Mathematical Biosciences and Engineering

MBE, 18(4): 3559–3577.
DOI: 10.3934/mbe.2021179
Received: 30 November 2020
Accepted: 14 April 2021
Published: 23 April 2021

http://www.aimspress.com/journal/MBE

Research article

An efficient attribute-based access control system with break-glass capability for cloud-assisted industrial control system

Yuanfei Tu[1,2,3], Jing Wang[1,*], Geng Yang[2,3] and Ben Liu[1]

[1] College of Electrical Engineering and Control Science, Nanjing Tech University, Nanjing 211800, China
[2] College of Computer Science & Technology, Nanjing University of Post & Telecommunication, Nanjing 210003, China
[3] Jiangsu Key Laboratory of Big Data Security& Intelligent Processing, Nanjing 210023, China

* **Correspondence:** Email: wj_cec@126.com.

Abstract: In the era of Industry4.0, cloud-assisted industrial control system (ICS) is considered to be the most promising technology for industrial processing automation systems. However, the emerging attack techniques targeted at ICS underlines the importance of data security. To protect the data from the unauthorized accesses, attribute-based encryption is utilized to meet the requirement of confidentiality and access control demand of an open network environment. In ICS scenarios, it is critically important to offer the timely and efficient service, especially in the emergency situations. This paper proposes an efficient access control strategy that enables two access modes: attribute-based access and emergency break-glass access. Normally, users can access the encrypted data as long as their attributes satisfy the access policy specified by the data owner. In emergency cases, emergency situation handlers can override the access control policy of the encrypted data by the break-glass access capability. To eliminate the overhead for data consumers, the scheme outsources the data decryption and policy updating to the semi-trusted fog and cloud. The scheme also implements the CP-ABE scheme in terms of an asymmetric Type-3 pairings instead of the symmetric Type-1 pairings, which are inefficient and have security issues. Finally, the paper analyses the security of the scheme, evaluates its performance, and compares it with related works.

Keywords: attribute based access; break-glass; asymmetric Type-3 pairings; efficiency; cloud-assisted industrial control system

1. Introduction

In order to seek more intelligent and flexible industrial solutions, enterprises resort to cloud-assisted industrial control system (ICS), which integrates cloud computing, fog computing, Internet of Things (IoT) and Supervisory Control and Data Acquisition (SCADA) systems to perceive the changing state in the real world and acquire enormous amounts of data and achieve optimization control [1–3].

However, this compelling paradigm is confronted with various security and privacy risk, such as unauthorized access and information disclosure [4–6]. Encryption plays a crucial role in protecting the data confidentiality. Using encryption methods, a secure communication channel in the cloud-assisted ICS framework can be established [7–10]. While in the cloud environment where multiple users share the data, the data owner would have to share the data according to some access control policy. Therefore, fine-grained access control has been increasingly recognized as a critical issue.

Attribute-based encryption (ABE) [11] is a promising cryptographic technology that is capable of achieving confidentiality protection and fine-grained access control simultaneously. ABE schemes are classified into two categories, key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE) [12]. In the KP-ABE scheme, a ciphertext is associated with a set of attributes and a user's decryption key is associated with a tree access structure. Only if the attributes associated with the ciphertext satisfy the tree access structure can be decrypted by the user. In the CP-ABE scheme, the ciphertext is encrypted according to an access policy selected by the data owner, while the decryption key is generated with respect to a set of attributes. As long as the set of attributes satisfies the access policy, the decryption key can be used to decrypt the ciphertext. Using the CP-ABE scheme, the data owner can easily determine who can access the data by setting policies in the ciphertext, which is conceptually closer to traditional access control models such as role-based access control. Therefore, the CP-ABE has been adopted in the cloud computing system to implement the fine-grained attribute-based access control (ABAC) model.

The cloud-assisted ICS requires much faster response than other regular cloud computing systems do. Therefore, users can not simply use some ABE schemes for the ICS environment. One of the main drawbacks of ABE is that decryption involves expensive pairing operations and the size of the ciphertext and the duration of decryption grows with the complexity of the access policy. It would be a challenge for a production operator to complete the decryption on their resource-constrained devices, such as PDA, and it hinders the effort to migrate ABE to the ICS.

How to enable the data owner to update the access policy quickly in the CP-ABE scheme without privacy leakage becomes another research hotspot. In the CP-ABE scheme, the access policy is embedded in the encrypted data. To update the access policy, the data should be re-encrypted under the new access policy. As a result, the data owner undertakes the re-encryption caused by the policy update but the re-encryption causes a large computational overhead.

In addition, most of the ABE schemes are described in symmetric pairings. These constructions are considered rather inefficient, and have security vulnerabilities [13]. Therefore, the faster asymmetric pairings (Type-3) have been the recommended choice by experts [14].

Although several ABE schemes have been modified to support specific requirements, they are not applicable in emergency situations. When an accident occurs, for example, a chemical tank is on fire, the fire-fighters need to obtain the detailed information such as the type and quantity of materials, material condition of adjacent tanks, temperature of relevant storage tanks, concentration of

combustible and toxic gases in the environment, the status of inlet and outlet valves and so on. At this point, the ICS administrator or its data owner may be unavailable. Thus, it's necessary to construct a break-glass access mechanism [15] that allows the emergency situation handlers (ESH) or rescuers who don't have access privileges can override the access policy and access the plaintext data quickly. Besides, to keep the emergency access from being abused, each break-glass access should be documented for later audits.

This paper proposes a new asymmetric pairing-based CP-ABE, namely CP-ABE with break-glass, and makes it more efficient and suitable for the cloud-assisted ICS. The scheme protects data confidentiality and effectively controls user access to resources with two modes. The paper has the following contributions.

1) This paper proposes an outsourced CP-ABE scheme with policy updating. The scheme outsources most of the decryption and the policy updating to the semi-trusted fog and cloud, thus reducing the computation on the user side.

2) Most of the ABE mechanisms are constructed by symmetric prime-order (Type-1) pairings, however, the pairings have been considered inefficient and insecure. Therefore, this paper describes CP-ABE scheme in terms of an asymmetric Type-3 pairings. Although researchers have rephrased some traditional CP-ABE schemes in asymmetric pairings [16], this is the first time (to authors' knowledge) that an outsourced CP-ABE with policy updating under the asymmetric setting is proposed.

3) We propose an improved CP-ABE with integrated break-glass access capability. In emergency, the ESH can utilize a password pre-shared by the data owner to break the glass and obtain the plaintext data.

The rest of the paper is organized as follows: Section 2 presents an overview of related works. Section 3 introduces the preliminaries and Section 4 provides the system architecture and workflow. Section 5 provides detailed construction method. In Section 6, the paper analyses the security of the proposed scheme, evaluates the performance, and presents the quantitative evaluation of the scheme, followed by a conclusion in Section 7.

2. Related works

Attribute-based Encryption (ABE) was proposed by Sahai and Waters [17] to meet the requirement of confidentiality and access control demand of an open network environment such as cloud computing. In 2007, Bethencourt et al. [18] proposed the first CP-ABE construction. The scheme adopts threshold gate to build access policy and is proven to be secure in the generic group model. However the expression is not flexible enough. It is only suitable for applications with simple access policy requirements. Waters [19] proposed an expressive CP-ABE scheme that can support and/or and threshold operations. The access structure of this scheme is based on the LSSS matrix, and the scheme is proven secure in the standard model. Then Rouselakis et al. [20] constructed a large universe CP-ABE on prime order bilinear pairings.

Considering a revoked employee may attempt to access the files using his old key, Sahai et al. [21] discussed the access policy updating problem in CP-ABE, however, their scheme only allows a ciphertext to be re-encrypted to a more restrictive policy. Lai et al. [22] proposed an adapt CP-ABE which can update the access policy in a ciphertext to a new policy without retrieving and decrypting the encrypted data. Yang et al. [23] developed a dynamic policy updating method for LSSS structure. With an update key, the proxy can update the ciphertext from the previous access policy to the new policy.

Because the ABE schemes [17–23] require additional pairing operations in decryption and the ciphertext size and the cost of decryption grow with the size of the access policy, the CP-ABE brings in a large computation cost compared to standard public key cryptography. To overcome this problem, Green et al. [24] presented a method for outsourcing the decryption of CP-ABE ciphertext. Applying Green's idea, Tu et al. [25] presented a multimedia data forwarding scheme based on the CP-ABE and outsourced most of the computation to the cloud.

Pairing-based cryptography (PBC) is an important ingredient in the construction of ABE. Most of the current ABE schemes [17–25] are described in the context of the symmetric Type-1 pairings, which are implemented on super singular elliptic curves, with maximum embedding degrees of 2, 4 and 6 respectively. Whereas the asymmetric Type-3 pairings support embedding degrees greater than 6 (on elliptic curves). To achieve the same security level, the Type-3 pairings are more efficient than Type-1 pairings. Besides, the symmetric pairings have been faced with many security threats, such as attacks by Kim and Barbulescu [13]. The attack can reduce the complexity of solving the discrete logarithm problem (DLP). Therefore, Scott [16] described a CP-ABE scheme in the context of the asymmetric Type-3 pairings due to Waters [19]. Morales-Sandoval et al. [26] rephrased the large universe ABE [27] using the asymmetric Type-3 pairings, and the experiment showed that the asymmetric type F curve (BN-curve) can lead to a better performance.

Considering the emergency access to data without the credentials, Scafuro [28] introduced the concept of break-glass encryption for cloud storage that people can break the encrypted data at most one time in a traceable way. Then, to make ABE suitable for emergency situations, Brucker et al. [29] first integrated ABE with the break-glass encryption. They use a hierarchy of emergency attributes as override attributes to access the resource. Schefer-Wenzl et al. [30] proposed a generic model to integrate the break-glass policy into RBAC. Aski et al. [31] put forward an idea of integrating ABE with break-glass capabilities. They used a pre-shared password as an attribute to extract the break-glass key, but lacking of the concrete algorithm implementation. Oliveira et al. [32] proposed a break-glass protocol based on CP-ABE. They constructed an emergency policy in the ciphertext, and the ESH can utilize emergency attributes to access the resource. Zhang et al. [33] also proposed a break-glass access control scheme for emergency situation, but it could not realize fine-grained access control. Recently, Yang et al. [34] proposed a concrete access control system, which simultaneously supports break-glass access control and ABAC. However, they didn't show how to identify the malicious users who abuse the key. In break-glass encryption, the encrypted data should be broken in a traceable way.

3. Preliminaries

3.1. Bilinear maps

Definition 1 (bilinear maps) Let G_1, G_2 and G_T be groups of prime order p. A bilinear map is a function $e: G_1 \times G_2 \to G_T$ with the following properties:

1) Bilinear: For all $g_1 \in G_1$ and $g_2 \in G_2$, $a, b \in Z_p$, there is $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$.

2) Non-degeneracy: $e(g_1, g_2)$ generates G_T.

3) Efficiency: There exists an efficient algorithm to output the bilinear group $(p, G_1, G_2, G_T, e, g_1, g_2)$ and compute $e(g_1, g_2)$ for $\forall g_1 \in G_1, g_2 \in G_2$.

Galbraith et al. [14] distinguish three types of pairings:

Type-1: $G_1 = G_2$.

Type-2: $G_1 \neq G_2$ and there exists an efficient homomorphism $\psi: G_2 \rightarrow G_1$.

Type-3: $G_1 \neq G_2$ there are no efficiently computable homomorphisms between G_1 and G_2.

Type 3 is the only choice which offers good performance and flexibility for high security parameters, and yet this choice does not permit a homomorphism from G_2 to G_1. Hence, this paper designs the CP-ABE scheme using asymmetric Type-3 pairings.

3.2. Access structure

Definition 2 (access structure [19]) Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\vee B, C:$ if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in A$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, ie $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

3.3. Linear secret sharing schemes (LSSS)

Definition 3 (linear secret-sharing schemes (LSSS) [19]) A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over Z_p) if

1) The shares for each party form a vector over Z_p.

2) There exists a matrix A with l rows and n columns called the share-generating matrix for Π. For all $i = 1, \ldots, l$, the i^{th} row of A is labeled by a party $\rho_{(i)}$ (ρ is a function from $\{1, \ldots, l\}$ to $\mathcal{P}$).

When considering the column vector $= (s, r_1, r_2, \ldots, r_n)$, where $s \in Z_p$ is the secret to be shared, and $r_2, \ldots, r_n \in Z_p$ are randomly chosen, then Av is the vector of l shares of the secret s according to Π. The share $(Av)_i$ belongs to party $\rho_{(i)}$.

Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, \ldots, l\}$ be defined as $I = \{i: \rho_{(i)} \in S\}$. Then, there exist constants $\{\omega_i \in Z_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$.

Let A_i denotes the i^{th} row of A, then $\sum_{i \in I} \omega_i A_i = (1,0, \ldots,0)$. Furthermore, these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix A. For unauthorized sets, no such constants $\{\omega_i\}$ exist.

4. System architecture and workflow

4.1. System architecture

The system consists of five entities: central authority (CA), cloud server, fog nodes, data owner (DO), data consumer (DC) and emergency situation handlers (ESH), as shown in Figure 1. The following subsections will summarize the major functions of each entity.

1) Central authority

The CA is a fully trusted agency that is designated to generate the master key and public key. It also generates the attribute key associated with the user's attributes and distributes the public key to the DO for data encryption then transmits the attribute key to the DC for decryption. In order to prevent the break-glass key from being abused, the CA is responsible for tracing the break-glass key.

2) Cloud server

The cloud server offers a data storage service for the DO. It is considered to be semi-trusted that it will manage the uploaded data honestly but curious about the data. It may read the content and compromise user privacy. In the proposed scheme, the cloud is also in charge of helping the DO to update the access policy and re-encrypt the ciphertext.

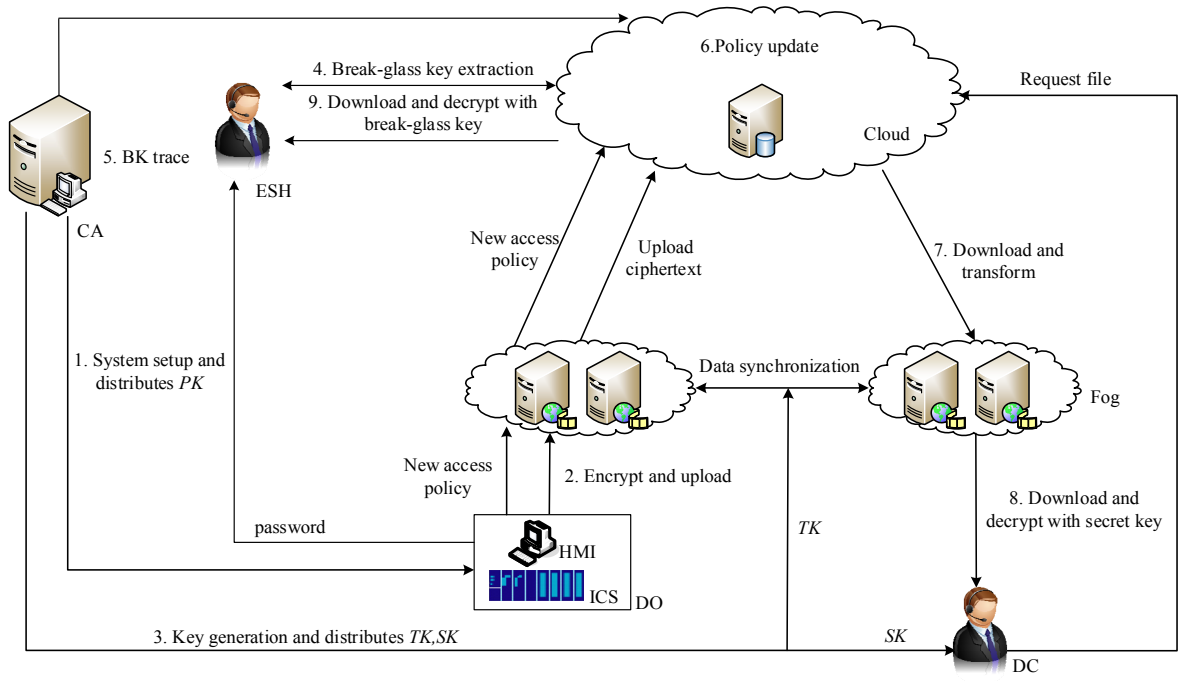


Figure 1. Access control system architecture.

3) Fog nodes

The fog nodes are also semi-trusted, and are responsible for keeping the DC's transform key. Using the transform key, the fog partially decrypt the ciphertext requested by the DC from the cloud, thus reducing the DC's decryption cost.

4) Data owner

The DO refers to the ICS or its data owner, which is used for industrial process control and have a great amount of data to be shared. Normally, the DO encrypts its data using the CP-ABE to realize the confidentiality and flexible access control. In response to some emergency case, the DO pre-shares a password with the ESH, and the ESH can utilize the password to recover the break-glass key to bypass the access policy and decrypt the encrypted data.

5) Data consumer

The DC refers to the party who is attached to the fog and would like to gain access to the ciphertext stored in cloud. A data consumer needs to contact the CA to obtain the attribute key corresponding to the attributes he claims to have. Once the DC's attributes satisfy the access policy, he can obtain a partially decrypted ciphertext from the fog and decrypt the encrypted data using his secret keys.

6) Emergency situation handlers

The ESH refers to the rescuers, fire fighters, polices or other disaster specialists. Once an emergency occurs, the ESH should start the disaster recovery plan and access the original data freely for safety and legal reasons. Using his identity and the password from the DO, the ESH interacts with

the cloud to obtain the break-glass key. Then the ESH can access the original data freely under a controlled environment, because the break-glass key is traceable.

4.2. System workflow

This subsection introduces the system workflow of attribute-based access control, break-glass access control and the access policy updating process.

In the ABAC, the CA runs the system setup algorithm and outputs the public key, then the DO encrypts its data and uploads the ciphertext to the cloud. It first encrypts the data with a random key K_a by applying a symmetric encryption algorithm (e.g., AES), then encrypts the key K_a by applying the CP-ABE algorithm under an access policy. The DC obtains his secret key *SK* and transform key *TK* associated with his attributes from the CA and stores the *TK* on the fog. When the DC requests the ciphertext, the cloud sends it to the fog for ciphertext transformation. Using the *TK*, the fog partially decrypts the ciphertext to reduce the decryption cost on the DC. Last the DC receives the partially decrypted ciphertext from the fog and recovers the plaintext by his secret key *SK*.

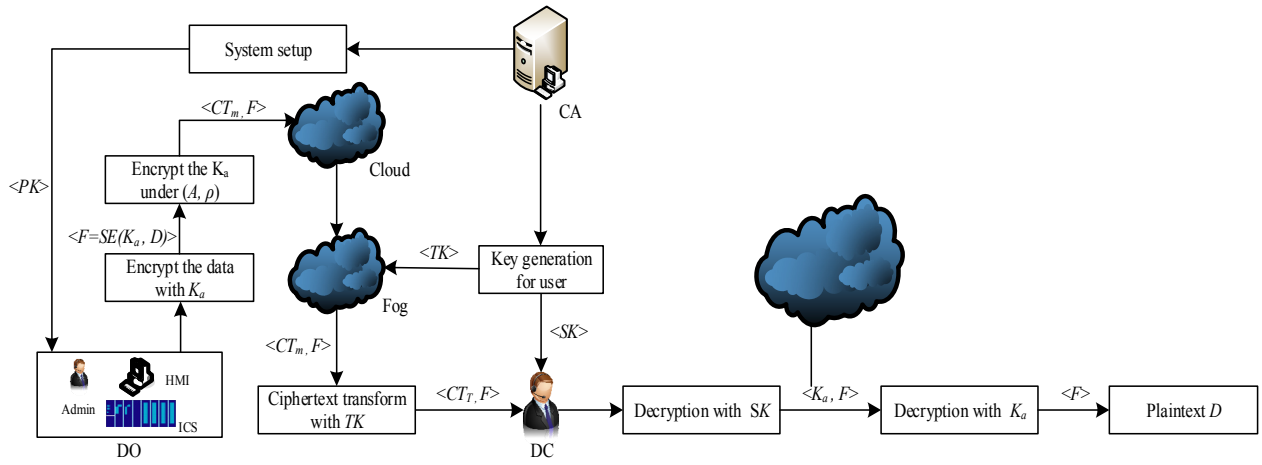


Figure 2. Attribute-based access control workflow.

In the Break-Glass Access Control, the DO pre-shares a password pw with the ESH. Then the DO calculates the ciphertext of the password and uploads it to the cloud. When an emergency occurs, the ESH and the cloud engage in a two party computation to extract the break-glass key BK. The ESH first calculates his identity elements and sends them to the cloud. Using the identity elements, the cloud computes a ciphertext BK' for the ESH. Then the ESH can compute the break-glass key BK by the BK'. Using the break-glass key BK, the ESH recovers the plaintext.

The ciphertext of the proposed scheme consists of two parts, one is the ciphertext of the symmetric key K_a, and the other is the ciphertext of the password pw, which are denoted as CT_m and CT_{pw} respectively. To update the access policy, the DO should update the CT_m and CT_{pw}. Firstly, the DO selects the new vector A'_i corresponding to the i-th row of the $l' \times n'$ matrix A' and sends it to the cloud to update the ciphertext of the key K_a. Then, he generates and transmits the elements of the password pw to the cloud to update the ciphertext of the password. Finally, the cloud runs the policy updating algorithm to generate the new ciphertext CT'_m and CT'_{pw}.

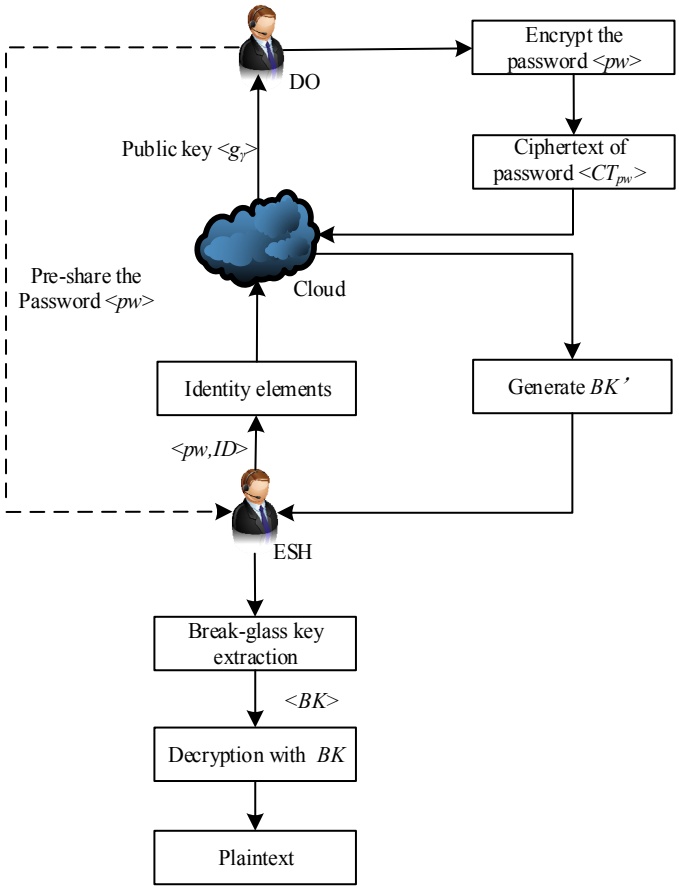


Figure 3. Break-glass access control workflow.

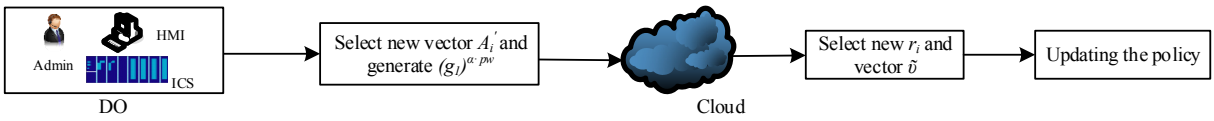


Figure 4. Updating the access policy embedded in the ciphertext.

5. Algorithm descriptions

To achieve the efficient fine-grained access control with break-glass capability, this paper modifies the algorithm in [22] to outsource the computation overhead and utilize asymmetric Type-3 pairings. This paper also applies the idea of the password-controlled encryption shown in [33] to the proposed scheme to realize the break-glass access control. Next, this paper presents the construction of the CP-ABE with break-glass for the cloud-assisted ICS.

1) System Setup: $Setup(1^{\lambda}) \rightarrow (PK, MK)$

The CA runs the setup algorithm to conduct an initial operation and setup important parameters. Take as input the security parameter λ, the CA selects an asymmetric bilinear map: $e: G_1 \times G_2 \rightarrow G_T$, where G_1, G_2 and G_T are groups of prime order p. Let U be the size of system's universe attribute set. Then, the CA chooses generators $g_1 \in G_1$, $g_2 \in G_2$ and random group elements $h_1, \ldots, h_U \in G_1$,

$g_3 \in G_2$. In addition, it randomly chooses $\alpha, \beta \in Z_p$.

Finally, the CA generates the public key PK and master key MK as

$$PK =< g_1, g_2, g_3, e(g_1, g_2)^{\alpha}, e(g_1, g_3)^{\alpha}, g_1^{\alpha}, g_1^{\beta}, h_1, \ldots, h_U > \tag{1}$$

$$MK =< \alpha, \beta > \tag{2}$$

2) Data Encryption: $Encrypt(PK, K_a, (A, \rho), pw) \rightarrow (CT_m, CT_{pw})$

First, a symmetric key K_a is used to encrypt the data D to be shared, then the DO encrypts the key K_a with the encrypt algorithm. The encrypt algorithm takes as input the public parameter PK, an LSSS access structure (A, ρ), and the key K_a, then the DO selects a random vector $v = (s, v_2, \ldots, v_n) \in Z_p^n$, and for $i = 1, \ldots, l$, it computes $\lambda_i = A_i v$, where A_i is the vector corresponding to the *i*-th row of the $l \times n$ matrix A, and the function ρ associates rows of A to the attributes. Then it creates the ciphertext of the key K_a

$$CT_m =< C, C', C'', C_i, D_i > \; = \; < K_a e(g_1, g_2)^{\alpha s}, g_2^s, g_1^s, g_1^{\beta \lambda_i} h_{\rho(i)}^{-r_i}, g_2^{r_i} > \tag{3}$$

Moreover, the DO generates the auxiliary ciphertext of the password *pw*. He randomly chooses a password $pw \in Z_p$ and calculates the ciphertext of the password *pw*:

$$CT_{pw} =< C_{k1}, C_{k2} > \; = \; < (g_\gamma (g_1^{\alpha})^{pw})^s, e(g_1, g_3)^{-\alpha s} > \tag{4}$$

Here, the $\gamma \in Z_p$ is the private key of the cloud and the $g_\gamma = (g_1^{\alpha})^{-\gamma}$ is its public key.

Last, the DO sends the ciphertext $CT =< CT_m, \; CT_{pw} >$ and the encrypted data $F = SE(K_a, D)$ to the cloud.

3) Key Generation for user: $KeyGen(MK, S) \rightarrow (TK, SK)$

The method takes as input the master key *MK* and a set of attributes *S*, the CA randomly selects $t, z \in Z_p$ and generates the attribute keys $AK =< TK, SK >$ for the attribute set S as

$$TK =< K, L, \forall x \in S \;\; K_x > \; = \; < g_1^{\alpha/z} g_1^{\beta t/z}, g_2^{t/z}, \forall x \in S \;\; h_x^{t/z} > \tag{5}$$

$$SK = \; < z > \tag{6}$$

Here, the TK is the transform key and the *SK* is the user's secret key. The TK can be stored on the fog and the *SK* must be secretly maintained by the user. Since the TK is associated with the user's attributes, the fog can partially decrypt the ciphertext for the user.

4) Break-Glass Key Extraction: $BKExt(pw) \rightarrow (BK)$

Considering the emergency situations, the DO pre-shares a password pw with the ESH with an $ID \in Z_p$. When the ESH wants to decrypt the ciphertext in emergency, he interacts with the cloud to obtain the break-glass key BK using the password pw.

The ESH randomly selects $\tau, \mu \in Z_p$ and calculates $R = g_2^{\tau}$, $\tau'_{pw} = \tau \cdot ID$. Then the identity elements $< R, \tau'_{pw}, \mu, pw >$ are sent to the cloud. Then the cloud computes $\delta = \mu(\gamma - pw)$, $g'_{pw} = (R \cdot g_3^{-\tau'_{pw}})^{1/\delta}$ and sends the $BK' =< \tau'_{pw}, g'_{pw} >$ to the ESH.

Finally, the ESH calculates the break-glass key as

$$BK =< \tau_{pw}, g_{pw} >=< \frac{\tau'_{pw}}{\tau}, (g'_{pw})^{\mu/\tau} > \tag{7}$$

and verifies whether or not $\tau_{pw} = ID$. If the equation holds, the key BK is valid. Otherwise, the cloud behaves dishonestly.

5) Trace: $Trace\ (BK) \rightarrow (ID)$

Given the break-glass key $BK =< \tau_{pw}, g_{pw} >$, the CA runs the algorithm to check the ID of the BK by verifying whether or not

$$e\left(g_\gamma g_1^{-pw}, g_{pw}\right) = e\left(g_1, g_2 g_3^{-\tau_{pw}}\right) \tag{8}$$

If the equation holds, it means that the BK is valid. Then the algorithm outputs the $\tau_{pw} = ID$.

6) Policy Update: $PolicyUp\ (PK, CT, (A', \rho')) \rightarrow (CT')$

Take as input the public key *PK*, the ciphertext *CT* and a new access policy (A', ρ'), the cloud runs the policy update algorithm to update the access policy (A, ρ) and generate the new ciphertext CT'.

The DO randomly selects $A'_i = (a_{i1}, a_{i2}, \dots, a_{in'})$, which is the vector corresponding to the *i*-th row of the $l' \times n'$ matrix A'. Then the cloud randomly selects $r'_i \in Z_p$ and a vector $\tilde{v} = (\tilde{s}, \tilde{v}_2, \dots, \tilde{v}_{n'}) \in Z_p^{n'}$. Assume $s' = \tilde{s} + s$, the cloud can output a vector $v' = (s', \tilde{v}_2, \dots, \tilde{v}_{n'})$.

According to the Eq (3), the cloud generates the new ciphertext CT'_m as

$$CT'_m =< \tilde{C}, \tilde{C}', \tilde{C}'', \tilde{C}_i, \widetilde{D}_i >_{i\in[1,l']} \tag{9}$$

It computes the new components as

$$\tilde{C} = C \cdot e(g_1, g_2)^{\alpha\tilde{s}} = K_a e(g_1, g_2)^{\alpha s} \cdot e(g_1, g_2)^{\alpha\tilde{s}} = K_a e(g_1, g_2)^{\alpha s'} \tag{10}$$

$$\tilde{C}' = C' \cdot g_2^{\tilde{s}} = g_2^{s} \cdot g_2^{\tilde{s}} = g_2^{s'} \tag{11}$$

$$\tilde{C}'' = C'' \cdot g_1^{\tilde{s}} = g_1^{s} \cdot g_1^{\tilde{s}} = g_1^{s'} \tag{12}$$

$$\begin{aligned}\tilde{C}_i &= g_1^{\beta A'_i v'} h_{\rho'_{(i)}}^{-r'_i} = g_1^{\beta(a_{i1}s' + a_{i2}\tilde{v}_2 + \cdots + a_{in'}\tilde{v}_{n'})} h_{\rho'_{(i)}}^{-r'_i} \\ &= g_1^{s'\beta a_{i1}} g_1^{\beta(a_{i2}\tilde{v}_2 + \cdots + a_{in'}\tilde{v}_{n'})} h_{\rho'_{(i)}}^{-r'_i} \\ &= \tilde{C}'' g_1^{\beta(a_{i2}\tilde{v}_2 + \cdots + a_{in'}\tilde{v}_{n'})} h_{\rho'_{(i)}}^{-r'_i}\end{aligned} \tag{13}$$

$$\widetilde{D}_i = g_2^{r'_i} \tag{14}$$

According to the Eq (4), the cloud also generates the new ciphertext CT'_{pw}

$$CT'_{pw} =< \tilde{C}_{k1}, \tilde{C}_{k2} > \tag{15}$$

It receives the component $(g_1^{\alpha})^{pw}$ from the DO and computes the new components as

$$\tilde{C}_{k1} = C_{k1} \cdot (g_\gamma (g_1^\alpha)^{pw})^{\tilde{s}} = (g_\gamma (g_1^\alpha)^{pw})^{s} (g_\gamma (g_1^\alpha)^{pw})^{\tilde{s}} = (g_\gamma (g_1^\alpha)^{pw})^{s'} \tag{16}$$

$$\tilde{C}_{k2} = C_{k2} \cdot e(g_1, g_3)^{-\alpha \tilde{s}} = \ e(g_1, g_3)^{-\alpha s} \ e(g_1, g_3)^{-\alpha \tilde{s}} = \ e(g_1, g_3)^{-\alpha s'} \tag{17}$$

Although the cloud doesn't know the secret s, it can also compute the ciphertext CT'_m and CT'_{pw}.

7) Ciphertext Transform: $Transform(TK, CT_m) \rightarrow (CT_T)$

With the input DC's transform key TK and the requested ciphertext CT_m, the fog runs the transform algorithm to generates the CT_T as

$$CT_T = \frac{e(C', K)}{\prod_{i \in I} \left(e(C_i, L) e\left(D_i, K_{\rho_{(i)}}\right)\right)^{\omega_i}} = e(g_1, g_2)^{s\alpha/z} \tag{18}$$

If the DC's attributes match the access policy of the encrypted data, the above equation holds. Otherwise, the algorithm outputs $\perp$, the fog deliveries nothing to the DC.

8) Decryption with Secret Key: $Decrypt_{ABE}(SK, CT_T) \rightarrow (K_a)$

If the DC has a valid set of attributes, he will receive the transformed ciphertext CT_T from the fog. Take as input the secret key *SK* and the ciphertext CT_T, the DC runs the $Decrypt_{ABE}$ algorithm and recover the key K_a as

$$K_a = \frac{C}{(CT_T)^{SK}} = \frac{K_a e(g_1, g_2)^{\alpha s}}{(e(g_1, g_2)^{\frac{s\alpha}{z}})^z} \tag{19}$$

9) Decryption with Break-Glass Key: $Decrypt_{BG}(BK, CT) \rightarrow (K_a)$

In emergency, the ESH executes the $Decrypt_{BG}$ algorithm to recover the key K_a. Take as input the break-glass key BK and the ciphertext CT, the algorithm outputs

$$K_a = C \cdot e\left(C_{k1}, g_{pw}\right) \cdot {C_{k2}}^{\tau_{pw}} \tag{20}$$

$$= K_a e(g_1, g_2)^{\alpha s} \cdot e\left((g_\gamma (g_1^\alpha)^{pw})^s, (g'_{pw})^{\mu/\tau}\right) \cdot \ e(g_1, g_3)^{-\alpha s \tau_{pw}} \tag{21}$$

$$= K_a e(g_1, g_2)^{\alpha s} \cdot e\left(\left((g_1^\alpha)^{-\gamma s} g_1^{\alpha * s * pw}\right), \left(\left(R \cdot g_3^{-\tau'_{pw}}\right)^{\frac{1}{\delta}} \right)^{\frac{\mu}{\tau}} \right) \cdot e(g_1, g_3)^{-\alpha s \tau_{pw}} \tag{22}$$

$$= K_a e(g_1, g_2)^{\alpha s} \cdot e\left(\left(g_1^{pw-\gamma}\right)^{\alpha s}, \left(\left(g_2^\tau \cdot g_3^{-\tau'_{pw}}\right)^{\frac{1}{\mu(\gamma - pw)}} \right)^{\frac{\mu}{\tau}} \right) \cdot e(g_1, g_3)^{-\alpha s \tau_{pw}} \tag{23}$$

$$= K_a e(g_1, g_2)^{\alpha s} \cdot e(g_1, g_2)^{-\alpha s} \cdot e(g_1, g_3)^{\alpha s \tau_{pw}} \cdot e(g_1, g_3)^{-\alpha s \tau_{pw}} \tag{24}$$

6. Analysis and performance

6.1. Security analysis

6.1.1. Confidentiality

In the proposed scheme, the confidentiality of the plaintext data depends on the security of the symmetric key K_a. It would be difficult for the attacker to obtain the K_a, which is hidden in the $K_a e(g_1, g_2)^{\alpha s}$.

There are two ways to obtain the K_a. One is using the key related to the user attributes to decrypt the ciphertext according to the CP-ABE. The other is to utilize the password pw pre-shared by the data owner to break the glass.

Scott [16] has outlined the definition of Waters CP-ABE [19] in the asymmetric Type-3 pairings, and this paper improves it by outsourcing the decryption and re-encryption caused by the policy updating. Since its structure has not been changed, the modified version is secure. Normally, in order to obtain the K_a, the user has to recover the secret s, which is hidden using the LSSS matrix for the access structure. Only if the user attributes satisfy the access policy embedded in the ciphertext, the fog can correctly transform the ciphertext by using the transform key TK which is associated with the user attributes. Otherwise, the transformation will be failed and the fog will deliver nothing to the user. In the transform process, the fog cannot obtain the plaintext. Therefore, this paper realizes the attribute-based access control and protects the confidentiality of the shared data.

When emergency occurs, the ESH utilizes the password pw pre-shared by the DO to break the glass and access the plaintext data. The underlying scheme adopts Zhang's mechanism [33] and has been proved to be *IND-ID-CPA* secure. In the secure two party computation, the ESH and the semi-trusted cloud jointly compute the break-glass key BK. In this progress, the cloud cannot compute the key BK by BK' because the τ is randomly selected by the ESH and hidden in the g_2^{τ}. Finding the τ from g_2^{τ} is hard due to the discrete logarithm problem.

Therefore, the proposed solution protects the data confidentiality.

6.1.2. Collusion attack resistance

In order to recover the key K_a from $K_a e(g_1, g_2)^{\alpha s}$, the malicious users need to get the secret s, which is stored on the DO privately. In the CP-ABE scheme, the DO utilizes the LSSS to share the secret s. Only if the user's attributes satisfy the access policy, his transform key TK can be combined with the ciphertext components C_i, D_i to complete the bilinear pairing operation and recover the secret s. In the key TK, its components K, L and K_x are all randomized by the parameter t, so each user has a different transform key TK. Therefore, users cannot collude with others to combine an available key TK with their key components K, L and K_x.

6.2. Performance analysis

This subsection analyses and compares the performance of the proposed solution with existing schemes. Since this paper rephrases the Lai's scheme [22] in terms of the asymmetric Type-3 pairings and supports break-glass access function, the analysis in this subsection compares the proposed scheme

with Lai's scheme [22] and the ABAC Schemes with break-glass access[32,34].

6.2.1. Capability analysis

A detailed comparison of CP-ABE is given in Table 1. The proposed scheme is defined in the asymmetric Type-3 pairings setting and supports the access policy updating, therefore a more efficient and fine-grained user access control can be achieved. To maintain the efficiency of data processing on resource-constrained devices, the policy updating and most of the decryption are outsourced to the cloud and fog. This paper also integrates the password-based break-glass encryption with the CP-ABE to enable timely access when emergency occurs, and the break-glass key is traceable.

Table 1. A comparative summary of related schemes.

The Schemes	Asymmetric Pairings	Updating Access Policy	Outsourcing Decryption	Break-Glass Access Control	Break-Glass Key Traceability
Lai [22]	No	Yes	No	No	No
Oliveira [32]	No	No	No	Yes	No
Yang [34]	No	No	Yes	Yes	No
The proposed scheme	Yes	Yes	Yes	Yes	Yes

6.2.2. Storage overhead on the data consumer

Table 2 lists schemes to be compared with the proposed algorithm in terms of storage overhead. In these schemes, the storage overhead of the data consumer mainly comes from the SK, BK, CT_m and CT_{pw}. Then, the comparison from the theoretical aspect is given in Table 2. Let $|G_1|$, $|G_2|$, $|G_T|$ and $|Z_p|$ denote the length of each element in G_1, G_2, G_T and Z_p respectively. Let $|S|$ denote the number of attributes in user's key, and $|l|$ denote the number of attributes in an access structure. Because the scheme in [32] lacks specific implementation algorithm, this paper applies the algorithm in the scheme [22] to the scheme [32].

In terms of the storage overhead of the DC, the proposed scheme has obvious advantages. Especially when the access structure is complex, the size of the ciphertext stored on the DC is kept small and constant in the proposed scheme. Moreover, other schemes are limited to the symmetric pairings, whereas the proposed method is described in the asymmetric Type-3 pairings. The proposed scheme will be more efficient in the actual implementation and the results will be reflected in subsection 6.2.4.

6.2.3. Computation cost

Let $|I|$ for number of user's attributes utilized for satisfying the policy. Let exp_1, exp_2, exp_T denote exponentiation in G_1, G_2, G_T and e denote bilinear pairing. This paper compares the computation cost of the proposed scheme with the schemes shown in Table 3.

The proposed scheme only has a higher computation overhead in the file access with break-glass key on the ESH. It is because this paper decomposes the break-glass key into two parts and design the trace algorithm. Utilizing the break-glass key, the trace algorithm can find out the ESH's *ID* in case that the ESH abuses the key.

Table 2. Comparison of storage overhead on the DC.

The Schemes	SK	BK	CT_T	CT_{pw}
Lai [22]	$(\|S\|+2)\|G_1\|$	*None*	$(2\|l\|+1)\|G_1\|+\|G_T\|$	*None*
Oliveira [32]	$(\|S\|+2)\|G_1\|$	$(\|S\|+2)\|G_1\|$	$(2\|l\|+1)\|G_1\|+\|G_T\|+2\|Z_p\|$	*None*
Yang [34]	$(\|S\|+1)\|G_1\|+3\|Z_p\|$	$\|G_1\|$	$\|G_T\|$	$4\|G_1\|+2\|Z_p\|$
The proposed scheme	$\|Z_p\|$	$\|G_2\|+\|Z_p\|$	$\|G_T\|$	$\|G_1\|+\|G_T\|$

Table 3. Comparison of computation cost.

The Schemes	Decryption with Secret Key on the DC	Generate the Ciphertext of *pw* on the DO	BK Extraction on the ESH	Decryption with Break-Glass Key on the ESH	Break-Glass Key Tracing
Lai [22]	$2\|I\|exp_T$ $+(2\|I\|+1)e$	*None*	*None*	*None*	*None*
Oliveira [32]	$2\|I\|exp_T$ $+(2\|I\|+1)e$	*None*	*None*	$2\|I\|exp_T$ $+(2\|I\|+1)e$	*None*
Yang [34]	exp_T	$5exp_1+H$	$4exp_1+H$	H	*None*
The proposed scheme	exp_T	$2exp_1+exp_2$	$2exp_2$	exp_T+e	exp_T+e

6.2.4. Quantitative evaluation

This subsection focuses on the quantitative performance of the proposed scheme based on the Java Pairing-Based Cryptography Library (JPBC) [34] libraries. It is a Port of the Pairing-Based Cryptography Library (PBC), and the library is developed to perform the mathematical operations underlying pairing-based cryptosystems directly in Java. All the implementations are executed on an Intel® Core ® CPU i5-3470 2.6 GHz with 8.00 GB of RAM running the Linux system and Eclipse. This paper evaluates the storage overhead and the encryption/decryption time of the related schemes [32,34] under different security levels (80 bit, 112 bit, 128 bit, 192 bit, 256 bit). The PBC library contains a series of elliptic curves to generate the bilinear pairs. The embedding degree of the Type-A curve is 2, which is often used to realize the symmetric bilinear pairings, but the security level is low. The embedding degree of the Type-F curve is 12, which is often used to realize the asymmetric bilinear pairings. Since the schemes in [32,34] are constructed based on the symmetric bilinear pairings, this paper will adopt the Type-A curve; while the proposed scheme is constructed based on the asymmetric bilinear pairings, so this paper will adopt the Type-F curve. In this subsection, these two kinds of curves are used to construct the two bilinear pairs $e: G_1 \times G_1 \to G_T$ and $e: G_1 \times G_2 \to G_T$, respectively. According to the recommended parameters in the literature [26], which is also recommended by the National Institute for Standards and Technology (NIST) and the European Union (ENISIA agency), we evaluate the relevant schemes. The parameters of each curve are set and shown in Table 4.

Due to the lack of specific implementation algorithm in the scheme [32], this paper applies the algorithm in the scheme [22] to the scheme [32]. Let $|S| = |l| = 10$. This paper first evaluate the storage overhead in the proposed scheme and the schemes [32,34] at different security levels.

Figure 5 (a) describes the ciphertext stored on the DC at different security levels. In the scheme [32], the ciphertext size on the DC is 2856, 5688, 8512, 21260 and 42368 bytes in different security levels.

In the scheme [34], the ciphertext size is 128, 256, 384, 962 and 1920 bytes in different security levels. In the proposed scheme, the ciphertext size is 240, 336, 384, 960 and 1920 bytes in different security levels.

Table 4. The size of the key and groups under the different security level.

Security Level	AES Key	Type-A $G_1 = G_2$	Type-A G_T	Type-A Z_p	Type-F G_1	Type-F G_2	Type-F G_T	Type-F Z_p
80 bit	128	1024	1024	160	320	640	1024	160
112 bit	128	2048	2048	224	448	896	2048	224
128 bit	128	3072	3072	256	512	1024	3072	256
192 bit	192	7696	7696	384	1280	2560	7696	640
256 bit	256	15360	15360	512	2560	5120	15360	1280

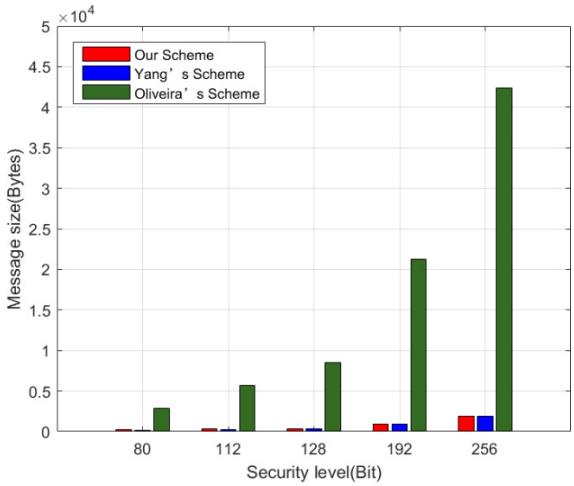
It is obvious that the storage space required by the data consumers in the scheme [32] is the largest. It is because the proposed scheme and the scheme [34] both outsource the decryption to the fog/cloud, the DC only needs to receive the transformed ciphertext, which is small. Although in the table 3, the storage consumptions on the DC of the proposed scheme and the scheme [34] are both $|G_T|$, the proposed scheme requires a higher security level in the real implementation. Because the proposed scheme is the only one implemented in the Type-F curve.

Figure 5(b) describes the secret key stored on the data consumer at different security levels. In the scheme [32], the secret key size on the DC is 1536, 3072, 4608, 11544 and 23040 bytes in different security levels. In the scheme [34], the secret key size is 1468, 2900, 4320 10726 and 21312 bytes in different security levels. In the proposed scheme, the secret key size is 20, 28, 32, 80 and 160 bytes in different security levels. It is obvious that the proposed scheme achieves the best performance. The storage consumption is small and constant, which even can be ignored.

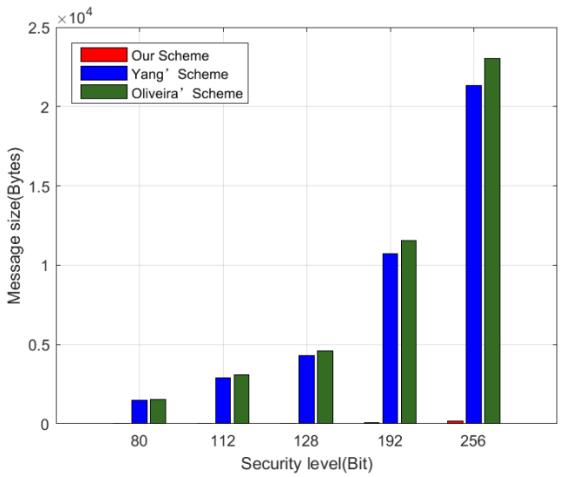
This paper also evaluates the computation cost in the proposed scheme and the scheme [32] at different security levels.

Figure 5(c) describes the encryption time at different security levels. In the scheme [32], the encryption time on the DO is 0.225s, 0.75s, 2.3s, 10.5s, and 45s in different security levels. In the proposed scheme, the encryption time on the DO is 0.09s, 0.15s, 0.22s, 1.5s, and 8.2s in different security levels. Since the proposed scheme adopts the Type-F curve, it reduces the encryption cost. When the security level reaches 128-bit, the encryption cost of the scheme [32] begins to increase significantly. And the higher the security level, the more computation cost. When the security level reaches 256-bit, the encryption time of the scheme [32] is 5 times as much as that of the proposed scheme.

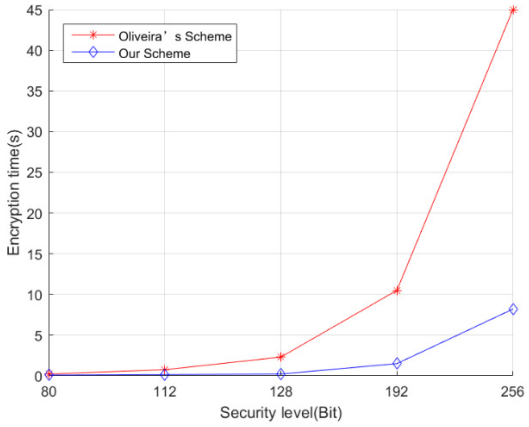
Figure 5(d) describes the decryption time at different security levels. In the scheme [32], the decryption time on the DC is 0.08s, 0.28s, 0.5s, 4s, and 16s in different security levels. Since the proposed scheme outsources the most of the decryption to the fog, the decryption cost of the DC is small and constant, even less than 0.005s. Moreover, this paper compares the decryption time in the scheme [32] with the ciphertext outsourced decryption (ciphertext transform on the fog) in the proposed scheme. The decryption time on the DC of the proposed scheme is 0.18s, 0.28s, 0.32s, 2s, and 10s. It can be seen that the decryption time on the DC of the scheme [32] is lower than that of the proposed scheme at the security level 80-bit and 112-bit only. However, at the security level 128-bit, 192-bit and 256-bit, the advantage of the Type-F curve selected in the proposed scheme has been obviously reflected. The proposed scheme achieves the lower decryption time than the scheme [32].



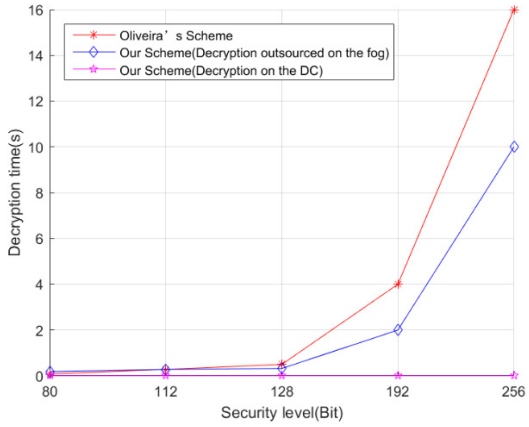
(a) The ciphertext stored on the DC at different security levels.



(b) The secret key stored on the DC at different security levels.



(c) The encryption time at different security levels.



(d) The decryption time at different security levels.

Figure 5. The performance of the proposed scheme.

7. Conclusions

This paper proposes an efficient access control scheme for the cloud-assisted ICS, which allows the attribute-based access control in normal state and the break-glass access control in emergency. The proposed scheme securely outsources the decryption and access policy updating to the fog and cloud, thus achieves small and constant local storage on the user, and especially suits the resource-constrained environment. Furthermore, this paper implements the CP-ABE scheme using the asymmetric Type-3 pairings, which are more efficient for an implementation than the symmetric Type-1 pairings. Finally, this paper analyses the security properties, compares it with other schemes and presents the qualitative and quantitative analysis.

Acknowledgments

This work was supported by the Nanjing Tech University [grant number 39809110], the National Natural Science Foundation of China [grant numbers 61572263, 61272084], The Natural Science Foundation of the Jiangsu Higher Education Institutions of China [grant number 11KJA520002].

Conflict of interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

1. A. Sajid, H. Abbas, K. Saleem, Cloud-assisted Iot-based SCADA systems security: a review of the state of the art and furture challenges, *IEEE Access,* **4** (2016), 1375–1384.
2. T. Ma, H. Rong, Y. Hao, J. Cao, Y. Tian, M. A. Al-Rodhaan, A novel sentiment polarity detection framework for Chinese, *IEEE Trans. Affective Comput.,* (2019), forthcoming.
3. H. Rong, T. Ma, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Deep rolling: a novel emotion prediction model for a multi-participant communication context, *Inf. Sci.,* **488** (2019), 158–180.
4. A. Ouaddaha, H. Mousannif, A. A. Elkalam, A. A. Ouahman, Access control in the Internet of Things: big challenges and new opportunities, *Comput. Networks,* **112** (2017), 237–262.
5. S. Plaga, N. Wiedermann, S. D. Anton, S. Tatschner, H. Schotten, T. Newe, Securing future decentralised industrial IoT infrastructures: challenges and free open source solutions, *Future Gener. Comput. Syst.,* **93** (2019), 596–608.
6. B. Al-Otibi, N. Al-Nabhan, Y. Tian, Privacy-preserving vehicular rogue node detection scheme for fog computing, *Sensors,* **19** (2019), 965.
7. Y. Tian, M. M. Kaleemullah, M. A. Rodhaan, B. Song, A. Al-Dhelaan, T. Ma, A privacy preserving location service for Cloud-of-Things system, *J. Parallel Distrib. Comput.,* **123** (2019), 215–222.
8. B. Song, M. M. Hassan, A. Alamri, A. Alelaiwi, Y. Tian, M. Pathan, A. Almogren, A two-stage approach for task and resource management in multimedia cloud environment, *Computing,* **98** (2016), 119–145.
9. A. Shahzad, S. Musa, A. Aborujilah, M. Irfan, Industrial Control Systems (ICSs) vulnerabilities analysis and SCADA security enhancement using testbed encryption, in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, ACM, 2014.
10. A. Rahman, E. Hassanain, M. Hossain, Towards a secure mobile edge computing framework for Hajj, *IEEE Access,* **5** (2017), 11768–11781.
11. W. Teng, G. Yang, Y. Xiang, T. Zhang, D. Wang, Attribute-based access control with constant-size ciphertext in cloud computing, *IEEE Trans. Cloud Comput.,* **5** (2017), 617–627.
12. V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ACM, (2006), 89–98.

13. T. Kim, R. Barbulescu, Extended tower number field sieve: a new complexity for the medium prime case, in *Proceedings of the 36th Annual International Cryptology Conference (CRYPTO 2016)*, Springer, 2016.
14. S. D. Galbraith, K. G. Paterson, N. P. Smart, Pairings for cryptographers, *Discrete Appl. Math.,* **156** (2008), 3113–3121.
15. A. D. Brucker, H. Petritsch, Extending access control models with break-glass, in *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, ACM, (2009), 197–206.
16. M. Scott, On the efficient implementation of pairing-based protocols, in *Proceedings of the 13th IMA International Conference on Cryptography and Coding*, Springer, (2011), 296–308.
17. A. Sahai, B. Waters, Fuzzy identity based encryption, in *Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques*, Springer, 2005, 457–473.
18. J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in *2007 IEEE Symposium on Security and Privacy*, IEEE, (2007), 321–334.
19. B. Waters, Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization, in *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography*, Springer, (2011), 53–70.
20. Y. Rouselakis, B. Waters, Practical constructions and new proof methods for large universe attribute-based encryption, in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ACM, (2013), 463–474.
21. A. Sahai, H. Seyalioglu, B. Waters, Dynamic credentials and ciphertext delegation for attribute-based encryption, in *Proceedings of the 32nd Annual Cryptology Conference (CRYPTO 2012)*, Springer, (2012), 199–217.
22. J. Lai, R. H. Deng, Y. Yang, J. Weng, Adaptable ciphertext-policy attribute-based encryption, in *International Conference on Pairing-Based Cryptography*, Springer, Cham, (2013), 199–214
23. K. Yang, X. Jia, K. Ren, R. Xie, L. Huang, Enabling efficient access control with dynamic policy updating for big data in the cloud, in *IEEE Annual Joint Conference: INFOCOM, IEEE Computer and Communications Societies*, IEEE, (2014), 2013–2021.
24. M. Green, S. Hohenberger, B. Waters, Outsourcing the decryption of ABE ciphertexts, in *Proceedings of the 20th USENIX Conference on Security*, ACM, (2011).
25. Y. Tu, G. Yang, J. Wang, Q. Su, A secure, efficient and verifiable multimedia data sharing scheme in fog networking system, *Cluster Comput.,* **24** (2020), 225–247.
26. M. Morales-Sandoval, J. L. Gonzalez-Compean, A. Diaz-Perez, V. J. Sosa-Sosa, A pairing-based cryptographic approach for data security in the cloud, *Int. J. Inf. Sec.,* **17** (2018), 441–461.
27. A. Lewko, B. Waters, New proof methods for attribute-based encryption: achieving full security through selective techniques, in *Proceedings of the 32nd Annual Cryptology Conference (CRYPTO 2012)*, Springer, (2012), 180–198.
28. A. Scafuro, Break-glass encryption, in *Proceedings of the 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC2019)*, Springer, (2019), 34–62.
29. A. D. Brucker, H. Petritsch, S. G. Weber, Attribute-based encryption with break-glass, in *Proceedings of the 4th IFIP International Workshop on Information Security Theory and Practices*, Springer, (2010), 237–244.

30. S. Schefer-Wenzl, M Strembeck, Generic support for RBAC break-glass policies in process-aware information systems, in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ACM, (2013), 1441–1446.
31. V. Aski, V. S. Dhaka, A. Parashar, An attribute-based break-glass access control framework for medical emergencies, in *Innovations in Computational Intelligence and Computer Vision,* Springer, (2021), 587–595.
32. M. T. de Oliveira, A. Bakas, E. Frimpong, A. E. D. Groot, H. A. Marquering, A. Michalas, et al., A break-glass protocol based on ciphertext-policy attribute-based encryption to access medical records in the cloud, *Ann. Telecommun.,* **75** (2020), 103–119.
33. T. Zhang, S. S. M. Chow, J. Sun, Password-controlled encryption with accountable break-glass access*,* in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ACM, (2016), 235–246.
34. Y. Yang, X. Liu, R. H. Deng, Lightweight break-glass access control system for healthcare Internet-of-Things, *IEEE Trans. Ind. Inf.,* **14** (2018), 3610–3617.

AIMS Press