Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.062324

ARTICLE

Blockchain-Based Framework for Secure Sharing of Cross-Border Trade Data

Shenjian Xiao[1], Xiaoli Qin[1], Yanzhao Tian[1,*] and Zhongkai Dang[2]

[1]School of Cyberspace Security, Hainan University, Haikou, 570208, China
[2]National Computer Network Emergency Response Technical Team, Coordination Center of China, Beijing, 100029, China
*Corresponding Author: Yanzhao Tian. Email: tiany2048@hainanu.edu.cn

Received: 16 December 2024; Accepted: 28 January 2025; Published: 16 April 2025

ABSTRACT: The advent of the digital age has consistently provided impetus for facilitating global trade, as evidenced by the numerous customs clearance documents and participants involved in the international trade process, including enterprises, agents, and government departments. However, the urgent issue that requires immediate attention is how to achieve secure and efficient cross-border data sharing among these government departments and enterprises in complex trade processes. In addressing this need, this paper proposes a data exchange architecture employing Multi-Authority Attribute-Based Encryption (MA-ABE) in combination with blockchain technology. This scheme supports proxy decryption, attribute revocation, and policy update, while allowing each participating entity to manage their keys autonomously, ensuring system security and enhancing trust among participants. In order to enhance system decentralization, a mechanism has been designed in the architecture where multiple institutions interact with smart contracts and jointly participate in the generation of public parameters. Integration with the multi-party process execution engine Caterpillar has been shown to boost the transparency of cross-border information flow and cooperation between different organizations. The scheme ensures the auditability of data access control information and the visualization of on-chain data sharing. The MA-ABE scheme is statically secure under the q-Decisional Parallel Bilinear Diffie-Hellman Exponent (q-DPBDHE2) assumption in the random oracle model, and can resist ciphertext rollback attacks to achieve true backward and forward security. Theoretical analysis and experimental results demonstrate the appropriateness of the scheme for cross-border data collaboration between different institutions.

KEYWORDS: Multi-authority attribute based encryption; blockchain; data sharing; access control

1 Introduction

With the advent of the digital era, global economic and trade activities are becoming increasingly digitized, and international cross-border trade has entered a new stage of facilitation and efficiency [1]. To effectively reduce trade costs and enhance efficiency, paperless trade has emerged as a new trend [2]. According to estimates by the WTO, the current scale of digital trade has reached approximately $4 trillion, accounting for about half of the global service exports [3]. The entire international trade process typically encompasses at least 25 participants and generates 30–40 trade documents, with over 200 copies produced [3,4]. These documents in the international trade process must be extensively shared with various stakeholders, including importers, exporters, banks, logistics providers, government agencies, and customs [5]. Therefore, the rapid development of digital international trade urgently demands improvements in the secure and efficient sharing of cross-border data, thereby augmenting the utilization of shared data and promoting the globalization of the digital economy [6].

Currently, several challenges impede the effective cross-border sharing of data in international trade, including data redundancy, lack of trust, insufficient collaboration, privacy concerns, interoperability issues, and data source management [7]. In the face of these challenges, developing a technological solution that both strengthens trust mechanisms and ensures information security has become increasingly urgent. As blockchain technology matures, it is gradually emerging as an ideal solution to these issues due to its unique advantageous features. Blockchain not only provides a decentralized structure to reduce reliance on a single entity but also ensures data authenticity and persistence through its immutable ledger. Moreover, the high transparency of blockchain and the automated execution of smart contracts greatly enhance trust among all participants [8]. For example, in cross-border e-commerce, blockchain technology has wide applications in combating counterfeit goods [9], providing immutable records of transactions [10], enabling secure electronic payments [11], safeguarding the privacy of international business users [12], and integrating with the Internet of Things (IoT) [13]. Blockchain also facilitates information sharing through a distributed network, uses smart contracts to prevent data tampering [14], validates trade documents [15], and provides traceability of supply chain processes to increase transaction transparency [16]. Liu et al. introduced a blockchain-based cross-border e-commerce supply chain framework that employs a multi-chain model to store diverse data and combines RSA and Elliptic Curve Cryptography (ECC) for product information encryption and signing, targeting improved product traceability [17]. Zhao proposed a similar method using SHA-256 and ECC for file encryption and authentication via user ID [12], though lacking support for fine-grained data access control. Singapore's TradeTrust project integrates blockchain and digital signature technologies, significantly enhancing the authenticity and traceability of shared documents in international trade [15]. The project uses blockchain to generate a unique digital fingerprint (hash value) for each document, recording it on the blockchain to ensure that any tampering with the original document can be detected, thereby improving document authenticity. However, its limitation lies in its ability to only verify whether a document was sent by a specific signer, without ensuring the privacy of the document itself. Wu et al. constructed a model based on blockchain combined with incentives to improve the regulation and data sharing of cross-border logistics in modular construction (CLMC) [18]. This model strengthens CLMC activity supervision and promotes information flow among participants, addressing issues of unclear responsibility, weak data tracking, and insufficient incentive for information exchange. Rahman et al. designed a cross-border data exchange platform with multi-layer secure gateway features, which requires two Elliptic Curve Digital Signature Algorithm (ECDSA) signing and verification steps during each data interaction [19]. Although the platform adopts a "relaxed trust assumption", meaning it does not fully trust either the data providers or receivers. It still relies on global cloud services as intermediaries, using an audit blockchain at the application layer to validate the secure gateway and enforce penalties. However, this centralized control mechanism may pose a single point of failure risk, and if the global cloud service itself is attacked or behaves maliciously, the security of the entire system will be significantly threatened.

In order to provide a more thorough explanation of the aforementioned scheme, the subsequent example will be employed to illustrate the process of cross-border trade. The exporter and importer establish contract terms and exchange necessary documents (contracts, orders, invoices, packing lists). Exporters prepare some documentation, including commercial invoices, packing lists, origin certificates, and inspection/quarantine certificates, which are submitted to to customs and governmental regulatory authorities for regulatory review. Often, companies struggle to track the status of document reviews, leading to information asymmetry and lack of transparency. Post-audit, the company sends transport documents and insurance policies to the carrier, who delivers the goods to the importing country via international transport. The importer then processes import declarations and collects the goods. This process involves repetitive document transmission.

Facing the complexities of diversified and fine-grained file-sharing among organizations, where single files may require access by multiple entities. During the document-sharing process, only policy-specified authoritative entities participate in document sharing; others may engage in different file exchanges but not in this process. We propose a decentralization MA-ABE scheme that allows each participating entity to freely join and manage the keys. Using IPFS for decentralized storage and integrated with Caterpillar, our solution enforces adherence to predefined processes and offers real-time status transparency, thereby enhancing trust and compliance in customs clearance document exchanges.

The main contributions are summarized as follows:

- We propose a practical MA-ABE scheme that supports proxy decryption, attribute revocation, and policy update. The security proof shows that our MA-ABE scheme is statically secure in the random oracle model. Theoretical analysis and simulation experiments demonstrate that our scheme is highly efficient and practical in terms of proxy decryption and attribute revocation.
- We improve the CTUpdate algorithm for the problem of ciphertext rollback attacks in general attribute revocation mechanisms. In our scheme, the update algorithms must all be operated by attribute privileges, which do not send any update cipher keys to any entity, thus achieving true backward security. While this increases the computational burden of attribute permissions, it strikes a reasonable balance between efficiency and security.
- We propose a Multi-Authority Data Exchange Architecture (MADEXA), which integrates hybrid encryption, MA-ABE, smart contracts, and IPFS. To enhance system decentralization, we implemented a mechanism in the MADEXA scheme where multiple institutions interact with smart contracts and jointly participate in the generation of public parameters.
- We demonstrate the integration of MADEXA with Caterpillar to provide data flow protection for business process management systems. This enables secure sharing of process-oriented data between different departments and enterprises, and enables visualization of the status of data sharing, and real-time monitoring of the clearance status by each participant.

2 Background Knowledge

2.1 Complexity Assumption

For our security proof, we utilize the q-type assumption over prime-order bilinear groups, specifically referring to the q-DPBDHE2 [20], which is a variant of the q-decisional parallel bilinear Diffie-Hellman exponent assumption presented in [21].

Definition 1. *(q-DPBDHE2 Assumption) Select a bilinear group $\mathbb{G}$ of prime order p, associated with a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, based on the security parameter λ. Let $s, a, b_1, b_2, \ldots, b_q$ be chosen uniformly at random from $\mathbb{Z}_p$ and let R be chosen uniformly at random from $\mathbb{G}_T$. Let $D = (\mathbb{G}, p, e, g, g^s, \mathscr{K}, \mathscr{D}, \mathscr{H}, \mathscr{M})$, where*

$$\mathscr{K} = \{g^{s/b_i} \mid i \in [1,q]\} \qquad \mathscr{H} = \{g^{b_j a^i} \mid i \in [1,2q]\backslash\{q+1\}, j \in [1,q]\}$$
$$\mathscr{D} = \{g^{a^i} \mid i \in [1,2q]\backslash\{q+1\}\} \qquad \mathscr{M} = \{g^{sa^i b_j/b_{j'}} \mid i \in [1,q+1], j \in [1,q], j' \in [1,q], j \neq j'\}$$

and this assumption states that no polynomial-time distinguisher can differentiate between the distributions $\left(D, e(g,g)^{sa^{q+1}}\right)$ and (D, R) with non-negligible advantage.

2.2 Business Process Modeling and Notation

Business Process Modeling and Notation (BPMN) and Decision Model and Notation (DMN) are the most widely used process and decision modeling languages [22]. BPMN defines the symbols and semantics

for collaboration diagrams, flowcharts, and processes. It is intended for direct use by stakeholders involved in designing, managing, and implementing business processes, while also being precise enough to allow BPMN diagrams to be converted into software process components. It features easy-to-use, flowchart-like symbols that are independent of any specific implementation environment. In Fig. 1, it shows a BPMN collaboration diagram that illustrates the customs clearance process based on a single window system.

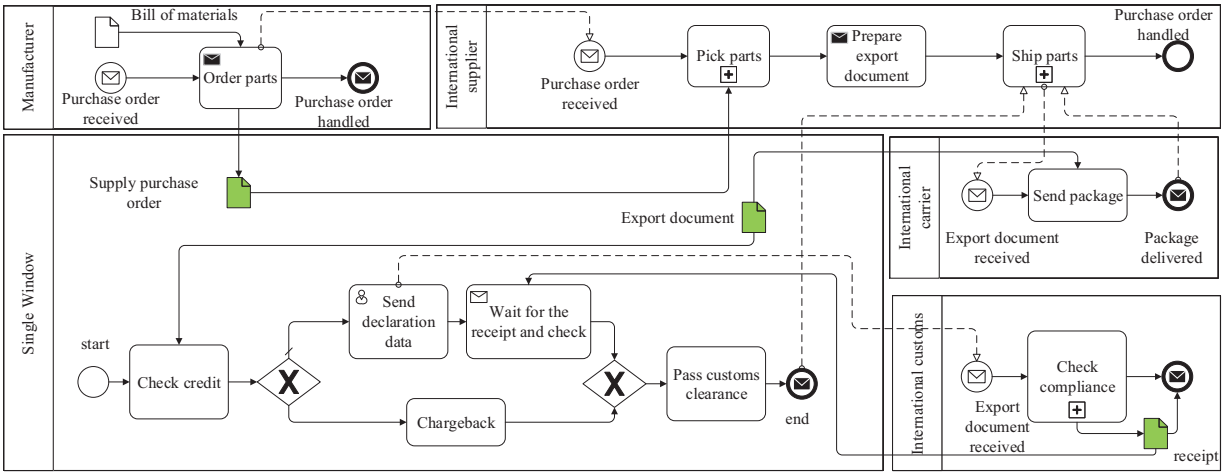

Figure 1: Customs clearance process based on single window

Caterpillar is designed to combine the convenience of Business Process Management System (BPMS) with the security and transparency features of blockchain platforms [23]. Similar to the traditional BPMS, it supports the instantiation of process models and allows users to monitor the status of process instances and perform tasks within them. What sets it apart is the preservation of each process instance's state on the Ethereum, where workflow management is facilitated by smart contracts produced via a BPMN-to-Solidity compiler. It is particularly suited for cross-organizational processes and is often referred to as "process-centric decentralized applications". The applications operate among untrusted participants and require ensuring that all parties adhere to predefined process model rules. Caterpillar aims to ensure design compliance, meaning that no party can execute a transaction that violates the collaborative process model.

3 MA-ABE Scheme

This paper proposes an enhanced Ciphertext-Policy Multi-Authority Attribute-Based Encryption (CP-MA-ABE) scheme that supports proxy decryption, attribute revocation and policy updates. It builds on the MA-ABE framework by Liu et al. and addresses the inadequacy of the model in resisting user collusion attacks [24]. The proposed improvements are designed to optimise proxy decryption and attribute revocation for practical applications, particularly for users with limited resources.The policy update method is consistent with that of Liu et al.'s scheme [24]. The key symbols utilised in the proposed scheme are listed in Table 1. The subsequent sections provide detailed descriptions of the algorithms.

Table 1: The ABE scheme system symbol

Symbol	Description	Symbol	Description
GP	The global public parameters	T	Maps each attribute to a unique attribute authority
$\mathbb{Z}_p^*$	The set $\mathbb{Z}_p - \{0\}$	BT_x	Binary state tree corresponding to attribute x
uid	Unique user identity	$S_{uid,\theta}$	The attribute set owned by uid under θ
U	The attribute universe	AUG_x	The attribute user group corresponding to attribute x
GID	The global identifier universe	ASK_θ/APK_θ	The authority secret key/public key
θ	The attribute authority	GK_x	Attribute group key corresponding to attribute x
$\mathscr{U}_\Theta$	The system authority universe	H/F	Maps each user identities/authority to elements of $\mathbb{G}$.
RK_{uid}	User uid retrieval key	$\alpha_\theta, y_\theta, d_x$	Random numbers belong to $\mathbb{Z}_p^*$
TK_{uid}	User uid transformation key	$UAK_{S,uid,\theta}$	User attribute key under authority θ

$\mathsf{GlobalSetup}(\lambda) \to GP$: This algorithm accepts an implicit security parameter λ as part of its input. It chooses a suitable bilinear group $\mathbb{G}$ of prime order p, and sets g as a generator of $\mathbb{G}$. It defines a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The algorithm outputs $GP = (\mathbb{G}, \mathbb{G}_T, p, e, g, e(g,g), U, \mathscr{U}_\Theta, GID, H, F, T)$.

$\mathsf{AAKeyGen}(GP, \theta) \to (APK_\theta, ASK_\theta)$: The attribute authority (AA) is responsible for executing this algorithm. For $\theta \in \mathscr{U}_\Theta$, it selects $\alpha_\theta, y_\theta \in \mathbb{Z}_p^*$, the final output $ASK_\theta = \{\alpha_\theta, y_\theta\}$ and $APK_\theta = \{e(g,g)^{\alpha_\theta}, g^{y_\theta}\}$.

$\mathsf{UserKeyGen}(GP, ASK_\theta, uid, S_{uid,\theta}) \mapsto UAK_{S,uid,\theta}$: For each attribute x in $S_{uid,\theta}$, the authority maintains AUG_x who possess the attribute x and selects $GK_x, d_x \in \mathbb{Z}_p^*$, then $K_{uid,x} = g^{\alpha_\theta} H(uid)^{y_\theta} F(x)^{d_x}$ and $KP_{uid,x} = g^{d_x GK_x}$ are computed. Finally, The authority sends $UAK_{S,uid,\theta} = \{K_{uid,x}, KP_{uid,x}\}_{x \in S_{uid,\theta}}$ and publishes $F(x)^{1/GK_x}$.

$\mathsf{Encrypt}(m, \mathfrak{A}, GP, \{APK_\theta\}) \to CT$: $\mathfrak{A}$ is the access structure, where $\mathfrak{A} = (\mathbf{A}, \delta)$. $\mathbf{A}$ is an $l \times n$ matrix, and δ maps each row $\mathbf{A}_i$ to an attribute $\delta(i)$. The function ρ associates each row $\mathbf{A}_i$ with the authority that grants the attribute $\delta(i)$, which can be represented as $\rho(i) = T(\delta(i))$. The encryption of the plaintext message m proceeds as follows: Choose random numbers $r_1, \ldots, r_l \in \mathbb{Z}_p^*$ and two random vectors $\mathbf{v} = (s, v_2, \ldots, v_n)^T$ and $\mathbf{z} = (0, z_2, \ldots, z_n)^T$. For $i \in L$, compute $\beta_i = \mathbf{A}_i \mathbf{v}$ and $w_i = \mathbf{A}_i \mathbf{z}$, where $L = \{1, 2, \ldots, l\}$. Then, calculate

$$C_0 = m \cdot e(g,g)^s, \quad C_{1,i} = e(g,g)^{\beta_i} e(g,g)^{r_i \alpha_{\rho(i)}}, \quad C_{2,i} = g^{-r_i}, \quad C_{3,i} = g^{r_i y_{\rho(i)}} g^{w_i},$$
$$C_{4,i} = F(\delta(i))^{r_i / GK_{\delta(i)}}.$$

The ciphertext CT is set to be $CT = (\mathfrak{A}, C_0, \mathfrak{B})$, where $\mathfrak{B} = \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}\}_{i \in L}$.

$\mathsf{Decrypt}(GP, uid, UAK_{S,uid}, CT) \rightarrow (m/\perp)$: Assuming user possesses the S_{uid} and wants to decrypt CT. If $S_{uid} \notin \mathfrak{A}$, the algorithm outputs stops. Otherwise, the algorithm finds the constant set $\{c_i | i \in I\}$ for which $I = \{i | \delta(i) \in S_{uid}\}$ and $\sum_{i \in I} c_i \mathbf{A}_i = (1, 0, \ldots, 0)$. Then compute:

$$\begin{aligned}
&\prod_{i \in I} (C_{1,i} \cdot e(K_{uid,\delta(i)}, C_{2,i}) e(H(uid), C_{3,i}) e(KP_{uid,\delta(i)}, C_{4,i}))^{c_i} \\
&= \prod_{i \in I} (e(g,g)^{\beta_i} e(g,g)^{r_i \alpha_{\rho(i)}} e(g^{\alpha_{\rho(i)}} H(uid)^{y_{\rho(i)}} F(\delta(i))^{d_{\delta(i)}}, g^{-r_i}) e(H(uid), g^{r_i y_{\rho(i)} w_i}) e(g^{d_\delta(i) GK_\delta(i)}, \\
&\quad F(\delta(i))^{r_i / GK_{\delta(i)}}))^{c_i} \\
&= \prod_{i \in I} (e(g,g)^{\beta_i} e(H(uid), g)^{w_i})^{c_i} \\
&= e(g,g)^s
\end{aligned}$$

Finally, the decrypted message is $m = C_0 / e(g,g)^s$.

3.1 Proxy Decryption

For decryption users with limited computational resources, proxy Decryption is an option. This phase comprises three algorithms: GenTR, PartDec, and FinalDec.

$\mathsf{GenTR}\,(uid, UAK_{S,uid}) \rightarrow (TK_{uid}, RK_{uid})$: User uid selects $t \in \mathbb{Z}_p^*$ and calculates $K^1_{uid,x} = (K_{uid,x})^{1/t}, KP^2_{uid,x} = (KP_{uid,x})^{1/t}$, where $x \in S_{uid}$. The algorithm outputs $TK_{uid} = \{K^1_{uid,x}, KP^2_{uid,x}\}_{x \in S_{uid}}$ and $RK_{uid} = t$. Finally, the user sends TK_{uid} to the proxy server and retains RK_{uid}.

$\mathsf{PartDec}(TK_{uid}, CT, GP) \rightarrow CT'/\perp$: The partial decryption algorithm is executed by the proxy server. If $S_{uid} \notin \mathfrak{A}$, the algorithm outputs stops. Otherwise, it calculates

$$CT'_1 = \prod_{i \in I} (C_{1,i} \cdot \mathrm{e}(H(uid), C_{3,i}))^{c_i} \quad \text{and} \quad CT'_2 = \prod_{i \in I} \left(\mathrm{e}(K^1_{uid,\delta(i)}, C_{2,i}) \cdot \mathrm{e}(KP^2_{uid,\delta(i)}, C_{4,i})\right)^{c_i}$$

Finally, it generates the partially decrypted ciphertext $CT' = \{\mathfrak{A}, C_0, CT'_1, CT'_2\}$ to be sent to the user.

$\mathsf{FinalDec}(CT', RK_{uid}) \rightarrow m/\perp$: The final decryption algorithm is run by the user. If $S_{uid} \notin \mathfrak{A}$, this algorithm outputs stops. Otherwise, compute:

$$\frac{C_0}{CT'_1 * {CT'_2}^{RK_{uid}}} = \frac{m e(g,g)^s}{e(g,g)^s} = m$$

3.2 Attribute Revocation

When the attribute authority revokes the user's attribute x, the authority should update the membership of AUG_x and generate a new attribute group key GK'_x. The authority then computes the revocation update key $RUK_x = g^{d_x(GK'_x - GK_x)}$. The authority sends RUK_x to users and generates the update ciphertext key $UCK_x = GK_x / GK'_x$.

$\mathsf{KeyUpdate}(uid, UAK_{S,uid}, x, RUK_x) \rightarrow UAK'_{uid,x}$: The user executes the revocation attribute key update algorithm. x is the revoked attribute. For non-revoked attribute sets, the corresponding attribute keys remain unchanged. Otherwise, the algorithm updates $KP'_{uid,x} = KP_{uid,x} \cdot g^{d_x(GK'_x - GK_x)}$. The revocation attribute key updated by the user is $UAK'_{uid,x} = \{K_{uid,x}, KP'_{uid,x}\}$.

$\mathsf{CTUpdate}(CT, x, UCK_x) \rightarrow CT^*$: This algorithm is executed by AA to update the ciphertext. The algorithm updates $C'_{4,i} = {C_{4,i}}^{GK_{\delta(i)}/GK'_{\delta(i)}}$ Finally, it generates the new re-encryption ciphertext $CT^* = (\mathfrak{A}, C_0, \mathfrak{C})$, where $\mathfrak{C} = \{C_{1,i}, C_{2,i}, C_{3,i}, C'_{4,i}\}_{i \in L}$.

3.3 Security Model

In this section, we extend the security model of RW15 by defining a security game between the challenger $\mathscr{C}$ and adversary $\mathscr{A}$ [20]. The game requires that all key-query requests from $\mathscr{A}$ are instantly forwarded to $\mathscr{C}$ after observing the global public parameters. Additionally, $\mathscr{A}$ is allowed to corrupt (or manipulate) a fixed number of attribute authorities, with these corrupted entities remaining constant throughout the duration of the game.

The security attribute revocation in ABE should have two security requirements: forward security and backward security. Forward security means that if one or more attributes are revoked and a user's remaining attributes no longer satisfy the access policy, that user will not be able to access data that was previously accessible using the revoked attributes. Backward security means that if one or more attributes are revoked, then the user will not be able to access subsequent data using the revoked attributes.

Ciphertext rollback attack refers to a scenario where, when updating a ciphertext, a malicious user obtains ciphertext update items from the relevant AA. For example, attribute revocation or policy update. The user rolls back the updated ciphertext to a previous version by using the ciphertext update item. Finally, the malicious user can decrypt the rolled-back ciphertext using the private key of the older version.

The security game proceeds as follows:

GlobalSetup: $\mathscr{C}$ execute GlobalSetup algorithm obtains the GP and then forwards them to $\mathscr{A}$.

Adversary's Queries: $\mathscr{A}$ establishes a set of corrupt permissions $\mathscr{U}_C \subset \mathscr{U}_\Theta$ and the remaining uncorrupted permissions $\mathscr{U}_N \subset (\mathscr{U}_\Theta - \mathscr{U}_C)$. Our query and reply is similar to RW15 regarding Authority Public Keys and Secret Keys. $\mathscr{A}$ statically issues the following queries:

- Transform Key query: $\mathscr{A}$ requests a key transformation by submitting a sequence of (uid, S_{uid}) tuples to $\mathscr{C}$, also under the condition that $T(S_{uid}) \cap \mathscr{U}_C = \emptyset$. Note that it is meaningless to query the transform key for a user uid who has already had their secret key queried.
- Encryption query: $\mathscr{A}$ submits a challenge access structures $(\mathbf{A}^*, \delta^*)$ and two equal-length messages m_0, m_1 to $\mathscr{C}$. The requirement is that these challenge access structures cannot be met through attributes $S_{uid,\theta}$ involved in previous secret key query operations, nor through attribute combinations controlled by the corrupt authorities.
- KeyUupdate query: $\mathscr{A}$ issues a revocation attribute key update query by submitting (uid, S_{uid}) and a revocation attribute x to $\mathscr{C}$.
- UKeyGen query: $\mathscr{A}$ issue a update the ciphertext key query by submitting two equal-length challenge messages m_0, m_1 and two access policies $(\mathbf{A}_i^*, \delta_i^*), (\mathbf{A}_j^*, \delta_j^*)$, where $i, j \in \{1, 2, \ldots, q\}$ and $i \neq j$.

Challenger's Replies: Upon receiving a query, the challenger $\mathscr{C}$ selects a random bit $b \in \{0, 1\}$ and responds accordingly.

- Transform Keys : $\mathscr{C}$ runs the GenTR algorithm to get the transformed key set TK_{uid} and send them to $\mathscr{A}$.
- Encryption : $\mathscr{C}$ run $\mathsf{Encrypt}(m_b, (\mathbf{A}^*, \delta^*), GP, \{APK_\theta\}) \to CT$, where $\{APK_\theta\}$ is the set of all authority public keys (corrupt and non corrupt).
- KeyUpdate: For (uid, S_{uid}, x) and $x \in S_{uid}$, $\mathscr{C}$ run KeyUpdate algorithm to get $UAK'_{S,uid}$ and subsequently send to $\mathscr{A}$.
- UKeyGen : $\mathscr{C}$ run $\mathsf{Encrypt}(m_b, (\mathbf{A}_i^*, \delta_i^*), GP, \{APK_\theta\}) \to CT$. Then $\mathscr{C}$ proceeds to execute UKeyGen $(GP, (\mathbf{A}_i^*, \delta_i^*), (\mathbf{A}_j^*, \delta_j^*), \{APK_\theta\}) \to UK_{m_b}$ and $\mathsf{CTUpdate2}(CT, UK_{m_b}) \to \widetilde{CT}$. At last, it sends $\widetilde{CT}$ to $\mathscr{A}$.

. **Guess.** *$\mathscr{A}$ outputs a guess bit $b' \in \{0,1\}$. If $b' = b$, $\mathscr{A}$ win the game.*

. **Definition 2.** *We say that an attacker statically breaks the scheme if it has a non negligible advantage in correctly guessing the bit b in the above security game. The advantage of $\mathscr{A}$ in this game is defined as $Adv_{\mathscr{A}} = |Pr[b' = b] - \frac{1}{2}|$.*

4 Our Framework

Throughout the paper, the scenario in Fig. 1 will be consistently referred to. Table 2 provides detailed information on three documents: i) the manufacturer's purchase order, ii) the international supplier's export documents–comprising the shipping bill and commercial invoice–which can be merged for efficiency, and iii) customs clearance data from national customs. This integrated design facilitates accessibility for multiple stakeholders, enabling all participants to validate the information's overall integrity while ensuring decryption access is restricted to intended recipients only.

4.1 Modeling Framework

As shown in Fig. 2, our architecture's main components consist of the following seven entities:

- Certifier Central: It is responsible for deploying smart contracts, assigning user roles, and generating unique global identifiers for each user and authority within the system.
- Data Owner: Responsible for encrypting information, setting access policies, and maintaining a blockchain account along with an RSA key pair.
- Data User: Participants who desire to obtain information, such as that of the manufacturer, customs, and the supplier. The Data user can choose whether to use proxy decryption or not.
- Authority Network: Each Authority is an independent attribute authority. Each Authority can manage any number of attributes, and each attribute is only associated with an Authority.
- Proxy Server: Responsible for providing robust computational support, enabling ciphertext updates and partial decryption for users.
- Smart Contracts: Utilized for the secure storage of resource locators, these contracts are detailed in subsequent sections.
- Data Store: This IPFS-based storage server generates a unique hash for each file based on its content. Utilizing IPFS's InterPlanetary Naming System (IPNS), it ensures stable and persistent content references by maintaining unaltered access links despite content updates. IPNS is used as the resource locator.

Table 2: Summary information on cross-border data sharing

Message	Metadata		Data	Body (slices)		Recipients
Supply purchase order	sender:	Manufacturer	Companyname: sigma			International supplier
		0×07e. . . A6	Address: 56, Sigma street	Key:	eyts. . . tmn	
	processId:	45896228	E-mail: cyna.sigma@mail.com	Fields:	DStg. . . 6yu	
	messageId:	27156063	Price: $10, 000			
Export document	sender:	Int supplier	Manufacturer-company: Beta	SliceId:	63298445	Manufacturer
		0x510. . . 39	Delivery-address: 10, Beta street	Key:	eJyt. . . YFo	National customs
	processId:	45896228	E-mail: mnfctr.beta@mail.com	Fields:	rXpN. . . 8rg	International customs
	messageId:	89562537	Amountpaid: $10, 000			International carrier
			InvoiceID: 156841			
			Address: 88, Omega street	SliceId:	18695481	Manufacturer
			Total: $10,000	Key:	eJyt. . .S6=	National customs
			CompanyVAT: U789456123	Fields:	0bTX. . . D2h	International customs
			Issuedate: 2023-10-22			
National customs clearance	sender:	Nat customs	Taxpayment: confirmed			
		0x358. . . 23	Conformitycheck: passed			
	processId:	45896228	Date: 2023-10-20	Key:	eJds. . . dju	Manufacturer
	messageId:	21453658	Sender: Beta	Fields:	hRB3. . . flj	International customs
			Receiver: Sigma			

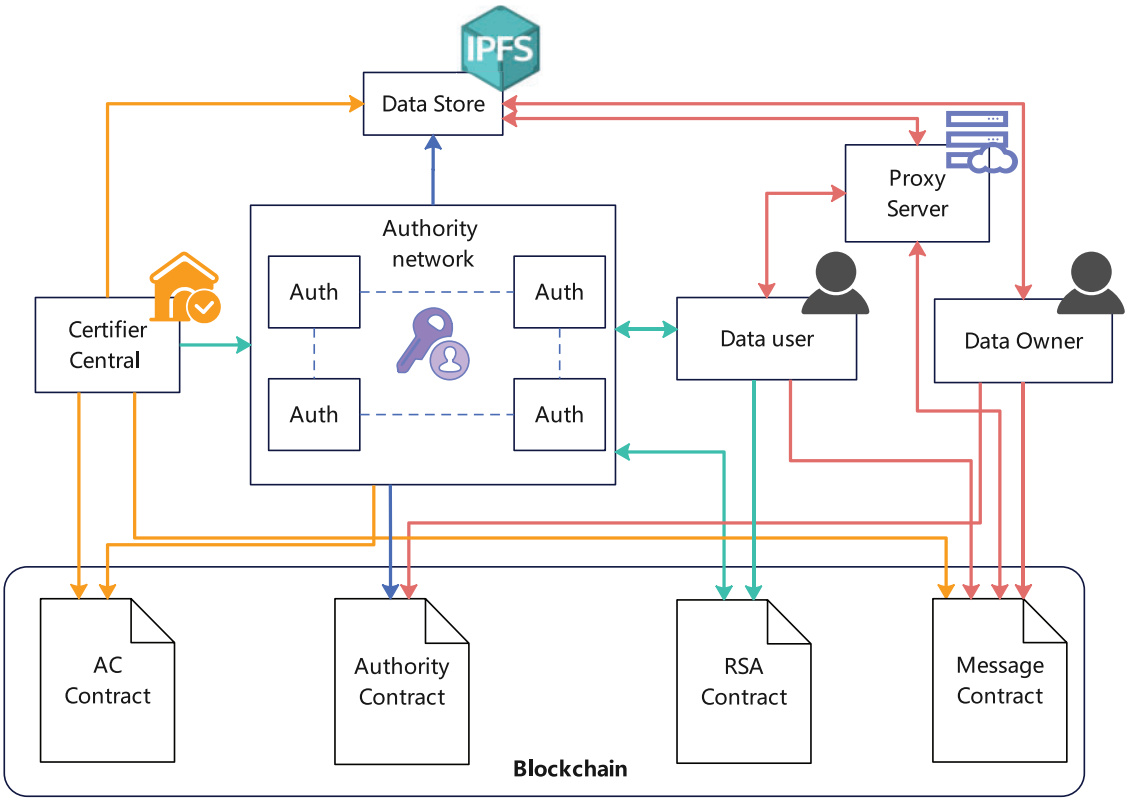

Figure 2: Interaction between the framework and components of the model

4.2 Model Process

In this section, a detailed introduction to the specific processes of system startup, authority initialization, key management, data sharing, proxy decryption, attribute revolution, and policy update is provided. First, it is assumed that all participants agree on the smart contract code and user roles. The Certifier Central deploys all smart contracts on the blockchain. The Certifier Central assigns participant roles to specific users identified by their blockchain accounts. Each Authority assigned the attributes to the appropriate user. Data users generate RSA key pairs and upload their public keys to the RSA smart contract, which Authorities use to verify identities and transmit attribute keys.

As shown in Algorithm 1, the Initialization phase involves constructing the Authorities network and establishing encryption parameters. To ensure complete decentralization, the public parameter generation procedure for MA-ABE has been redesigned as a Multi-Party Computation protocol. Specifically, a commit-then-open coin-tossing protocol is used to produce a random generator. Each Authority publishes the hash of the locally generated random pairing elements on the blockchain, then reveals these elements themselves, completing the commit-then-open protocol. Each Authority verifies that the hash of the pairing elements matches their revealed values and then processes all revealed values to compute the public parameters, which become part of the public parameters. Each Authority uses the AAKeyGen algorithm to create its own public/private key.

Algorithm 1: Authority initialization algorithm

Input: $process\_instance\_id$, $authorities\_names$, $addr$
Output: *public/private key*
initialization participant roles
Each Authority send metadata file to IPFS and get $metadata\_link$
Authority run $setAuthoritiesNames(process\_instance\_id, metadata\_link)$
checks the metadata files of other published Authority
if metadata file == participant roles **then**
    return *false*;
end if
Authority run $setElementHashed(process\_instance\_id, firstHashed, secondHashed)$
if Authority all submitted **then**
    return *true*;
else
    return *false*;
end if
Authority run $setElement(process\_instance\_id, firstelement, secondelement)$
gets all published elements
$generate\_public\_parameters(MAABE, IPFS\_api, process\_instance\_id)$
if all public parameters equal **then**
    return *true*;
else
    return *false*;
end if
creates *public/private key*
store Authority public key to Data store and blockchain

Key management securely distributes and assembles attribute keys, enabling authorized data users to access encrypted information. As shown in Fig. 3a, it represents the steps of the key management phase, this phase involves the Certifier Central storing user attributes and records of their process participation. For instance, the manufacturer holds the address **0 × 07e. . . A6**, associated with process 45,896,228. To access message data, users request keys from Authorities using their uid. Each Authority retrieves relevant user information from the Attribute Certifier Contract and generates an attribute key using this information, public parameters, and its own key. The Authorities then send these keys to the user for decryption. Only after collecting keys from all Authorities can a user combine them into the complete attribute key. No single authority is capable of generating the complete attribute key independently.

The process of data sharing is divided into encryption and decryption phases, with Algorithm 2 illustrating the data owner's encryption procedure. Based on MA-ABE, the data owner establishes an access control policy. Considering that business processes may require encryption of plaintext data of arbitrary size, we employ a two-stage hybrid encryption strategy. The data owner encrypts a randomly generated symmetric key using the Encrypt algorithm. The symmetric key is used to encrypt the actual business information using a symmetric key encryption algorithm. The encrypted symmetric key and data are stored in a formatted file (message). Decryption mirrors this process. Access to the original information requires the symmetric key, ensuring only policy-compliant users can decrypt the data, thus enforcing fine-grained access control.

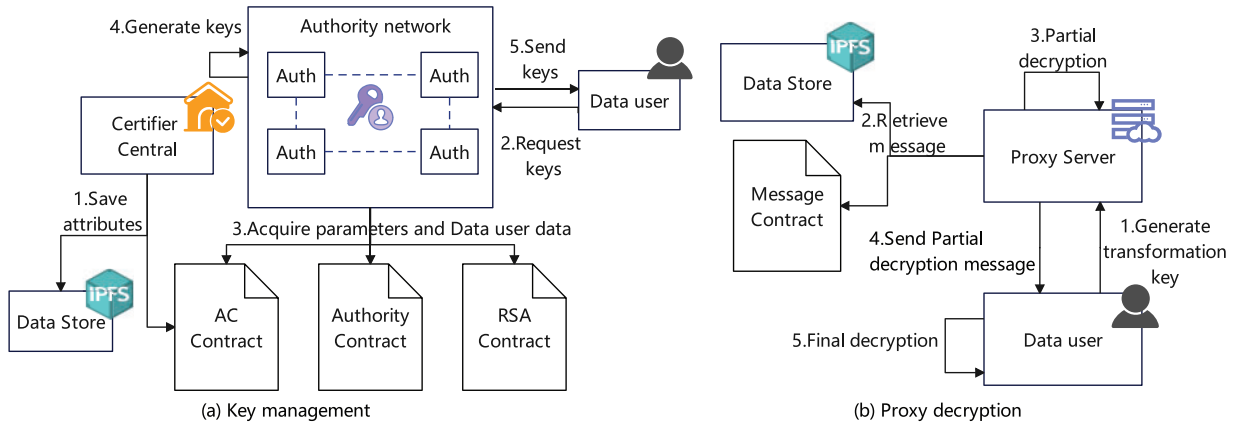

Figure 3: Key management and proxy decryption phases

Algorithm 2: Message encrypt algorithm

1: **Input:** GP, $(\mathbf{A}, \delta)$, $\{APK_\theta\}$
2: **Output:** *true or false*
3: get the *metadata* and GP
4: **if** *metadata* and GP are consistent **then**
5: return *true*;
6: **else**
7: return *false*;
8: **end if**
9: obtains all authority public keys
10: Symmetric key *Syk* randomly generated by the data owner
11: run $\mathsf{Encrypt}(Syk, (\mathbf{A}, \delta), GP, \{APK_\theta\}) \rightarrow CT$
12: encrypts business information with *Syk*.
13: send information and CT to IPFS and get $message\_link$
14: construct $TX = (message\_link, process\_instance\_id)$
15: store TX on the blockchain
16: **if** store TX successfully **then**
17: return *true*;
18: **else**
19: return *false*;
20: **end if**

As illustrated in Fig. 3b, the process of proxy encryption involves several key steps. Firstly, the data user generates transformation key and a retrieval key from the attribute key, then transmits the transformation key along with the message ID to the Proxy Server. The Proxy Server retrieves the corresponding message from the Message Contract and Data Store using the message ID and applies the transformation key for partial decryption. It forwards the partially decrypted message to the data user, who completes the decryption process using the retrieval key to obtain the symmetric key, thereby gaining access to the information.

In Algorithm 3, it describes the process of attribute revocation in MADEXA. The policy update process in MADEXA can be obtained through the Algorithm 4. In their algorithm, "link" refers to the content identifier (CID) used for rebinding in IPNS.

Algorithm 3: Attribute revocation algorithm

1: **Input:** *attribute x*, $UAK_{S,uid}$, RUK_x, UCK_x, *message*
2: **Output:** *true or false*
3: Authority generates new attribute group key
4: Authority update Attributes_user set and publish to IPFS
5: Data user run $\mathsf{KeyUpdate}(uid, UAK_{S,uid}, x, RUK_x) \rightarrow UAK'_{uid,x}$
6: Authority run $\mathsf{CTUpdate}(message, x, UCK_x) \rightarrow CT^*$
7: Data owner store message CT^* to Data Store and get $re\_message\_link$
8: publish $re\_message\_link$ on the IPFS
9: **if** publish $re\_message\_link$ successfully **then**
10: return *true*;
11: **else**
12: return *false*;
13: **end if**

Algorithm 4: Policy update algorithm

1: **Input:** $(\mathbf{A}, \delta)$, $(\mathbf{A}^*, \delta^*)$, GP, $\{APK_\theta\}$, *message*
2: **Output:** *true or false*
3: run $\mathsf{UKeyGen}((\mathbf{A}, \delta), (\mathbf{A}^*, \delta^*), GP, \{APK_\theta\}) \rightarrow UK$
4: run $\mathsf{CTUpdate2}(message, UK) \rightarrow \widetilde{CT}$
5: store new message $\widetilde{CT}$ and get $re\_message\_link$
6: publish $re\_message\_link$ on the IPFS
7: **if** publish $re\_message\_link$ successfully **then**
8: return *true*;
9: **else**
10: return *false*;
11: **end if**

4.3 Data Structure

In Table 2, it presents the format of messages that are encrypted and stored in the Data Store. Each storage entry is composed of one or more slices, designed to meet the access needs of different participants. Each file consists of metadata and a body. The metadata contains the sender, process id, and unique message id, while the body contains encrypted key-value pair data for easy indexing and retrieval, with each slice identified by a unique slice id. The sender corresponds to the user's ethereum address and is used to identify the sender of the message. The process id and message id are fixed-length random numbers generated by the entity to uniquely identify each process and message instance. Key corresponds to the symmetric key that has been encrypted using MA-ABE, and Fields indicates the JSON-formatted data encrypted by the above symmetric key. We achieve finer-grained access control management by setting access policies on message slices. The messages are stored in the Data Store, retrievable via resource locators, and their integrity and authenticity are verified through hashing.

5 Security Proof

Our main security theorem is shown below:

Theorem 1. *If the architecture proposed by RW15 maintains static security under q-DPBDHE2 assumptions in the random oracle model, then all probabilistic polynomial-time (PPT) adversaries attempting to statically break our scheme within this model will have a negligible success probability.*

Proof. Suppose there exists a PPT adversary capable of compromising our scheme with a non-negligible advantage ε. Then, we can construct a simulator $\mathscr{B}$ in the random oracle model that breaks the static security of the RW15 scheme with the same advantage ε. □

GlobalSetup:$\mathscr{B}$ obtains the GP from the challenger $\mathscr{C}$, and then forwards them to the adversary $\mathscr{A}$.

Static security:$\mathscr{A}$ establishes a set of corrupt permissions $\mathscr{U}_C \subset \mathscr{U}_\Theta$ and the remaining uncorrupted permissions $\mathscr{U}_N \subset (\mathscr{U}_\Theta - \mathscr{U}_C)$. The attacker $\mathscr{A}$ outputs the lists m_0, m_1, $(\mathbf{A}^*, \delta^*)$, $(\mathbf{A}_i^*, \delta_i^*)$, $(\mathbf{A}_j^*, \delta_j^*)$ and a sequence of (uid, S_{uid}). $\mathscr{B}$ obtains this information and subsequently forwards it to $\mathscr{C}$. Our proof is similar to RW15 regarding Authority Public Keys and Secret Keys.

Transform Keys: $\mathscr{B}$ performs Secret key query to obtain the $UAK_{S,uid}$ from $\mathscr{C}$. $\mathscr{B}$ then selects $t \in \mathbb{Z}_p^*$ and computing TK_{uid}. Subsequently, $\mathscr{B}$ sends TK_{uid} to $\mathscr{A}$.

Encryption: Upon receiving a ciphertext query from the adversary, $\mathscr{B}$ forwards this request to $\mathscr{C}$. $\mathscr{C}$ randomly selects $b \in \{0,1\}$ and run Encrypt$(m_b, (\mathbf{A}^*, \delta^*), \{APK_\theta\}, GP) \to CT$. Finally, $\mathscr{B}$ transmits the ciphertexts CT to $\mathscr{A}$.

KeyUpdate Keys: In response to a KeyUpdate query from the adversary, $\mathscr{B}$ conducts a secret key query operation to obtain $UAK_{S,uid}$ from $\mathscr{C}$. $\mathscr{B}$ executes the KeyUpdate algorithm to refresh the attribute key $UAK'_{S,uid}$. This updated key is subsequently delivered to $\mathscr{A}$.

UKeyGen Keys: For a UKeyGen query, $\mathscr{B}$ forwards the request to $\mathscr{C}$, which does not provide an advantage to $\mathscr{A}$. $\mathscr{C}$ independently selects the challenge message m_b and generates encrypted messages EnMess (m_b) using $(\mathbf{A}_i^*, \delta_i^*)$. Assuming identical encryption randomness for both m_0 and m_1, it follows that EnMess (m_0) = EnMess (m_1). Thus, when $\mathscr{C}$ executes UKeyGen and CTUpdate2 to obtain the updated ciphertext $\widetilde{CT}$, no information about the chosen message is revealed. $\mathscr{B}$ then relays $\widetilde{CT}$ to $\mathscr{A}$.

Guess. *$\mathscr{A}$ outputs a guess bit $b' \in \{0,1\}$, subsequently forwarding b' to $\mathscr{C}$. If $b' = b$, $\mathscr{A}$ win the game.*

If the adversary $\mathscr{A}$ has an advantage $Adv_{\mathscr{A}}(\lambda) = \varepsilon$ in destroys the scheme, then the simulator $\mathscr{B}$ can also be destroyed the RW15 scheme has the same advantage $Adv_{\mathscr{A}}(\lambda) = \varepsilon$. As demonstrated in [20], the RW15 scheme is non-adaptively secure under the q-DPBDHE2 assumption within the random oracle model, thus the proposed scheme also satisfies this security property.

Theorem 2. *Our proposed scheme is secure against user collusion attacks.*

Proof. When an attribute x is revoked from a user uid, a different randomly generated value of d_x corresponds in the revocation update key $RUK_x = g^{d_x(GK'_x - GK_x)}$ for each unrevoked user uid'. This prevents a revoked user from using another user's revocation update key to update their secret key. At the same time, it is difficult for non-revocation users to calculate d_x and $GK'_x - GK_x$ by solving the discrete logarithm problem, which prevents it from helping revocation users to update keys. □

Theorem 3. *Our attribute revocation enables forward and backward security.*

Proof. When attribute x is revoked from a data user uid, all relevant ciphertexts are re-encrypted by the authority according to $C'_{4,i} = C_{4,i}^{GK_x/GK'_x}$, making it highly difficult for revoked users to revert the re-encrypted

ciphertexts. Meanwhile, the attributes of the newly joined users satisfy the access policy of the ciphertext, and they can still decrypt the previously published ciphertext.□

Theorem 4. *Our scheme is resistant to ciphertext rollback attacks.*

Proof. Our plan involves updating ciphertext through attribute revocation and policy update. When updating ciphertext, our strategy is not to give any ciphertext update items to any other entity. In attribute revocation, AA does not provide ciphertext update items to any other entity, while in policy update, users do not provide them. By way, the attacker cannot roll back the updated ciphertext to its previous state. □

6 Analysis and Experiment

6.1 Theoretical Analysis

In Table 3, we compare our scheme with various MA-CP-ABE schemes. The GLGL23 [25] employs an access tree structure, while the proposed scheme and others use LSSS structures. The GLGL23 and ZLWR24 support user-level attribute revocation, while our scheme and others implement attribute-level revocation. The schemes [24–27] are susceptible to collusion attacks. The LJ18 [24] direct transmission of attribute group keys from AA to users can lead to adversaries obtaining revoked keys. Additionally, the scheme [24–28] are vulnerable to ciphertext rollback attacks, where adversary may exploit cloud-obtained update items to compromise backward security. In contrast, our scheme and ZHZL22 [29] conduct all ciphertext updates independently by AA without delegating update items to the cloud. Furthermore, CBZ22 [26], GLGL23 [25], and GWZ23 [27] exhibit significant security flaws, potentially allowing malicious entities to obtain plaintext.

Table 3: Feature comparison with other MA-CP-ABE schemes

Scheme	Access structure	PD	RT	PU	Security model	WCA	CRA	Blockchain
LJ18 [24]	LSSS	✓	Attribute	✓	ROM	✗	✗	✗
HKQ21 [28]	LSSS	✓	Attribute	✗	ROM	✓	✗	✗
ZHZL22 [29]	LSSS	✓	Attribute	✗	ROM	✓	✓	✗
CBZ22 [26]	LSSS	✓	Attribute	✗	ROM	✗	✗	✗
GLGL23 [25]	Access tree	✓	User	✗	ROM	✗	✗	✓
GWZ23 [27]	LSSS	✓	Attribute	✗	SM	✗	✗	✓
ZLWR24 [30]	LSSS	✓	User	✓	SM	✓	✗	✓
Ours	LSSS	✓	Attribute	✓	ROM	✓	✓	✓

Note: PD: Proxy Decryption. RT: Revocation Type. PU: Policy Update. ROM: Random Oracle Model. SM: Standard Model. WCA: Withstand Collusion Attack. CRA: Ciphertext Rollback Attack.

In the CBZ22 scheme, a critical security flaw occurs during the Online.Enc phase, where parameters $C_{4,j}$ and $C_{5,j}$ exposes λ, closely associated with the secret value s. This exposure allows an adversary with access to these ciphertext components to deduce s, thereby decrypting the ciphertext and recovering the plaintext. In the GLGL23 scheme, due to the desire to achieve user Escrow Free, random numbers related to attributes are generated by users. This design allows an adversary to forge random numbers for attributes they do not possess, thereby generating false attribute keys. Once an adversary acquires sufficient valid attribute keys, they can decrypt the ciphertext. Algorithm 2 of the GWZ23 scheme also contains potential security risks. Here, $\varepsilon\xi_i$ is part of the transformation key (TK_{uid}), while the retrieval key (RK_{uid}) equals ε. According to the scheme description, TK_{uid} is directly provided by the authoritative sending server. It is noteworthy that $\varepsilon\xi_i$ is the result of multiplying two random numbers, and this operation is not based on any widely recognized

hard mathematical problem. Consequently, it is feasible for an adversary to find ε through brute force or exhaustive search. Once the adversary determines RK_{uid}, they could gain access to the plaintext, threatening the confidentiality and integrity of the system.

In Table 4, we present a comparative analysis of computational complexities between our scheme and other studies. Here, P and E denote bilinear pairing and exponentiation operations in group $\mathbb{G}$, while H and M represent hash and multiplication operations, respectively. We use n_k, n_c, and n_d to indicate the number of attributes for key generation, policy setting, and decryption, respectively. For the proxy decryption phase, the complexity of the GenTR algorithm in our scheme and those from [24,26–29] scales linearly with the number of user attributes. GWZ23 [27] incorporates symmetric decryption and hash verification in the final decryption step, due to its encryption mechanism. Preprocessing for proxy decryption in CBZ22 [26] and GWZ23 occurs during key generation, not proxy decryption. Although ZHZL22 [29] and CBZ22 [26] have reduced preprocessing operations, this leads to increased P operations in ciphertext updates for attribute revocation. Our scheme and HKQ21 maintain identical computational complexity for the final decryption operation, adding only one extra multiplication compared to LJ18. When discussing attribute revocation, our scheme aligns with [28,29] when users update their attribute keys. The CTUpdate in other schemes modifies the revoked attribute's associated ciphertext portion but incurs additional operations compared to ours. Overall, our attribute revocation and proxy decryption have low computational complexity.

Table 4: Comparison of computation cost

Schemes	Proxy decryption			Attribute revocation[1]	
	GenTR	PartDec	FinalDec	KeyUpdate[2]	CTUpdate [3]
LJ18 [24]	$2n_kE + 2n_cE$	$3n_dP + n_dE + (4n_d - 1)M + H$	$E + M$	E	$(2n_c + 1)P + (6n_c + 1)E + (2n_c + 1)M + n_cH$
HKQ21 [28]	$2n_kE$	$3n_dP + 2n_dE + (4n_d - 2)M + H$	$E + 2M$	M	$E + M$
ZHZL22 [29]	$(n_k + 1)E$	$2n_dP + 3n_dE + 3(n_d - 1)M$	$E + 2M + 2H$	M	$P + 2E + 2M$
CBZ22 [26]	$(n_k + 1)E$	$3n_dP + (4n_d + 1)E + (6n_d - 2)M + H$	$2E + 2M$	$E + M$	$P + E + 2M$
GWZ23 [27]	$(2n_k + 3)E$	$(3n_d + 1)P + n_dE + 3n_dM + H$	$E + M + H + Dec_{sym}$	–	n_tE
Ours	$2n_kE$	$3n_dP + 2n_dE + (4n_d - 2)M + H$	$E + 2M$	M	E

Note: Dec_{sym} is a symmetric decryption operation. n_t is the number of layers in the KEK tree. [1]One atribute is revoked from one user. [2]One user updates his/her secret keys. [3]Proxy server/AA updates one relevant.

6.2 Experiment and Evaluation

The performance of the proposed scheme was assessed by the Charm[1]. The experiments utilized a super singular symmetric elliptic curve group("SS512"), which is defined over a 512-bit base field. The experiments were run on a VMware® Workstation 15 Pro virtual machine platform, which was configured with a 2.60 GHz Intel Core processor, 2.0 GB of memory, and a 64-bit Linux Ubuntu 16 operating system. The findings from all experiments are calculated as the mean of 10 separate runs.

As shown in Fig. 4a–c, the efficiency of proxy decryption is primarily explored. For this experiment, the system included 4 AAs, and the number of attributes per AA was increased according to the encryption policy. In Fig. 4a, our GenTR algorithm incurs less time than LJ18, with a slower increase as the number of

[1]https://github.com/JHUISI/charm (accessed on 27 January 2025)

attributes grows. The LJ18 scheme includes an extra ciphertext processing step that increases linearly with the number of attributes. In Fig. 4b, our FinalDec decryption times are slightly longer but generally remaining stable compared to LJ18. After increasing the number of experiments, LJ18 still occasionally has a little more than us, which may be an experimental inaccuracy due to the short decryption time. As shown in Fig. 4c, the decryption phase time cost for users is presented. The proxy decryption time of user is the sum of the GenTR and FinalDec algorithms' times. When the number of attributes is 20, the LJ18 scheme's proxy decryption requires 88.016 ms compared to 123.414 ms without it; our scheme demands 52.564 ms with proxy decryption *vs*. 98.278 ms without. Through proxy decryption, our scheme reduces the overall computational workload for users by approximately 50%, despite the added overhead. Overall, these figures demonstrate that our scheme performs better in handling a large number of attributes during proxy decryption.

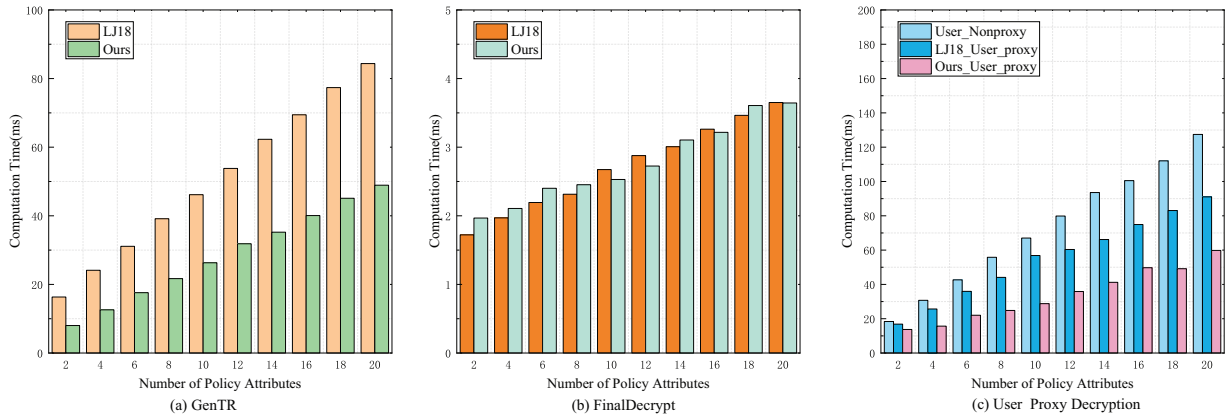

Figure 4: Experimental performance of proxy decryption in our scheme. (a) Running time of GenTR algorithm. (b) Running time of FinalDec algorithm. (c) The decryption time for the user not using proxy decryption and the decryption time of the user when the user using proxy decryption

For proxy decryption, the handled key and ciphertext constitute the main communication overhead. In Table 5, it shows the communication overhead comparison of GenTR algorithm under different numbers of attributes. Compared to the LJ18 scheme, our scheme only needs to transmit the key during processing, instead of having to transmit both the ciphertext and the key for processing as in LJ18. In addition, due to this algorithm are based on attributes. Therefore, when the number of attributes increases, the communication overhead of the LJ18 scheme will be significantly higher than our proposed scheme. As shown in Table 6, the communication overhead of the ciphertext is compared. In the PartDec algorithm, our scheme requires the transmission of two attribute-independent ciphertext parameters, in contrast to the LJ18 scheme which requires the transmission of only one similar parameter. So for messages of the same length, the costs in our scheme and the LJ18 scheme are essentially fixed, with slight fluctuations. Our scheme incurs slightly higher costs than the LJ18 scheme, which is consistent with the PartDec algorithm. In practice, our scheme reduces network bandwidth usage and improves communication efficiency.

Table 5: GenTR algorithm communication overhead (KB)

Schemes	Number of attributes									
	2	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
[24]	3.19	5.24	7.24	9.20	11.17	13.12	15.07	17.02	18.96	20.84
Ours	1.16	1.75	2.34	2.92	3.49	4.06	4.61	5.18	5.73	6.29

Table 6: Communication overhead of ciphertext in PartDec algorithm (KB)

Schemes	Number of attributes									
	2	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
[24]	0.81	0.82	0.83	0.84	0.85	0.85	0.86	0.86	0.87	0.87
Ours	1.11	1.12	1.12	1.13	1.14	1.15	1.15	1.16	1.16	1.17

In Fig. 5, it shows the time consumption under different numbers of revoked attributes. As shown in Fig. 5b, the execution time of the LJ18 scheme's CTUpdate algorithm is basically constant, about 198 ms. The LJ18 scheme's CTUpdate algorithm involves re-randomizing the entire ciphertext, this process is not affected by the number of revocation attributes [24]. As shown in Fig. 5c, it represents the overall attribute revocation time. The execution time of our scheme's KeyUpdate and CTUpdate algorithm increases linearly with the number of revoked attributes. Our execution time is significantly lower than that the LJ18 scheme. In Fig. 5, it indicates that our scheme performs better when handling the revocation of a large number of attributes.

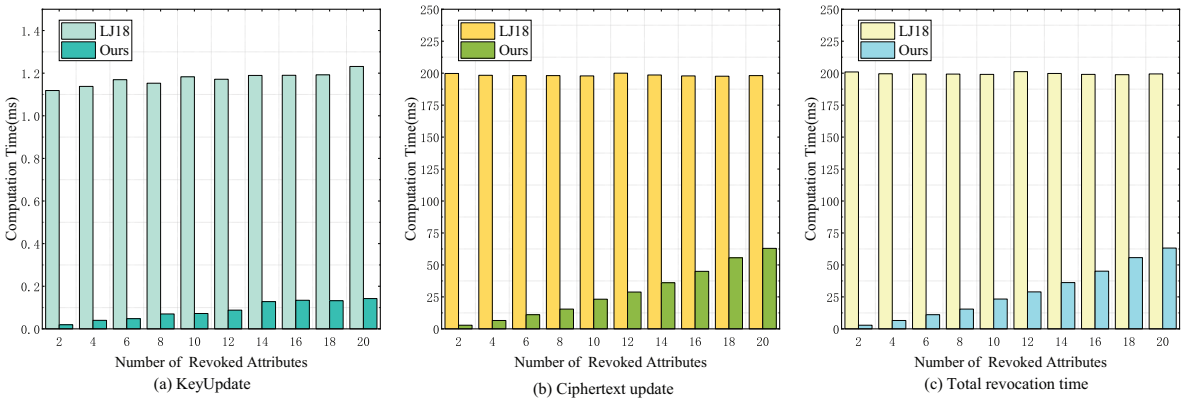

Figure 5: Experimental performance of attribute revocation in our scheme. (a) Running time of KeyUpdate algorithm. (b) Running time of CTUpdate algorithm. (c) Total time for attribute revocation

6.3 Integrate with Process Execution Engine

In the Ethereum protocol, the cost of executing or deploying a smart contract is measured in gas, reflecting the number of opcodes invoked [31]. We use ganache to simulate the merged version of Ethereum. The version used for Smart Contract is Solitidy^0.8.0. To estimate mainnet expenses, we adopted an average gas price of 6.50 Gwei and an ETH to USD exchange rate of 3192.15 as of 09 July 2024. Contract 1 is

primarily designed to manage and record users' RSA public keys as well as establish user identities. Through this contract, each user can register their public key and associate it with a unique identity. This process ensures the verifiability of all participants' identities while providing the necessary public key infrastructure for subsequent encryption and decryption. Authority Contract invokes a decentralised public parameter generation mechanism that allows multiple authorities or nodes to jointly participate in the public parameter creation process. Message Contract focuses on implementing the information-sharing functionality.

Contract 1 RSA and Attribute Certifier Contract

Structures

UserpublicKeys: mapping Authority_attributes: mapping
Authority_attributes_users: mapping

Functions

setUserPublicKey(UserpublicKey) getUserPublicKey(address)
- Append or get user asymmetric public Keys
setA_attributes(instanceID, Authority_attribute) getA_attributes(instanceID)
- Append or get the attribute of authority control
setA_attributes_U(instanceID, Attributes_user) getA_attributes_U(instanceID)
- Append or obtain the user set of attributes.

Contract 2 Authority and Message Contract

Structures

authoritiesNames: mapping firstelementHasheds: mapping
secondelementHasheds: mapping firstelements: mapping
secondelements: mapping parameters: mapping
publicKeys: mapping Ciphertexts: mapping

Functions

setAuthoritiesNames (instanceID, authName)
- Append authority metadata file IPFS hash
setElementHashed (instanceID, firstHashed, secondHashed) getElementHashed (address, instanceID)
- Append or get the hash of two random pairing elements
setElement (instanceID, firstelement, secondelement) getElement (address, instanceID)
- Append or get two random pairing elements
PublicParameters (instanceID, parameter) PublicKey (instanceID, publicKey)
- store public parameters and each authority public key
equalparameters (parameters)
- check the legality of public parameters.
function setCiphertext (messageID, Ciphertexts) function getCiphertext (messageID)
- Append or get ciphertext

As illustrated in Table 7, four stages were evaluated: smart contract deployment, system startup, authority initialization, and data sharing. Specifically, smart contract deployment consumes the most gas at 2,357,360 units, approximately $49. In the long run, this cost will be amortized, as the smart contract will continue to be used for future similar scenarios. System startup and authority initialization consume 299,231

gas units ($6.2) and 473,516 gas units ($9.8), respectively, with both being one-time costs. Data sharing incurs a cost of 89,420 gas units ($1.80) per file, also a one-time expense. The results indicate that the average gas consumption across these stages is relatively low.

Table 7: Execution cost analysis

	Execution cost			
	Deploy	**System startup**	**Authority initialization**	**Data sharing**
Gas	2357360	299231	473516	89420
Fee (ETH)	0.01532284	0.00194500	0.00307785	0.00058123
USD	48.91280371	6.20873654	9.82497165	1.85537334

We integrated MADEXA with Caterpillar v1.0[2], serving as its data management layer to enhance secure data storage. As depicted in Fig. 6, the experimental results from the previous scenario are illustrated, the right side corresponds to the process of data sharing for MADEXA integration with Caterpillar. Specifically, the BPMN corresponding to the Single Window process depicted in Fig. 1 is uploaded to Caterpillar, which then converts it into a smart contract to be uploaded to Ethereum. The data sharing of the entire process is then visualised. Users input data through Caterpillar's interface and mark it with the prefix "@MAABE:" to indicate that they wish to encrypt the information with MAABE. This example focuses on the transmission of customs clearance data by international suppliers to customs authorities, specifically the order information in the export documents listed in Table 2. After encryption, the first parameter remains unchanged, while the second is replaced by an IPFS link. Consequently, the execution engine logs the resource locator on the blockchain for audit purposes. The process outlined enables the maintenance of each customs clearance process instance on the ethereum, the execution of workflow routing through smart contracts, and the real-time monitoring of the process state by all participants, thereby facilitating on-chain data sharing and visualisation.

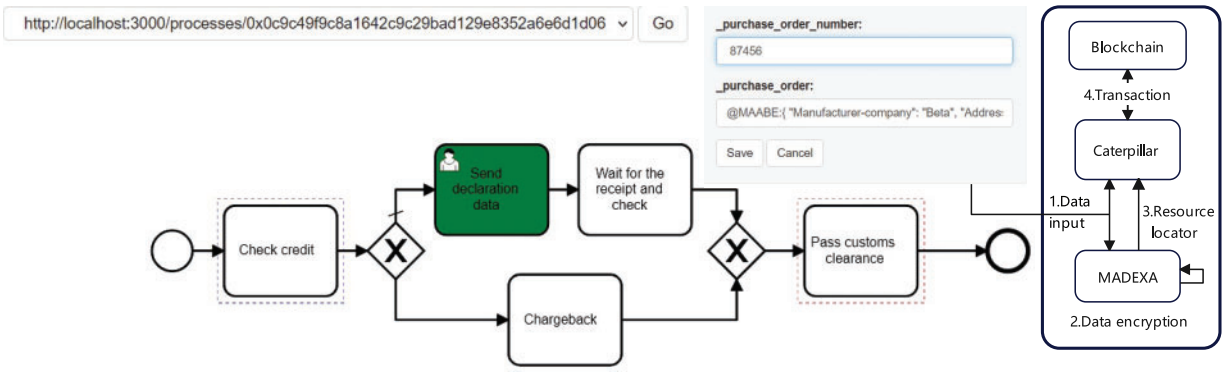

Figure 6: Integration with Caterpillar

[2] https://github.com/orlenyslp/Caterpillar (accessed on 27 January 2025)

7 Conclusion and Future

This paper presents an attribute-based multi-authoritative data exchange architecture and integrates it with a process execution engine, which aims to address the challenges associated with cross-border trade data sharing. The architecture provides fine-grained data access control management for cross-border trade and supports proxy decryption, attribute revocation, and policy update. The proposed scheme ensures the security and visibility of cross-border trade data-sharing processes, effectively overcoming the challenges of effective sharing and trust in traditional cross-border trade information systems. We conducted a thorough discussion and provided a formal demonstration of the security aspects of our scheme. Through detailed performance analysis and simulation experiments, the superiority and feasibility of the proposed scheme in real-world application scenarios are demonstrated, effectively balancing security requirements and operational efficiency.

In our approach, we have identified several limitations that need to be addressed in future research. The user attribute keys are currently generated by an authoritative entity, which raises concerns regarding key management. We envision propose a more decentralized approach to generating attribute keys, wherein users collaborate with one or more trusted third parties in the key generation process, using techniques like zero-knowledge proofs or secure multi-party computation to ensure security [25]. This design realizes the paradigm of self-sovereign keys [32]. Although the current research focuses more on the theoretical aspect, aiming to provide a reference framework for cross-border trade data sharing, in future practical applications, we will need to consider the performance and applicability in different blockchain environments, as well as the system scalability under large-scale users and high-concurrency access.

Acknowledgement: The authors would like to express special thanks to the collaborators for their valuable insights and discussions in completing this review.

Funding Statement: This work is supported by Hainan Provincial Natural Science Foundation of China Nos. 622RC617, 624RC485. Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) (SKLNST-2023-1-07).

Author Contributions: Shenjian Xiao: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing—original draft, Visualization. Xiaoli Qin: Supervision, Methodology, Funding acquisition, Writing—original draft. Yanzhao Tian: Supervision, Conceptualization, Methodology, Project administration, Writing—review & editing, Funding acquisition. Zhongkai Dang: Editing, Polishing, Conceptualization. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The authors confirm that the data supporting the findings of this study are available within the article.

Ethics Approval: This study did not involve any human or animal subjects, and therefore, ethical approval was not required.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Chang YL, Iakovou E, Shi WD. Blockchain in global supply chains and cross border trade: a critical synthesis of the state-of-the-art, challenges and opportunities. Int J Prod Res. 2020 Apr;58(7):2082–99. doi:10.1080/00207543.2019.1651946.
2. Du RQ, Duval Y. An exploration of paperless trade implementation in UN and other international conventions involving cross-border exchange of trade-related data and documents. In: UNNExT working paper series no. 10. Bangkok: ESCAP; 2024 Jun.

3. Castellani L, Ramirez Ortiz CM. Driving digitalization of global trade: UNCITRAL model law on electronic transferable records. Manila, Philippines: Asian Development Bank; 2023.
4. Casanova D, Dierker D, Hausmann L, Jensen B, Stoffels J. The multi-billion-dollar paper jam: unlocking trade by digitalizing documentation. McKinsey's Travel, Logist Infrastruct Practice. 2022;1–8. [cited 2024 Jun 1]. Available from: https://www.mckinsey.com/industries/logistics/our-insights/the-multi-billion-dollar-paper-jam-unlocking-trade-by-digitalizing-documentation.
5. Chen TW, Yu YM, Duan ZT. Data access & sharing approach for trade documentations based on blockchain technology. In: 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE); 2019 Oct; IEEE. p. 1732–6.
6. Bondarenko A. Advancing the digitalization of trade supporting documents in the Eurasian Economic Union member States and beyond. In: UNNExT working paper series no. 3. Bangkok: ESCAP; 2023 Sep.
7. Sarker S, Henningsson S, Jensen T, Hedman J. The use of blockchain as a resource for combating corruption in global shipping: aan interpretive case study. J Manag Inf Syst. 2021 Apr;38(2):338–73. doi:10.1080/07421222.2021.1912919.
8. World Trade Organization. Securing cross-border trade through advanced technologies. WTO; 2022 Mar. p. 4–9. [cited 2024 Jun 1]. Available from: https://www.wto-ilibrary.org/content/books/9789287071002c002.
9. Lee H, Yeon C. Blockchain-based traceability for anti-counterfeit in cross-border e-commerce transactions. Sustainability. 2021 Jan;13(19):11057. doi:10.3390/su131911057.
10. Zhang XM, Zha XY, Zhang HY, Dan B. Information sharing in a cross-border e-commerce supply chain under tax uncertainty. Int J Electron Commer. 2022 Jan;26(1):123–46. doi:10.1080/10864415.2021.2010007.
11. Theodouli A, Manganopoulou E, Kalfoutzos A, Tzikas A, Tsislianis C, Ioannidis D, et al. Towards a secure and transparent blockchain-based system for e-commerce deliveries. In: Manulis M, Maimuț D, Teşeleanu G, editors. Innovative security solutions for information technology and communications. Cham: Springer Nature Switzerland; 2024. p. 113–25.
12. Zhao HM. A cross-border e-commerce approach based on blockchain technology. Mob Inf Syst. 2021;2021(1):2006082. doi:10.1155/2021/2006082.
13. Hazarika B, Mousavi R. Review of cross-border e-commerce and directions for future research. J Glob Inf Manag. 2022 Jan;30(2):1–18.
14. Thoppae C, Praneetpolgrang P. An analysis of a blockchain-enabled e-government document interchange architecture (DIA) in Thailand. TEM J. 2021 Aug;10(3):1220–7. doi:10.18421/TEM.
15. Singapore's Infocomm Media Development Authority (IMDA). TradeTrust-digitised global trade. [cited 2024 Jun 1]. Available from: https://www.tradetrust.io/.
16. Singapore Together Alliance for Action (AfA). Singapore Trade Data Exchange (SGTraDex). [cited 2025 Feb 14]. Available from: https://sgtradex.com/.
17. Liu ZY, Li ZP. A blockchain-based framework of cross-border e-commerce supply chain. Int J Inf Manag. 2020;52(10):102059. doi:10.1016/j.ijinfomgt.2019.102059.
18. Wu LPF, Li X, Zhao R, Lu WS, Xu JY, Xue F. A blockchain-based model with an incentive mechanism for cross-border logistics supervision and data sharing in modular construction. J Clean Prod. 2022;375(7):133460. doi:10.1016/j.jclepro.2022.133460.
19. Rahman SM, Al Omar A, Bhuiyan MZA, Basu A, Kiyomoto S, Wang G. Accountable cross-border data sharing using blockchain under relaxed trust assumption. IEEE Trans Eng Manag. 2020;67(4):1476–86. doi:10.1109/TEM.2019.2960829.
20. Rouselakis Y, Waters B. Efficient statically-secure large-universe multi-authority attribute-based encryption. In: Okamoto T, editor. International Conference on Financial Cryptography and Data Security. Berlin/Heidelberg: Springer; 2015. p. 315–32.
21. Waters B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano D, Fazio N, Gennaro R, Nicolosi A, editors. Public key cryptography–PKC 2011. Berlin/Heidelberg: Springer; 2011. p. 53–70.

22. Chinosi M, Trombetta A. BPMN: an introduction to the standard. Comput Stand Interfaces. 2012;34(1):124–34. doi:10.1016/j.csi.2011.06.002.
23. López-Pintado O, García-Bañuelos L, Dumas M, Weber I, Ponomarev A. Caterpillar: a business process execution engine on the ethereum blockchain. Softw: Pract Exp. 2019 Jul;49(7):1162–93. doi:10.1002/spe.2702.
24. Liu ZC, Jiang ZL, Wang X, Yiu SM. Practical attribute-based encryption: outsourcing decryption, attribute revocation and policy updating. J Netw Comput Appl. 2018 Apr;108:112–23.
25. Guo YY, Lu ZH, Ge H, Li JG. Revocable blockchain-aided attribute-based encryption with escrow-free in cloud storage. IEEE Trans Comput. 2023;72(7):1901–12.
26. Cui J, Bian F, Zhong H, Zhang Q, Xu S, Gu C, et al. An anonymous and outsourcing-supported multiauthority access control scheme with revocation for edge-enabled IIoT system. IEEE Syst J. 2022;16(4):6569–80.
27. Guo ZZ, Wang GL, Zhang GY, Li YX, Ni JQ. A multifactor combined data sharing scheme for vehicular fog computing using blockchain. IEEE Internet Things J. 2023;10(22):20049–64.
28. Huang KQ. Secure efficient revocable large universe multi-authority attribute-based encryption for cloud-aided IoT. IEEE Access. 2021;9:53576–88. doi:10.1109/ACCESS.2021.3070907.
29. Zhang ZS, Huang W, Zhou SJ, Liao YJ. A revocable multi-authority fine-grained access control architecture against ciphertext rollback attack for mobile edge computing. J Syst Archit. 2022;129(2):102589. doi:10.1016/j.sysarc.2022.102589.
30. Zhang LY, Li XM, Wu Q, Rezaeibagha F. Blockchain-aided anonymous traceable and revocable access control scheme with dynamic policy updating for the cloud IoT. IEEE Internet Things J. 2024;11(1):526–42. doi:10.1109/JIOT.2023.3287190.
31. Wood G. Ethereum: a secure decentralised generalised transaction ledger. Vol. 151. In: Ethereum project yellow paper; 2014. p. 1–32.
32. Di Francesco Maesa D, Lisi A, Mori P, Ricci L, Boschi G. Self sovereign and blockchain based access control: supporting attributes privacy with zero knowledge. J Netw Comput Appl. 2023;212(6):103577. doi:10.1016/j.jnca.2022.103577.