1)^\alpha, \{H_i, \forall i \in U\}, f_0, f_1, \{h_i\}_{i\in[1,\ldots,k]})$ to A.
– Phase 1: The secret key generation algorithm $Keygen_0$ is the same as the $sExtract_C$ and $dExtract_C$, so A can get queries of private key from C through B, where B forwards requests of A to C.
– Challenge: B sends two equal length message m_0 and m_1 to C. Then C calls the algorithm $Encrypt_C$, and then returns $CT = (C = m_b \cdot e(g_1, g)^{\alpha s}, C_2 = g^s, C'_{1,i} = g^{a\lambda_i} H^{-r_i}_{\rho(i)}, C'_{2,i} = g^{r_i})$ to the B. Then

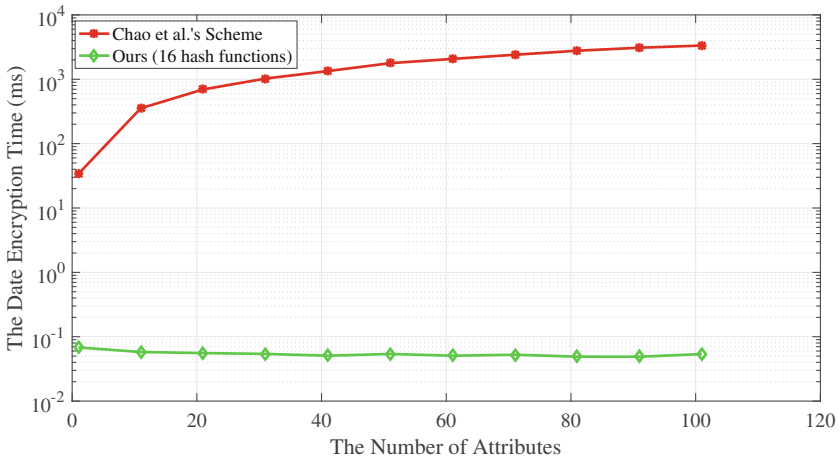


Fig. 5 Efficiency comparison of data encryption

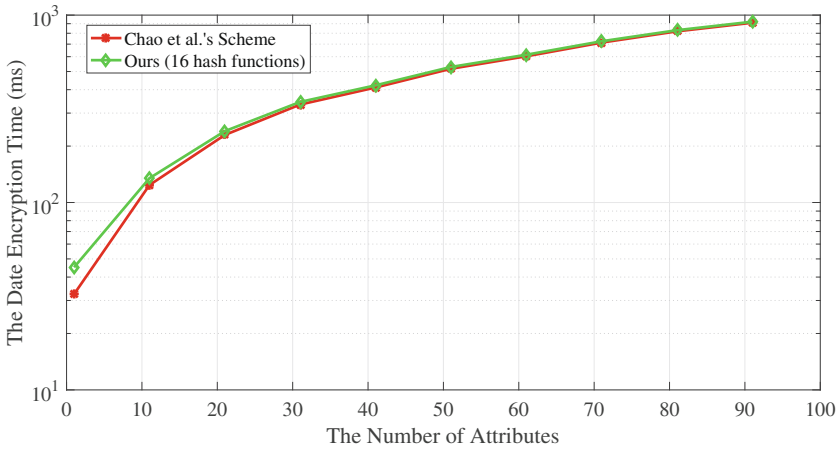


Fig. 6 Efficiency comparison of data decryption

random numbers $z_1, z_2, ..., z_{|S|} \in Z_p$ are selected. After that, the ciphertext is as follows:

$$C_{1,i} = C'_{1,i} \cdot g^{-az_i}, C_{2,i} = C'_{2,i}, C_{3,i} = z_i.$$

Then, $CT^* = (C_2, \{C_{1,i}, C_{2,i}, C_{3,i}\})$.

Finally, B outputs a guess m_b that is encrypted, then calculates $key_b = C/m_b$, and then sents (key_b, CT^*) to A.

– Phase 2: This process is the same as Phase 1.
– Guess: A outputs a bit t_b.

If $t_b = 0$, then we think the adversary A guesses that key_b is a secret key encapsulation by CT^*. Finally B outputs t. If $t_b = 1$, then we think the adversary A guesses that key_b is a random key. Finally B outputs $1 - t$. If A has the advantage ε to break through our scheme, B will have the same advantage to break the scheme [43]. □

Theorem 2 *If the adversary A against our scheme has an advantage ε to generate a decryption key. The simulator B can solve the CDH assumption with the same advantage.*

The proof of public verification is based on the signature's unforgeability of identity related to the decryption key and the CDH assumption. The details can be referred to the scheme [43].

6 Performance analysis

The scheme we proposed is compared with the scheme [43] in this section. In order to make the experimental results more accurate, the efficiency comparisons were conducted on the same platform including the encryption phase and the decryption phase as shown in Fig. 5 and in Fig. 6. These two figures reflect the average performance, and the number of simulations is one hundred at each attribute node.

From the Fig. 5, in the encryption phase, the efficiency of our scheme is obviously better than the scheme [43]. That's because pre-encryption technology is employed, which does a lot of precomputation before the message to be encrypted is known. Next, the ciphertext can be quickly generated when the message to be encrypted is known. From the Fig. 6, in the decryption phase, although our scheme is slightly lower than the scheme [43] in efficiency, our scheme additionally adds the function of the attribute hiding. In a word, our scheme has been much more efficient in the encryption phase, and no more burdens are added on the decryption phase. Besides, our scheme additionally adds the function of the attribute hiding.

7 Conclusion and future work

In view of the efficiency of ABE, the privacy protection of the attribute and the abuse of the secret key, the efficient and privacy-preserving traceable attribute-based encryption in blockchain is proposed. In order to solve the problem of efficiency of ABE and the privacy protection problem of attributes, the pre-encryption technology and ABF are applied to our scheme, which improves the encryption efficiency of ABE and hides attributes in an anonymous access control structure. In order to solve the abuse of the secret key, the user's signature and the main secret key of AA are embedded in the user's secret key. When a secret key is abused, any third party organization can judge whether the private key comes from the user or the attribute authority.

Our scheme can improve the efficiency of the encryption phase, but the efficiency of decryption phase is not improved. In the next work, we will improve the efficiency of the decryption phase.

Acknowledgements This work is supported by National Key R&D Program of China (No. 2017YFB0802000), National Natural Science Foundation of China (No. 61772418, 61472472, 61402366), Natural Science Basic Research Plan in Shaanxi Province of China (No. 2018JZ6001, 2015JQ6236), and the Youth Innovation Team of Shaanxi Universities. Yinghui Zhang is supported by New Star Team of Xi'an University of Posts and Telecommunications (No. 2016-02).

References

1. Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: International conference on theory and applications of cryptographic techniques, pp 457–473
2. Zhang Y, Zheng D, Guo R, Zhao Q (2018) Fine-grained access control systems suitable for resource-constrained users in cloud computing. Computing and Informatics 37(2):327–348
3. Zhang Y, Wu A, Zheng D (2018) Efficient and privacy-aware attribute-based data sharing in mobile cloud computing. J Ambient Intell Humaniz Comput 9(4):1039–1048
4. Zheng D, Wu A, Zhang Y, Zhao Q (2018) Efficient and privacy-preserving medical data sharing in internet of things with limited computing power. IEEE Access 6:28019–28027
5. Wu A, Zheng D, Zhang Y, Yang M (2018) Hidden policy attribute-based data sharing with direct revocation and keyword search in cloud computing. Sensors(Basel, Switzerland) 18(7):1–17
6. Gaetani E, Aniello L, Baldoni R, Lombardi F, Margheri A, Sassone V (2017) Blockchain-based database to ensure data integrity in cloud computing environments. In: Italian conference on cybersecurity
7. Hari A, Lakshman TV (2016) The internet blockchain: a distributed, tamper-resistant transaction framework for the internet. In: ACM workshop on hot topics in networks, pp 204–210
8. Ostrovsky R, Sahai A, Waters B (2007) Attribute-based encryption with non-monotonic access structures. In: CCS 07 ACM conference on computer & communications security, pp 195–203
9. Li J, Chen X, Chow SSM, Huang Q, Wong DS, Liu Z (2018) Multi-authority fine-grained access control with accountability and its application in cloud. J Netw Comput Appl 112:89–96
10. Zhang Y, Zheng D, Deng RH (2018) Security and privacy in smart health: efficient policy-hiding attribute-based access control. IEEE Internet Things J 5(3):2130–2145
11. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: ACM conference on computer and communications security, pp 89–98
12. Green M, Hohenberger S, Waters B (2011) Outsourcing the decryption of ABE ciphertexts. Usenix Conference on Security 2011(3):1–16
13. Li J, Huang X, Li J, Chen X, Xiang Y (2014) Securely outsourcing attribute-based encryption with checkability. IEEE Trans Parallel Distrib Syst 25(8):2201–2210
14. Even S, Goldreich O, Micali S (1996) Online/offline digital signatures. J Cryptol 9(1):35–67
15. Hohenberger S, Waters B (2014) Online/Offline attribute-based encryption. In: International workshop on public key cryptography, pp 293–310
16. Zhang Y, Li J, Zheng D, Li P, Tian Y (2018) Privacy-preserving communication and power injection over vehicle networks and 5G smart grid slice. J Netw Comput Appl 122:50–60
17. Zhang Y, Shu J, Liu X, Li J, Zheng D (2018) Security analysis of a large-scale concurrent data anonymous batch verification scheme for mobile healthcare crowd sensing. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2018.2862381
18. Wang X, Zhang Y, Zhu H, Jiang L (2018) An identity-based signcryption on lattice without trapdoor. Journal of Universal Computer Science
19. Li T, Chen W, Tang Y, Yan H (2018) A homomorphic network coding signature scheme for multiple sources and its application in IoT. Security and Communication Networks, 2018. https://doi.org/10.1155/2018/9641273
20. Zhang Y, Yang M, Zheng D, Lang P, Wu A, Chen C (2018) Efficient and secure big data storage system with leakage resilience in cloud computing. Soft Comput 22(23):7763–7772
21. Li J, Li J, Chen X, Jia C, Lou W (2015) Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans Comput 64(2):425–437
22. Li J, Chen X, Li M, Li J, Lee PPC, Lou W (2014) Secure deduplication with efficient and reliable convergent key management. IEEE Trans Parallel Distrib Syst 25(6):1615–1625
23. Gao C, Lv S, Wei Y, Wang Z, Liu Z, Cheng X (2018) M-SSE: an effective searchable symmetric encryption with enhanced security for mobile devices, vol 6
24. Zhang Y, Deng RH, Shu J, Yang K, Zheng D (2018) TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain. IEEE Access 6:31077–31087
25. Nishide T, Yoneyama K, Ohta K (2008) Attribute-based encryption with partially hidden encryptor-specified access structures. In: International conference on applied cryptography and network security, pp 111–129
26. Lai J, Deng RH, Li Y (2011) Fully secure cipertext-policy hiding CP-ABE. In: International conference on information security practice and experience, pp 24–39
27. Zhang Y, Chen X, Li J, Wong DS, Li H, You I (2017) Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. Inf Sci 379:42–61
28. Wang H, Zheng Z, Wu L, Li P (2017) New directly revocable attribute-based encryption scheme and its application in cloud storage environment. Clust Comput 20(3):2385–2392
29. Zhang Y, Li J, Zheng D, Chen X, Li H (2017) Towards privacy protection and malicious behavior traceability in smart health. Pers Ubiquit Comput 21(5):815–830
30. Li J, Ren K, Kim K (2009) A2BE: accountable attribute-based encryption for abuse free access control. IACR Cryptology ePrint Archive 2009:118
31. Liu Z, Cao Z, Wong DS (2013) White-Box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. IEEE Trans Inf Forensics Secur 8(1):76–88
32. Li J, Huang Q, Chen X, Chow SSM, Wong DS, Xie D (2011) Multi-authority ciphertext-policy attribute-based encryption with accountability. In: ACM symposium on information, computer and communications security, ASIACCS 2011, Hong Kong, China, March, pp 386–390
33. Yu G, Cao Z, Zeng G, Han W (2016) Accountable ciphertext-policy attribute-based encryption scheme supporting public verifiability and nonrepudiation. In: International conference on provable security, pp 3–18
34. Chen X, Li J, Weng J, Ma J, Lou W (2014) Verifiable computation over large database with incremental updates. In: European symposium on research in computer security, pp 148–162
35. Chen X, Li J, Huang X, Ma J, Lou W (2015) New publicly verifiable databases with efficient updates. IEEE Trans Dependable Secure Comput 12(5):546–556
36. Meng W, Tischhauser EW, Wang Q, Wang Y, Han J (2018) When intrusion detection meets blockchain technology: a review. IEEE Access 6(99):10179–10188
37. Zhang Y, Deng Rh, Liu X, Zheng D (2018) Outsourcing service fair payment based on blockchain and its applications in cloud computing. IEEE Transactions on Services Computing. https://doi.org/10.1109/TSC.2018.2864191

38. Zhang Y, Deng Rh, Liu X, Zheng D (2018) Blockchain based efficient and robust fair payment for outsourcing services in cloud computing. Inf Sci 462:262–277
39. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426
40. Yang K, Han Q, Li H, Zheng K, Su Z, Shen X (2017) An efficient and fine-grained big data access control scheme with privacy-preserving policy. IEEE Internet Things J 4(2):563–571
41. Dong C, Chen L, Wen Z (2013) When private set intersection meets big data: an efficient and scalable protocol. In: ACM SIGSAC conference on computer & communications security, pp 789–800
42. Seo JH (2014) Short signatures from diffie-hellman, revisited: sublinear public key, CMA security, and tighter reduction. IACR Cryptology ePrint Archive 138:2014
43. Yuan C, Xu M, Si X, Li B (2017) Blockchain with accountable CP-ABE: how to effectively protect the electronic documents. In: 2017 IEEE 23rd international conference on parallel and distributed systems (ICPADS), pp 800–803. https://doi.org/10.1109/ICPADS.2017.00111