information

MDPI

Article

Cross-Domain Data Sharing Scheme Based on Threshold Proxy Re-Encryption

Bingtao Wu, Qiuling Yue *, Jie Zhu and Sidi Jiang

School of Cyberspace Security, Hainan University, Haikou 570228, China; wubingtao@hainanu.edu.cn (B.W.); zhujie@hainanu.edu.cn (J.Z.); 23220854110011@hainanu.edu.cn (S.J.)
* Correspondence: yueqiuling@hainanu.edu.cn

Abstract

Cross-domain data exchange is an important technical approach for realizing the value of data assets. However, lacking a single trusted root CA across domains, cross-domain schemes often encounter difficulties in authentication, controlled data flow, and fine-grained authorization. We propose a cross-domain data sharing scheme that uses decentralized identifiers and threshold proxy re-encryption. This scheme adopts the intra-domain leader node to verify the user identity, and the inter-domain multi-agent nodes collaborate in a threshold manner to handle cross-domain registration requests and re-encryption requests. Through threshold cooperation, the problem of single point of failure is effectively solved. The hash value of cross-domain registration information is stored on the blockchain, leveraging the immutable and traceable characteristics of blockchain to achieve trusted cross-domain data sharing. In addition, we introduce a ciphertext version tag to enable fast updates of re-encryption keys and use zero-knowledge proofs to verify re-encrypted ciphertext correctness. The security analysis indicates that our scheme has IND-CCA2 security under the DBDH assumption and can effectively resist collusion attacks. Performance analysis shows that our scheme is efficient, and can better meet the needs of cross-domain data sharing.

Keywords: cross-domain data sharing; blockchain; decentralized identifier; threshold proxy re-encryption; zero-knowledge proof

1. Introduction

The rapid development of information technology has increasingly highlighted the importance of data sharing in sensitive fields such as healthcare, finance and intelligent manufacturing [1–3]. The efficient and secure sharing of data not only promotes information flow and accelerates innovation, but also enhances the overall efficiency of various industries. However, due to the often existing differences in security policies and varying trust bases among different domains, cross-domain data sharing faces difficulties in achieving cross-domain mutual trust in practical applications [4,5]. This severely restricts the circulation of cross-domain data, thereby leading to the data island problem and hindering the release of the value of data elements. To solve this problem, many schemes have been proposed. For instance, ref. [6] introduced blockchain technology to achieve cross-domain certificate information sharing within a multi-CA architecture. Ref. [7] proposed the XD-BAuth framework based on the consortium chain, enabling identity mutual recognition among multiple domains in the Internet of Things. However, it is worth noting that the establishment of cross-domain trust also requires giving users the ability to autonomously

Check for updates

Academic Editor: Jiguo Li

Received: 9 January 2026
Revised: 26 February 2026
Accepted: 24 March 2026
Published: 31 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

generate and manage cross-domain identities, in order to meet the flexible and dynamic cross-domain identity management requirements in a decentralized environment.

In data sharing scenarios, proxy re-encryption (PRE) [8,9] serves as an efficient ciphertext access authorization mechanism. It protects data confidentiality by ensuring the secure flow and controllable sharing of encrypted data. However, the traditional single-proxy architecture heavily relies on a central node and has the risk of single-point failure. To address this issue, the threshold proxy re-encryption (TPRE) scheme was proposed [10,11]. By introducing multiple proxy nodes to collaboratively complete the re-encryption operation, a distributed architecture is adopted to enhance the fault tolerance and security of the system. Nevertheless, TPRE still lacks a verification mechanism for the re-encrypted ciphertext fragments, which cannot ensure the correctness of the ciphertext conversion.

To solve above problems, this paper proposes a cross-domain data sharing scheme based on decentralized identity (DID) [12] and a committee mechanism. In this scheme, users generate decentralized cross-domain DID, while the leader nodes of each domain serve as committee members to perform cross-domain authentication and process proxy re-encryption requests for users. When computing ciphertexts, a ciphertext version tag and non-interactive zero-knowledge proofs (NIZKPs) are introduced to achieve a flexible re-encryption key update strategy and ensure the verifiability of re-encrypted ciphertext fragments. Additionally, storing the hash value of users' cross-domain registration information on the blockchain effectively achieves tamper resistance and traceability [13]. The main contributions are as follows:

(1) This study proposes a threshold proxy re-encryption scheme based on DID, which solves the centralized single point of failure problem of a single proxy node. The ciphertext version tag was designed to control the correspondence between ciphertext and re-encryption key, which enhanced the security and flexibility of the system.
(2) This study employs NIZKPs to ensure the verifiability of the re-encrypted ciphertext, which improves the integrity and correctness.
(3) We conducted both security and performance analysis to demonstrate the security of our scheme. Comparisons with existing schemes show that our scheme enhances efficiency and better practicality.

The rest sections are organized as described below. Section 2 discusses previous study. Section 3 outlines foundational concepts. Section 4 presents the system model and defines key symbols. Section 5 details the scheme's construction. Section 6 provides security proofs, analysis, and performance evaluation. Section 7 concludes with a summary.

2. Related Work

Proxy re-encryption is an encryption technology that has been widely applied in various fields to enhance data privacy and flexibility, and to achieve secure data sharing. The PRE scheme is usually used in combination with IBE [14,15] and ABE [16,17]. For instance, ref. [18] proposed an IBE-PRE scheme and applied it to the medical Internet of Things to achieve data sharing and access control. Through blockchain for storage and permission management, it effectively prevents data tampering and substitution. Ref. [19] proposed an identity-based PRE to achieve internet data sharing, combined with the decentralization of blockchain. This method alleviates the performance bottleneck of centralized systems and achieves fine-grained access control. Ref. [20] proposes an identity-based broadcast PRE scheme that supports one-to-many data sharing. Ref. [21] proposed a concrete attribute-based proxy re-encryption for encrypted data sharing. Ref. [22] proposed a proxy re-encryption scheme based on CP-ABE. However, these schemes usually rely on a single proxy node to perform ciphertext conversion using a re-encryption key, making them vulnerable to collusion attacks and the risk of a single point of failure [23]. If the proxy node

colludes maliciously with other entities, it may lead to the leakage of sensitive information or the disruption of data integrity. Moreover, a single-point failure could also cause the interruption of the encryption service, threatening the security and availability of data sharing. To solve the above problems, refs. [24,25] proposed the TPRE scheme and integrated it with the blockchain to achieve secure cross-domain data sharing. Ref. [26] presents an ABE-TPRE scheme based on dot matrix cryptography that can resist quantum attacks on encrypted data. For the industrial Internet scenario, ref. [27] proposes a certificateless TPRE scheme. Adopt cloud-chain cooperation to enhance system efficiency and security. These schemes eliminate the central dependency through multi-agent nodes, they effectively avoid the single point of failure and improve the security of the system. However, there is still a lack of a systematic ciphertext verification mechanism, especially the correctness verification of the re-encrypted ciphertext fragments generated by the agent nodes, which cannot ensure the integrity and reliability of the ciphertext conversion process.

3. Preliminaries

3.1. Bilinear Maps

Let G_1 and G_2 be cyclic groups of prime order p and let P be a generator of G_1. A bilinear mapping $e : G_1 \times G_1 \to G_2$ exists and satisfies the following conditions:

(1) Bilinearity: for $\forall A, B \in G_1$ and $a, b \in \mathbb{Z}_p^*$, the pairing satisfies $e(aA, bB) = e(bA, aB) = e(A, B)^{ab}$;
(2) Non-degeneracy: $e(P, P) \neq 1$;
(3) Computability: for all $A, B \in G_1$, $e(A, B)$ can be calculated.

3.2. Decentralized Identifiers

Decentralized identifiers are unique identity codes created by users. They let users manage their own identity through distributed ledger systems. This removes the need for traditional central registration bodies.

Initially, users locally generate DID and the corresponding DID document in accordance with the W3C standards, and then upload them to the blockchain or distributed ledger. A trusted third party can issue verifiable credential (VC) for DID holders by using cryptographic methods, such as electronic signatures, to prove the authenticity of their identities or attributes. When it is necessary to present identity or attribute proof, users do not need to show the original credential VC. Instead, they can independently control the scope of information sharing by generating verifiable expression (VP), thereby ensuring the maximum protection of user privacy while completing the verification.

3.3. Fiat–Shamir Heuristic

The Fiat–Shamir heuristic [28] provides an effective approach to building NIZKPs. Its main method is to use hash functions instead of random challenges generated by the verifier. This method was first proposed by Fiat and Shamir in 1986 [29], and the detailed explanation is as follows:

(1) The prover randomly selects r from $\mathbb{Z}_p^*$. Then, calculate $u = r \cdot P$, where P is the generator of G. Let $h = x \cdot P$, with x being the secret value. After that, the prover calculates the challenge $c = H(P, h, u)$.
(2) The prover calculates $z = r + c \cdot x$. Then, the proof $\pi = (u, c, z)$ is transmitted to the verifier.
(3) After receiving the proof π, the verifier computes $H(P, h, u)$ and checks whether the result equals c. If the check succeeds, the verifier computes $z \cdot P$ and verifies whether $z \cdot P = u + c \cdot h$.

In this approach, security and zero-knowledge are ensured by the Fiat–Shamir heuristic without interaction.

4. System Model

4.1. System Architecture

This paper proposed scheme is shown in Figure 1. It involves six different entities: trusted institutions, leadership nodes, data owners, data users, cloud servers, and blockchain. Table 1 presents the key symbols used in this work, along with their definitions.

Table 1. Notations and Definitions.

Notations	Definitions
G	Additive group
G_T	Multiplicative group
p	The large prime order of groups G and G_T
P	Generator of G
PP	System public parameters
DID	Decentralized identifier
DID_doc	Document of DID
$UserInfo$	Information of user
n	Secret shares
t	Threshold value
VC	Verifiable credential
VP	Verifiable presentation
DK	Identity key for DID
Sig_i	Signature of i
Rk	Secret key of re-encryption
m	Plaintext
CT	Ciphertext for plaintext m
RCT	Re-encryption ciphertext for CT

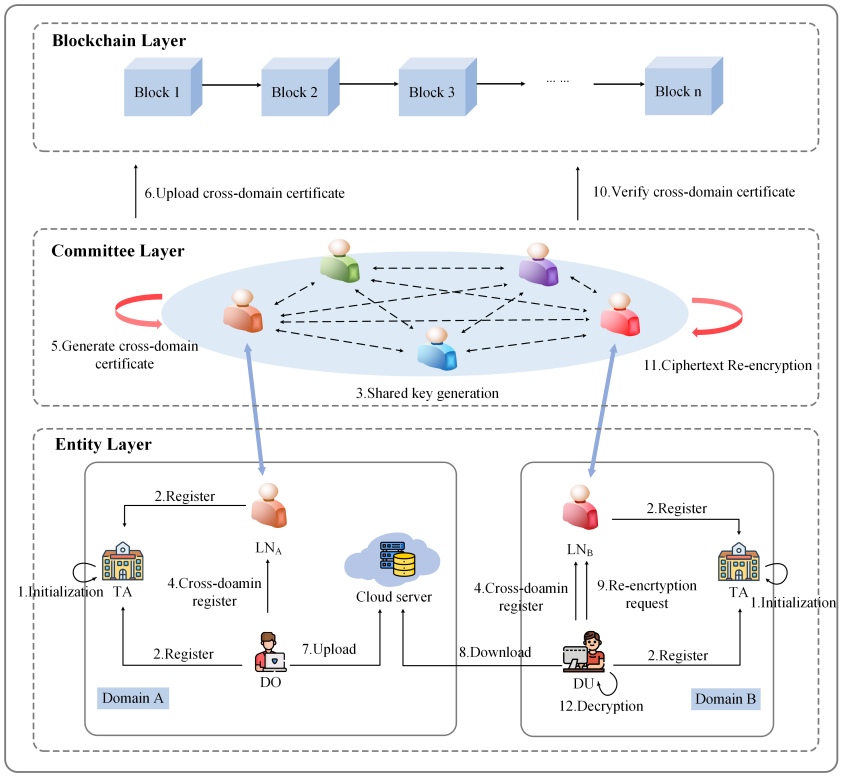

Figure 1. Cross-domain data sharing system model.

Data owner (DO): The DO acts as the data provider and sharing initiator, with its primary responsibilities being to encrypt plaintext messages prior to uploading it to the cloud and to generate Rk for authorized users.

Data user (DU): The DU is the data access and usage entity. Its data acquisition process is as follows: Before accessing data, the DU must first apply to the committee composed of domain leader nodes for cross-domain registration. After that, the DU initiates a re-encryption request for the target ciphertext to its own domain's leader node. Finally, the DU receives the re-encrypted ciphertext returned by the leader node and performs decryption to obtain plaintext messages.

Trusted authority (TA): The TA is responsible for issuing the VC for users within the domain, and it serves as the trust foundation for cross-domain identity authentication.

Leader node (LN): The LN communicates with leader nodes from other domains to form a committee. The LN is not only responsible for cross-domain registration of users within the domain and ciphertext re-encryption requests, but also acts as a proxy node to perform proxy re-encryption operations. Each LN maintains a local re-encryption key cache list, denoted as $KList = (DO, DU, C_v, Rk)$.

Cloud server (CS): The CS is responsible for storing encrypted data and providing data access services to users.

Blockchain (BC): The BC serves as an access-oriented consortium blockchain for storing the hash value of users' cross-domain information, providing decentralized cross-domain identity registration and auditing capabilities, and ensuring that the data sources are traceable and unalterable. This consortium blockchain is maintained jointly by the committee members.

4.2. Threat Model and Mitigation Strategies

This section analyzes the security threats faced by the scheme from four aspects: insider attack, collusive behavior, metadata leakage, and denial of service risk, and proposes corresponding mitigation strategies.

Internal attack: The LN may maliciously disclose the re-encryption key, tamper with the re-encryption key, or forge zero-knowledge proofs. We assume that at least half of the committee members are honest. Since this scheme is implemented through a threshold collaboration approach, a single node cannot obtain the complete re-encryption key (refer to Section 5.3.2). Additionally, the verifier can use the algorithm ProofVer (refer to Section 5.4.6) to detect malicious computing behavior.

Collusion behavior: For collusion attacks, this scheme employs a threshold mechanism ($t > n/2 + 1$) to provide effective defense. On the one hand, for the collusion of multiple leader nodes, since more than half of the leader nodes in the committee are honest nodes, the malicious leader nodes cannot restore the complete re-encryption key by sharing information. On the other hand, for the collusion between users and leader nodes, the user attempts to make the leader nodes re-encrypt the ciphertext into a form that the user can decrypt. However, since the malicious leaders cannot restore the complete re-encryption key, it still cannot correctly perform the re-encryption operation.

Metadata leakage: In response to the behavior of adversaries attempting to decrypt and obtain metadata by stealing ciphertext data in storage or transmission. In this scheme, even if any unauthorized third party obtains the ciphertext data, they will not be able to decrypt it. This is because direct decryption requires the private key of the data owner, and the re-encrypted ciphertext can only be decrypted by the designated user using their private key. All private keys are kept locally by the users, and adversaries cannot obtain the private keys, thereby ensuring the security of the metadata during storage and transmission.

In addition, this scheme can ensure the confidentiality of plaintext under the IND-CCA2 security model (proved by Theorem 1).

Denial of service risk: For the adversary, through frequent requests to disrupt the normal service behavior of the system, this scheme utilizes the user cross-domain registration information stored on the blockchain to verify the legitimacy of the requests. It can quickly identify and discard invalid requests. If it is found that an already registered user has maliciously made frequent requests, this scheme supports blocking their access by updating the user status on the blockchain, thereby ensuring the service quality of the proxy node.

4.3. Security Objectives

The following security objectives are achieved by the proposed scheme:

(1) Correctness: The LN can effectively verify the ciphertext CT integrity. The DU can correctly decrypt the re-encrypted ciphertext RCT.
(2) Confidentiality: The scheme prevents illegal theft and unauthorized access, ensuring plaintext data confidentiality. Only authorized legitimate users can obtain the original plaintext.
(3) Integrity: Ensure that CT and RCT remain unaltered during storage or transmission.
(4) Traceability: Ensure that uploaded data is traceable to its owner, meaning that when the source of the data needs to be traced. Trusted institutions can accurately obtain the identity of the data uploader through ciphertext.

5. Proposed Scheme

This section presents the operational workflow and algorithms of the scheme. The overall workflow is given below.

5.1. Initialization Phase

$\boldsymbol{Setup}(\mathbf{1}^{\lambda}) \rightarrow \boldsymbol{PP}$. Given the security parameter λ, output is shown in (1).

$$\begin{gathered} e : G \times G \rightarrow G_T \\ PP = (p, P, G, G_T, e, H_1, H_2, H_3, H_4, H_5) \end{gathered} \tag{1}$$

Among these, G is the additive group of prime order p, P is a generator of the group G, G_T is the multiplicative group of prime order p, $e : G \times G \rightarrow G_T$ is defined as a bilinear mapping, $H_1 : \{0,1\}^* \rightarrow G$, $H_2 : G \rightarrow \{0,1\}^*$, $H_3 : G \times G_T \times \{0,1\}^* \rightarrow G$, $H_4 : G_T \times \{0,1\}^* \rightarrow G$ and $H_5 : G \times G_T \rightarrow \mathbb{Z}_p^*$ are hash functions.

5.2. Cross-Domain Alliance Committee Key Establishment Phase

First, the leader nodes $LN_i (i \in 1, 2, \ldots, n)$ from each domain jointly establish a cross-domain federation committee. Based on PBFT consensus mechanism, the committee is responsible for user cross-domain registration application and proxy re-encryption application. Next, to establish the committee key, the procedure proceeds as follows:

(1) The $LN_i (i \in 1, 2, \ldots, n)$ in each domain, respectively, select secret values $s_i \in \mathbb{Z}_p^*$. Then, using the Verifiable Secret Sharing (VSS) [30] $Feldman(t, n)(t > \frac{n}{2})$ algorithm, the secret value s_i is split into n secret shares, denoted as $(s_{i,1}, s_{i,2}, \ldots, s_{i,n})$. The secret share $s_{i,j}$ is transmitted via a secure channel to the corresponding member $LN_j (j \neq i)$.
(2) Upon receiving the secret value $s_{i,j}$ transmitted by another leader node LN_i, LN_j first employs the VSS algorithm to validate the correctness of this secret share. If validation fails, it returns a failed message. If validation succeeds, it retains the share. When $LN_i (i \in 1, 2, \ldots, n)$ receives $n - 1$ valid secret shares, it proceeds with the following steps.

(3) LN_i sums all the secret shares it receives to obtain the shared key $s_i = \sum_{j=1}^{n} s_{j,i}$ $(j \neq i)$, then calculates its public key $pk_i = s_i \cdot P$, and broadcasts pk_i as its public key to other leader nodes in the committee.

(4) The LN_i in each domain compute the committee's public key individually using (2).

$$PK = \sum_{i=1}^{t} pk_i \cdot \lambda_i \mod p, \quad \text{where} \quad \lambda_i = \prod_{j=1, j\neq i}^{t} \frac{0-j}{i-j} \mod p \tag{2}$$

5.3. User Cross-Domain Registration Phase

This phase comprises two components: user identity authentication and cross-domain identity key registration. The specific workflow is illustrated in Figure 2. (The orange text indicates the cross-domain communication among the committee members).

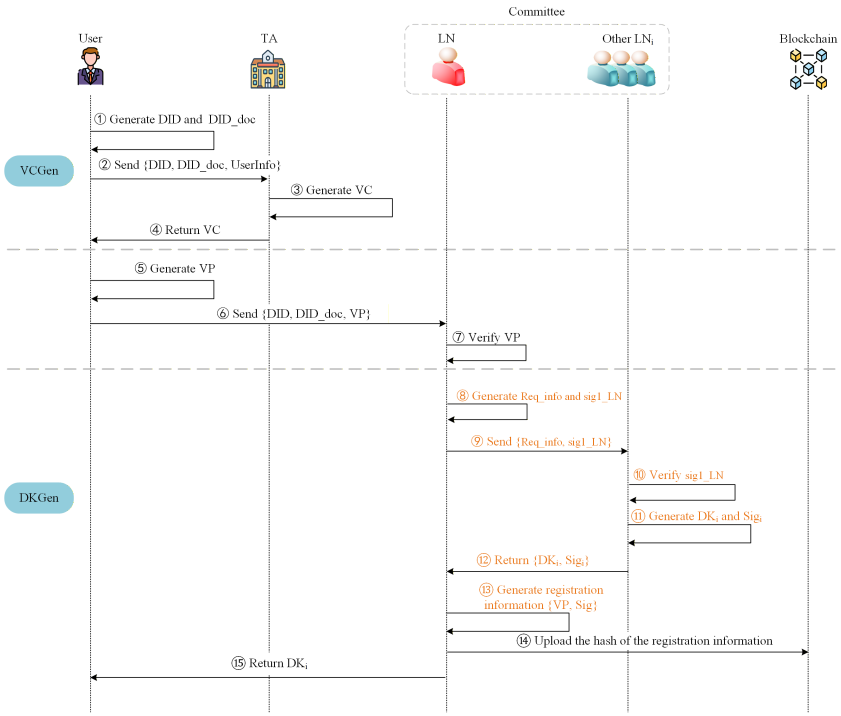

Figure 2. User registration workflow.

5.3.1. User Authentication

(1) First, the user sends their DID, DID_doc, and personal information $UserInfo$ to the TA within this domain to request identity authentication. The $UserInfo$ contains the necessary information for generating the VC.

(2) Then, TA uses the user's public key to verify the integrity of DID_doc and the validity of the signature. Once the verification is successful, a VC is generated locally for the user based on the $UserInfo$.

(3) The user can generate VP using the Selective Disclosure Protocol [31], thereby achieving privacy protection. Then, the user sends their DID, DID_doc, and VP to the local domain's leader node LN_i for cross-domain application.

(4) Upon receiving the application, LN_i verifies the VP submitted by the user. If the VP verification passes, the process proceeds to the next stage. If the VP verification fails, LN_i returns a verification failure message to the user.

5.3.2. Cross-Domain Identity Key Registration

(1) The LN_i acts as an agent node to generate a cross-domain request proxy information packet $Req_info = (DID_u, DID_LN_i, pk_u, VP, Request, T, \Delta T)$ along with the signature $sig1_LN_i$ for the user. Here, DID_LN_i represents the LN_i's identifier, Request denotes the user's requested permission, T is the timestamp, ΔT is the request validity period, and $sig1_LN_i$ is the LN_i's signature on the $Req_i nfo$ packet. LN_i broadcasts $Req_i nfo$ and signature $sig1_LN_i$ to all leader nodes within the committee.

(2) Upon receiving the message, the leading node $LN_j (j \neq i)$ within the committee first verifies the signature $sig1_LN_i$ by LN_i's pk_i. If the signature verification succeeds, it calculates the cross-domain identity key fragment $DK_j = s_j \cdot H_1(DID)$ for that user. Simultaneously, it generates a threshold signature $sig2_LN_j$ for VP using the threshold signature algorithm $TS.Sign$ [32]. LN_j encrypts DK_j using pk_u to generate ciphertext C_j. Then, the message $\{C_j, sig2_LN_j\}$ is transmitted to LN_i via secure channel.

(3) After receiving at least k $(t \leq k \leq n)$ messages, the LN_i first transmits k ciphertexts C_j to the user. It then aggregates the complete signature Sig using the threshold signature aggregation algorithm $TS.Comb$ and uploads the hash value of user's cross-domain registration information $\{VP, Sig\}$ to the blockchain to complete the user's cross-domain registration.

(4) The user receives at least t C_j messages, performs decryption, and uses the Lagrange interpolation algorithm to recover the complete identity key DK. The specific process is shown in (3) and the λ_i is shown in (2).

$$DK = \sum_{i=1}^{t} DK_i \cdot \lambda_i \mod p \tag{3}$$

5.4. Data Sharing Phase

There are seven primary parties involved in the data sharing phase: data encryption, re-encryption key generation, ciphertext re-encryption, proof generation, proof verification, ciphertext aggregation, and decryption. The specific flowchart is shown in Figure 3. (The orange text indicates the cross-domain communication among the committee members. The superscript (i) in the figure denotes information associated with the ith leader node.)

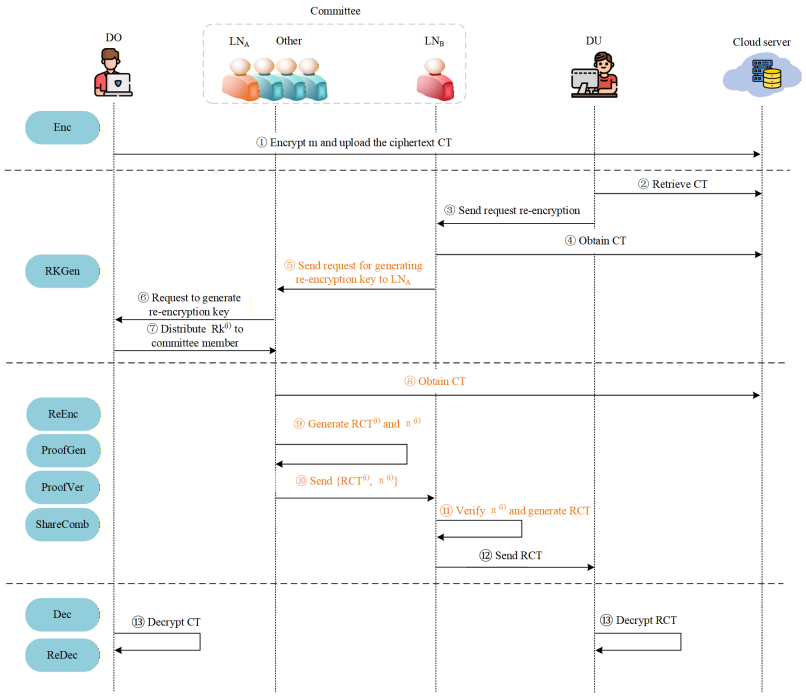

Figure 3. Cross-domain data sharing workflow.

5.4.1. Data Encryption

The DO (for example, Alice from domain A) inputs the committee public key PK, their decentralized identifier DID_{DO}, the plaintext m to be shared, and the ciphertext version tag $ver \in G$ into the algorithm Enc to output the ciphertext CT. Subsequently, the ciphertext CT and the leader node LN_A's identity identifier are uploaded to the CS. The LN_A obtains the corresponding registration information from the blockchain based on the uploader's DID. Algorithm 1 describes the particular procedure.

Algorithm 1 Enc

Input: DID_{DO}, PK, m, ver
Output: CT
Select random $r \in \mathbb{Z}_p^*$
Calculate $C_v = H_2(ver)$;
$C_1 = r \cdot P$;
$C_2 = m \cdot e(PK, H_1(DID_{DO}))^r \cdot e(P, ver)^r$;
$C_3 = r \cdot H_3(C_1, C_2, C_v)$;
return $CT = (C_v, C_1, C_2, C_3)$;

5.4.2. Re-Encryption Request

When DU (for example, Bob from domain B) initiates a ciphertext re-encryption request to the local domain's leader node LN_B, LN_B checks its local table $KList$ for a re-encryption key matching the ciphertext's version tag C_v. If not found, it must contact the leader node LN_A of the domain where the data owner DO is located. Then, via LN_A, it initiates a re-encryption key generation request to DO and proceeds to Section 5.4.3. If $KList$ contains the corresponding re-encryption key, it proceeds directly to Section 5.4.4.

5.4.3. Re-Encryption Key Generation

The re-encryption key is generated by using the Shamir secret sharing scheme [33] to generate re-encryption key fragments. The DO inputs its identity identifier DID_{DO} and identity key DK_{DO}, DU's identity identifier DID_{DU}, ciphertext version tag ver, secret share size n, and threshold value t. It executes the $RKGen$ to output $\{Rk_{DO\to DU}^{(i)}\}_{i=1}^{n}$, sends it to the LN_A, and the LN_A forwards it to the corresponding leader node LN_i in the committee. Algorithm 2 describes the particular procedure.

Algorithm 2 RKGen

Input: $DID_{DO}, DID_{DU}, DK_{DO}, n, t, ver$
Output: $\{Rk_{DO\to DU}^{(i)}\}_{i=1}^{n}$
Calculate $C_v = H_2(ver)$;
$H = H_4(e(DK_{DO}, H_1(DID_{DU})), DID_{DO}, DID_{DU})$;
$rk_{DO\to DU} = H - DK_{DO} - ver$;
Choose $y \in \mathbb{Z}_p^*, x_i \in \mathbb{Z}_p^*, (i \in 1, 2, \ldots, n)$;
Calculate $Y = y \cdot P$;
Select random $a_1, a_2, \ldots, a_{t-1} \in \mathbb{Z}_p^*$;
Polynomial $f(x) = a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_{t-1} \cdot x^{t-1}$;
For each x_i, do
Calculate $rk_{DO\to DU}^{(i)} = rk_{DO\to DU} + f(x_i) \cdot P$;
$K_i = e(P, rk_{DO\to DU}^{(i)})$;
$S_{i1} = H_6(Y, K_i, DID_{DO}, DID_{DU})$;
$S_{i2} = Y - S_{i1} \cdot DK_{DO}$;
return $Rk_{DO\to DU}^{(i)} = (x_i, rk_{DO\to DU}^{(i)}, K_i, S_{i1}, S_{i2}, C_v), i \in (1, 2, \ldots, n)$;

It is worth noting that only when the input *ver* of the algorithm *RKGen* is the same as the version tag *ver* of the ciphertext CT, can the correct proxy re-encryption key be generated. Then LN_i can correctly calculate the re-encrypted ciphertext fragment. In this way, DO can update the plaintext encryption policy by flexibly changing the value of *ver*, and generate different versions of ciphertext structures and corresponding re-encryption keys.

5.4.4. Re-Encryption of Ciphertext

LN_i first verifies the integrity of the CT by computing $C3' = H3(C_1, C_3, C_v)$. It then compares $e(C_1, C_3')$ with $e(P, C_3)$. If they are unequal, the ciphertext is discarded. Subsequently, the CT is re-encrypted by the algorithm *ReEnc*, generating re-encrypted ciphertext fragments $RCT^{(i)}$. Algorithm 3 describes the particular procedure.

Algorithm 3 ReEnc

Input: CT, $Rk_{DO\to DU}^{(i)}$
Output: $RCT^{(i)}$
Calculate $C_3' = H_3(C_1, C_2, C_v)$;
If $e(C_3, P) \neq e(C_3', C_1)$, **then**
  return $\perp$;
end
If there does not exist a re-encryption key in the *KList* with a version tag equal to C_v, **then**
  Request to generate a re-encryption key;
end
Calculate $C_4^{(i)} = e(C_1, rk_{DO\to DU}^{(i)})$;
  $C_5^{(i)} = C_2 \cdot C_4^{(i)}$;
return $RCT^{(i)} = (C_4^{(i)}, C_5^{(i)})$;

5.4.5. Verifiable Proof Generation

To achieve verifiability of re-encrypted ciphertext, LN_i executes the algorithm *ProofGen* concurrently with generating the re-encrypted ciphertext. Inputting the C_1 in CT, the $C_4^{(i)}$ in $RCT^{(i)}$, and the $Rk_{DO\to DU}^{(i)}$, it computes and outputs the proof $\pi^{(i)}$. Then, it sends $\pi^{(i)}$ along with $RCT^{(i)}$ to the LN_B. Through $\pi^{(i)}$, LN_i can demonstrate to LN_B that it indeed possesses the correct re-encryption key $Rk_{DO\to DU}^{(i)}$, and that $RCT^{(i)}$ is indeed the re-encrypted ciphertext fragment obtained through correct computation. Algorithm 4 describes the particular procedure.

Algorithm 4 ProofGen

Input: C_1, $C_4^{(i)}$, $Rk_{DO\to DU}^{(i)}$
Output: $\pi^{(i)}$
Select random $Q^{(i)} \in G$;
Calculate $u_1 = e(C_1, Q^{(i)}), u_2 = e(P, Q^{(i)})$;
  $c = H_5(C_1, C_4^{(i)}, K_i, u_1, u_2)$;
  $z = Q^{(i)} + c \cdot rk_{DO\to DU}^{(i)}$;
$\pi^{(i)} = (c, z, u_1, u_2, Y, K_i, S_{i1}, S_{i2})$;
return $\pi^{(i)}$;

5.4.6. Re-Encryption Ciphertext Aggregation

LN_B receives the $RCT^{(i)}$ and the proof $\pi^{(i)}$ sent from other leader nodes in the committee. Using the C_1 in CT, the $C_4^{(i)}$ in $RCT^{(i)}$, and the $\pi^{(i)}$ as input, LN_B verifies the validity of $\pi^{(i)}$ via the algorithm *ProofVer*. Algorithm 5 describes the particular procedure. If valid, it accepts the re-encrypted ciphertext fragment $RCT^{(i)}$. When at least t valid $RCT^{(i)}$

fragments are received, it executes the algorithm *ShareComb* to generate the complete RCT and sends it to the DU. Algorithm 6 describes the particular procedure.

Algorithm 5 ProofVer

Input: $C_1, C_4^{(i)}, \pi^{(i)} = (c, z, u_1, u_2, Y, K_i, S_{i1}, S_{i2})$
Output: *result*
Calculate $S'_{i1} = H_6(Y, K_i, DID_{DO}, DID_{DU})$;
If $S_{i1} \neq S'_{i1}$ or $e(Y, P) \neq e(S_{i2}, P) \cdot e(H_1(DID_{DO}), PK)^{S_{i1}}$ **then**
  return $\perp$;
end
Calculate $c' = H_5(C_1, C_4^{(i)}, K_i, u_1, u_2)$;
If $c' \neq c$ **then**
  return $\perp$;
end
If $e(C_1, z) \neq u_1 \cdot (C_4^{(i)})^{c'}$ *or* $e(P, z) \neq u_2 \cdot (K_i)^{c'}$ **then**
  return $\perp$;
end
return *true*;

Algorithm 6 ShareComb

Input: $\{RCT^{(i)}\}_{i=1}^{t}$
Output: RCT
Calculate $C_5 = \prod_{i=1}^{t} \left(C_5^{(i)}\right)^{\lambda_i}$, $\lambda_i = \prod_{j=1, j\neq i}^{t} \frac{0-j}{i-j} \mod p$;
return $RCT = (C_1, C_5)$;

5.4.7. Decryption Phase

The decryption phase consists of *DO* decrypting the ciphertext *CT* and *DU* decrypting the re-encrypted ciphertext *RCT*.

DO retrieves *CT* from the CS. Using *CT*, identity key DK_{DO}, and version tag *ver*, DO executes algorithm *Dec* to obtain plaintext *m*. Algorithm 7 describes the particular procedure.

Algorithm 7 Dec

Input: CT, DK_DO, ver
Output: m
Calculate $m = C_2 / e(C_1, DK_{DO} + ver)$;
return m;

After receiving *RCT*, *DU* performs algorithm *ReDec* using the *RCT*, the identity identifier DID_{DO}, DID_{DU}, and identity key DK_{DU} to obtain the plaintext *m*. Algorithm 8 describes the particular procedure.

Algorithm 8 ReDec

Input: $RCT, DID_{DO}, DID_{DU}, DK_{DU}$
Output: m
Calculate $H' = H_4(e(DK_{DU}, H_1(DID_{DO})), DID_{DO}, DID_{DU})$;
Calculate $m = C_5 / e(C_1, H')$;
return m;

6. Experiments and Analysis

This section will evaluate the proposed scheme through security proof, security analysis, and performance analysis.

6.1. Security Proof

Theorem 1. *The scheme is IND-CCA2 secure under the DBDH assumption.*

Proof. Let $P, aP, bP, cP \in G$ and $Q \in G_T$ be given. The values $a, b, c \in \mathbb{Z}_p^*$ are unknown. The $DBDH$ problem is to decide whether $Q = e(P,P)^{abc}$. Suppose there is a challenger C and a adversary $\mathcal{A}$. $\mathcal{A}$ wins the IND-CCA2 game with a non-negligible probability within polynomial time. The interaction process between C and A is as follows:

- **Initialization**. C operating system initializes the algorithm $Setup(1^\lambda) \rightarrow PP$, sending PP to $\mathcal{A}$. Simultaneously, C aggregates the committee's complete private key s to respond to $\mathcal{A}$'s queries. C initializes several empty lists to record $\mathcal{A}$'s behavior.
- **Phase-1**: The adversary $\mathcal{A}$ makes the following adaptive queries. C responds to $\mathcal{A}$'s queries and records each query in the list.

(1) *H_1 query Q_{H_1}*: C maintains list $L_1 = \{DID_i, h_{1i}\}(i = 1, \ldots, n_1)$. When $\mathcal{A}$ inputs DID_i for a query, if the tuple (DID_i, h_{1i}) already exists in L_1, C returns the matching h_{1i}. Otherwise, randomly generates $h_{1i} \in G$, returns it, and updates the list L_1.

(2) *H_2 query Q_{H_2}*: C maintains list $L_2 = \{ver_i, h_{2i}\}(i = 1, \ldots, n_2)$. When $\mathcal{A}$ inputs ver_i for a query, if the tuple (ver_i, h_{2i}) already exists in L_2, C returns the matching h_{2i}. Otherwise, C randomly generates $h_{2i} \in \{0,1\}^*$, returns it, and updates the list L_2.

(3) *H_3 query Q_{H_3}*: C maintains list $L_3 = \{C_{1i}, C_{2i}, C_{vi}, h_{3i}\}, (i = 1, \ldots, n_3)$. When $\mathcal{A}$ inputs (C_{1i}, C_{2i}, C_{vi}) for a query, if the tuple $(C_{1i}, C_{2i}, C_{vi}, h_{3i})$ already exists in L_3, C returns the matching h_{3i}. Otherwise, C randomly generates $h_{3i} \in G$, return it, and updates the list L_3.

(4) *Identity key query Q_{sk}*: C maintains list $L_{sk} = \{DID_i, h_{1i}, DK_{DID_i}, tag_i\}$, where $tag_i \in \{0,1\}$. When $\mathcal{A}$ inputs DID_i to perform a query, C consults list L_{sk}. If $tag = 0$, C returns $\perp$ and terminates the game. Otherwise, C executes Q_{H_1} and queries list L_1 to obtain the tuple (DID_i, h_{1i}), calculates and returns $DK_{DID_i} = s \cdot H_1(DID_i) = s \cdot h_{1i}$ to $\mathcal{A}$.

(5) *H_4 query Q_{H_4}*: C maintains list $L_4 = \{DID_i, DID_j, h_{4ij}\}(i = 1, \ldots, n_4 . j = 1, \ldots, n_5)$. When $\mathcal{A}$ queries with (DID_i, DID_j), if tuple (DID_i, DID_j, h_{4ij}) already exists in L_4, C returns the corresponding h_{4ij}. Otherwise, C executes Q_{sk} and queries list L_{sk} to obtain the tuple DK_{DID_i}, calculates and returns $h_{4ij} = H_4(e(DK_{DID_i}, H_1(DID_j)), DID_i, DID_j)$ to $\mathcal{A}$.

(6) *Re-encryption key query Q_{rk}*: C maintains list $L_{rk} = \{DID_i, DID_j, ver, tag_i\}$. When $\mathcal{A}$ inputs (DID_i, DID_j, ver) for query, C consults list L_{sk}. If $tag = 0$, C returns $\perp$ and terminates the game. Otherwise, C queries list L_{sk} and L_4 based on DID_i and DID_j, computes the result $rk_{i \rightarrow j} = h_{4ij} - DK_{DID_i} - ver$, and returns $rk_{i \rightarrow j}$ to $\mathcal{A}$.

(7) *Re-encryption query Q_{rec}*: When $\mathcal{A}$ initiates a ciphertext re-encryption request, it queries using input $(DID_i, DID_j, CT_{DID_i})$. C queries list L_{rk} for the $rk_{i \rightarrow j}$ based on $\mathcal{A}$'s input, re-encrypts CT_{DID_i} using the $rk_{i \rightarrow j}$, computes, and outputs the re-encrypted ciphertext $RCT_{DID_j} = ReEnc(CT_{DID_i}, rk_{i \rightarrow j})$.

(8) *Decryption query Q_{dec}*: C responds as follows based on different scenarios:

a. If $\mathcal{A}$ performs a decryption query on the original ciphertext CT_{DID_i} for DID_i, where $CT_{DID_i} = (C_v, C_1, C_2, C_3)$. C executes a Q_{sk} query on the lookup list L_{sk} to retrieve the corresponding identity key DK_{DID_i}, calculates $m = C_2 / e(C_1, DK_{DID_i} + ver)$, and returns it to $\mathcal{A}$.

b. If $\mathcal{A}$ performs a decryption query on the re-encrypted ciphertext RCT_{DID_j} of DID_j, where $RCT_{DID_j} = (C_1, C_5)$. C queries the corresponding value h_{4ij} in list L_4, computes $m = C_5 / e(C_1, h_{4ij})$, and returns it to $\mathcal{A}$.

- **Challenge Phase**: After $\mathcal{A}$ concludes the initial query phase, $\mathcal{A}$ selects an unchallenged user DID^* that does not belong to L_{sk}, and outputs two equal-length plaintext messages m_0 and m_1 to C. C randomly selects a bit $b \in \{0,1\}$ and computes the challenge ciphertext $CT_{DID^*} = Enc(PK, DID^*, m_b, ver)$, where ver denotes the ciphertext version tag and PK represents the committee's public key.
- **Phase-2**: Queries of the same type may be issued by $\mathcal{A}$ to C after the challenge phase. However, to ensure the indistinguishability of challenge ciphertexts [27], the following constraints must be satisfied:

(1) If (DID_i, CT_i) is the ciphertext corresponding to the challenge ciphertext, then $\mathcal{A}$ cannot initiate a decryption request for user DID_i;
(2) If DID_i is the challenged user identity, such as $DID_i = DID^*$, then $\mathcal{A}$ cannot initiate a private key query request for DID_i;
(3) If $DID_i = DID^*$ and $\mathcal{A}$ has already performed a private key query on DID_j, then $\mathcal{A}$ cannot initiate a re-encryption key generation request for (DID_i, DID_j).
(4) If DID_i is the challenged user and $\mathcal{A}$ has already performed a private key query on DID_j, a re-encryption request cannot be issued for (DID_i, DID_j).

- **Guessing**: Finally, $\mathcal{A}$ outputs a guessed bit $b' \in \{0,1\}$. If $b' = b$, then $\mathcal{A}$ wins. C computes $D = C_2^* / m_d$ and outputs (P, b^*, C_1^*, D) as a $DBDH$ tuple.

When $\mathcal{A}$ executes Q_{sk}, Q_{rk}, and Q_{rec}, if the corresponding tag $tag_i = 1$, it indicates that challenger C has not terminated the game. Suppose $\mathcal{A}$ issues q_{sk} private key queries, q_{rk} re-encryption key generation queries, and q_{rec} re-encryption queries. For each query, the probability that the game does not terminate is $Pr[tag = 1] = 1 - \delta$. Therefore, the probability that C successfully simulates the entire game process (the game does not terminate) is $(1-\delta)^{q_{sk}+q_{rk}+q_{rec}}$. Assume that the probability of $\mathcal{A}$ winning the challenge is $\frac{1}{2} + \epsilon$, and the probability that C correctly guesses the challenger's identity is δ. Accordingly, the advantage of C against the $DBDH$ problem is stated as follows:

$$Adv_{IND-CCA2} \approx \delta\epsilon(1-\delta)^{q_{sk}+q_{rk}+q_{rec}} \geq \frac{\epsilon}{e^{q_{sk}+q_{rk}+q_{rec}+1}} \tag{4}$$

Since the advantage ϵ that adversary $\mathcal{A}$ gains by winning the $IND-CCA2$ game is non-negligible, the challenger C can solve the $DBDH$ problem with non-negligible probability, thereby contradicting the $DBDH$ assumption. □

In summary, this scheme satisfies $IND-CCA2$ semantic security under adversary $\mathcal{A}$'s attacks.

6.2. Security Analysis

(1) Correctness:
We primarily analyzed the user identity key DK, the re-encrypted ciphertext aggregation C_5, and the decryption algorithm. The calculation method for λ_i is shown in (2). The computational analysis is as follows.
The identity key DK:

$$\begin{aligned} DK &= \sum_{i=1}^{t}(DK_i \cdot \lambda_i) \\ &= \sum_{i=1}^{t}(s_i \cdot H_1(DID) \cdot \lambda_i) \\ &= H_1(DID) \cdot \sum_{i=1}^{t}(s_i \cdot \lambda_i) \\ &= s \cdot H_1(DID) \end{aligned}$$

The re-encrypted ciphertext aggregation C_5:

$$\begin{aligned} C_5 &= \prod_{i=1}^{t} (C_5^{(i)})^{\lambda_i} \\ &= \prod_{i=1}^{t} \left(C_2 \cdot e\left(C_1, rk_{DO \to DU}^{(i)} \right) \right)^{\lambda_i} \\ &= C_2 \cdot e(C_1, rk_{DO \to DU}) \\ &= C_2 \cdot e(C_1, H - DK_{DO} - ver) \\ &= m \cdot e(C_1, DK_{DO}) \cdot e(C_1, ver) \cdot e(C_1, H) \cdot e(C_1, DK_{DO})^{-1} \cdot e(C_1, ver)^{-1} \\ &= m \cdot e(C_1, H) \end{aligned}$$

Algorithm *Dec*:

$$\begin{aligned} m &= \frac{C_2}{e(C_1, DK_{DO} + ver)} \\ &= \frac{m \cdot e(PK, H_1(DID_{DO}))^r \cdot e(P, ver)^r}{e(C_1, DK_{DO}) \cdot e(C_1, ver)} \\ &= \frac{m \cdot e(C_1, DK_{DO}) \cdot e(C_1, ver)}{e(C_1, DK_{DO}) \cdot e(C_1, ver)} = m \end{aligned}$$

Algorithm *ReDec*:
because

$$e(DK_{DO}, H_1(DID_{DU})) = e(DK_{DU}, H_1(DID_{DO}))$$

therefore

$$\begin{aligned} H &= H_4(e(DK_{DO}, H_1(DID_{DU})), DID_{DO}, DID_{DU}) \\ &= H_4(e(DK_{DU}, H_1(DID_{DO})), DID_{DO}, DID_{DU}) \\ &= H' \end{aligned}$$

then

$$\begin{aligned} m &= \frac{C_5}{e(C_1, H')} \\ &= \frac{m \cdot e(C_1, H)}{e(C_1, H')} = m \end{aligned}$$

(2) Confidentiality: The plaintext is always stored and transmitted in ciphertext form. The ciphertext CT can only be decrypted by the DO using their private key sk_{DO}. The re-encrypted ciphertext RCT is generated by the proxy node using its own re-encryption key through threshold proxy re-encryption. The DU can decrypt it using their private key sk_{DU}. Since the private keys are held secretly by the users, other entities cannot decrypt the data and thus cannot obtain the plaintext m.

(3) Integrity: In our proposed scheme, the CT can be verified for tampering by calculating whether $e(C_1, C_3')$ is equal to $e(P, C_3)$, thereby ensuring CT's integrity. The re-encrypted ciphertext RCT verifies both correctness and integrity through the zero-knowledge proof π.

(4) Traceability: Each ciphertext in the CS corresponds to the DID of its data owner, and DO must register with the committee based on the VP prior to uploading data.

Therefore, LN can trace the *pk* of the uploader based on the DID-corresponding document on the blockchain.

6.3. Performance Analysis

The experimental environment for this scheme is based on a Windows 11 PC equipped with an 11th Gen Intel® Core™ i7-11800H @ 2.30 GHz processor and 8 GB of memory. The Ubuntu 20.04 virtual machine was deployed on this host, and the Charm library of Python 3.8 was used as the programming basis. Charm is a Python library specifically dedicated to pairing cryptography. In the experimental tests of this section, the elliptic curve we adopted is SS512, and the sizes of each element in groups G and G_T are 90 bytes and 174 bytes, respectively. This paper mainly compares the computational costs of each scheme, while the communication costs are only used for theoretical analysis and comparison.

Table 2 compares the differences in computational overhead and communication overhead between the scheme proposed in this paper and schemes [24,27,34]. Let $|G|$ and $|G_T|$ denote the element lengths of G and G_T, respectively. Let T_p denote a bilinear pair operation and T_e denote a point multiplication operation in G. c represents lightweight operations that do not involve exponential calculations or bilinear pairing operations, and this overhead can be disregarded. For the CT, slightly higher communication overhead is observed compared with [24,34]. This is due to the introduction of an extra field for ciphertext integrity and correctness verification. Due to the added key commitment in our scheme for correctness, non-modifiability, and zero-knowledge support, the communication cost of Rk is higher than that of other schemes. In contrast, lower communication overhead is achieved for the RCT. Overall, our scheme adds ciphertext integrity verification and key source verification. Despite certain communication costs, it significantly enhances security and auditability.

Table 2. Comparison of scheme performance.

Schemes	Communication Overhead			Computation Overhead			
	CT	Rk	RCT	Enc	Dec	ReEnc	ReDec
[24]	$\|G\| + \|G_T\|$	$\|G\|$	$2\|G_T\|$	$T_e + T_p$	T_p	T_p	c
[27]	$2\|G\| + \|G_T\|$	$\|G\|$	$3\|G_T\|$	$2T_e + T_p$	T_p	$2T_p$	c
[34]	$\|G\| + \|G_T\|$	$\|G\|$	$2\|G_T\|$	$T_e + T_p$	T_p	$T_e + 3T_p$*	c
Our proposed method	$2\|G\| + \|G_T\|$	$2\|G\| + 2\|G_T\|$	$\|G\| + \|G_T\|$	$2T_e + 2T_p$	T_p	T_p	$2T_p$

* The re-encryption overhead in this scheme is divided into the authorization computation overhead $T_e + 2T_p$ for the data owner and the computation overhead T_p for the proxy node.

In the encryption phase, a computational overhead of $2T_e + 2T_P$ is incurred by the proposed scheme. This overhead is slightly higher than that of the other three schemes. The increase is caused by the addition of ciphertext integrity verification. As a result, ciphertext security is enhanced. For re-encryption key generation, T_P is required, compared with T_e in the other schemes. In the re-encryption phase, a overhead of T_P is incurred by each proxy node. This is equal to the schemes in [24], but lower than [27,34]. In the decryption phase of the re-encrypted ciphertext phase, the overhead reaches $2T_P$, exceeding the other schemes. Overall, a trade-off between computational efficiency and security is achieved.

Figure 4 presents the time overhead of the algorithms ProofGen, ProofVer, and ShareComb in the proposed scheme. The results are shown for different numbers of proxy nodes n. The aggregation time of re-encrypted ciphertext stays nearly constant when the number of nodes grows and is always maintained at a low level, reflecting strong scalability. When n = 100, the time overhead is 6.35 ms. In contrast, zero-knowledge proof generation and verification costs grow roughly linearly as nodes increase. However, they have a smooth growth trend and good verification efficiency, which can still satisfy the system's performance requirements in large-scale user scenarios.

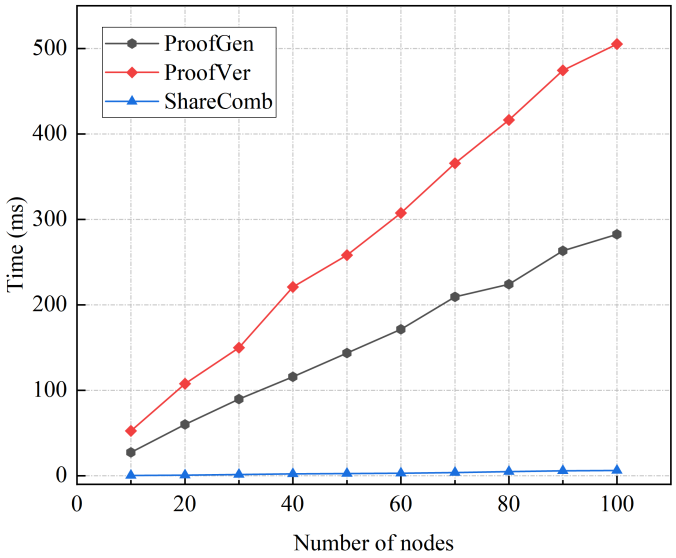

Figure 4. Computation overhead of *ProofGen*, *ProofVer*, and *ShareComb*.

Figure 5 shows the time overhead performance of the data re-encryption and aggregation phases based on the secret sharing mechanism in our scheme under different parameter combinations (t, n). Upon receiving the ciphertext re-encryption request, n agent nodes in the committee perform re-encryption operation on the ciphertext, respectively, and complete the aggregation operation after collecting t copies of re-encrypted ciphertexts, thus generating the complete re-encrypted ciphertext. As shown in the figure, the computation time of the re-encryption phase tends to grow as the number of committee members n increases. In addition, when the threshold value t increases, the overall time overhead also increases slightly, mainly in the aggregation phase of the re-encrypted ciphertext fragmentation. Therefore, in practice, parameters t and n should be carefully chosen. The choice depends on security and performance requirements. This controls overhead while improving fault tolerance and system security.

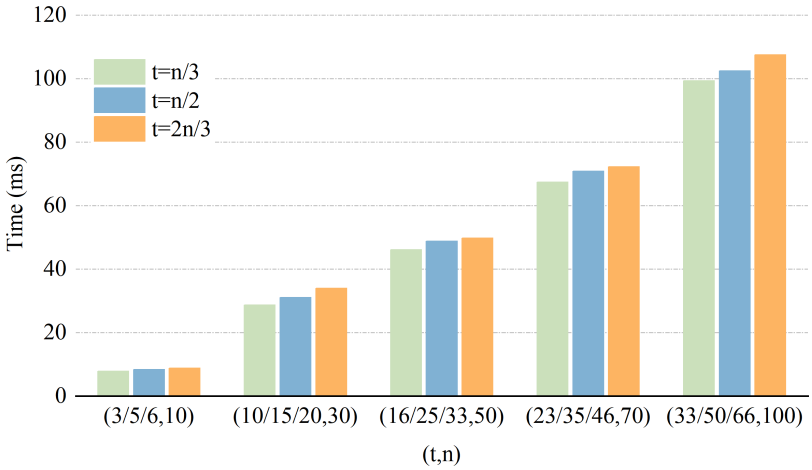

Figure 5. (t, n) Threshold secret sharing scheme time overhead.

Figure 6 compares the time overhead of *Enc*, *ReEnc*, *Dec*, and *ReDec*. The comparison includes our scheme and schemes [24,27,34]. Among them, the threshold is set as $t = n/2 + 1$, as can be seen from the figure, since only the computational overhead of a single node is considered, the computational overhead of each node does not increase with the growth of users and remains generally stable within a small constant range.

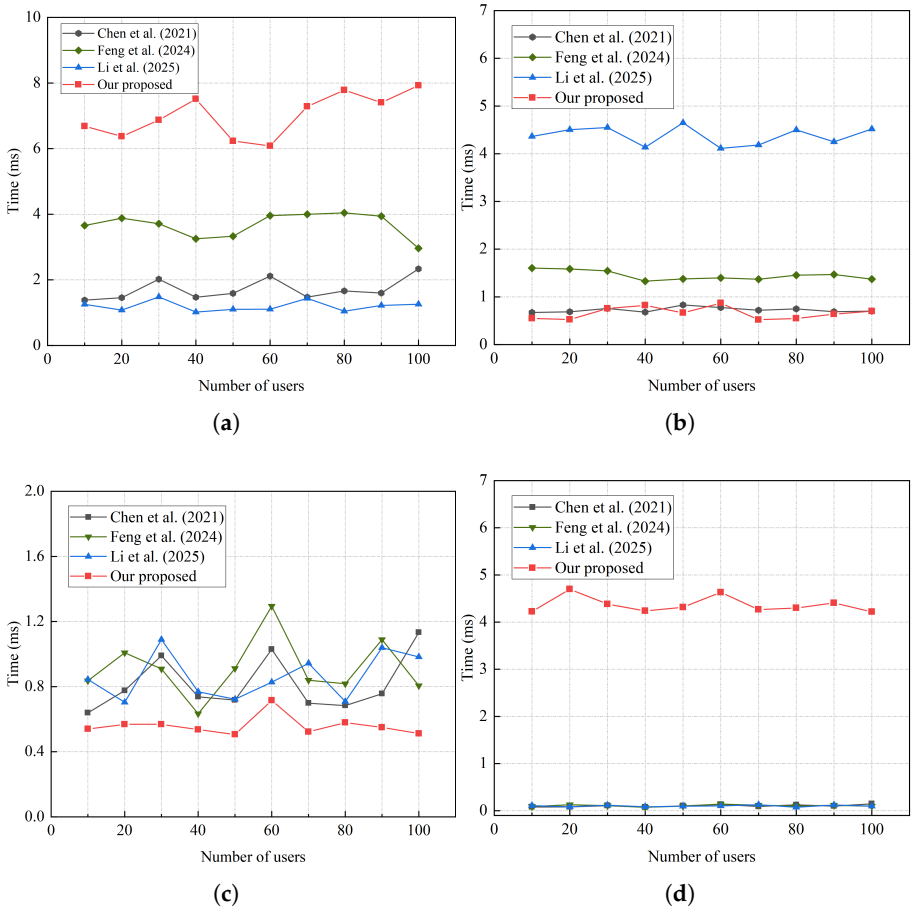

Figure 6. Comparison of computation overhead [24,27,34]: (**a**) Enc. (**b**) ReEnc. (**c**) Dec. (**d**) ReDec.

Figure 7 shows the time overhead comparison of the re-encryption key generation phase. Among them, the threshold is set as $t = n/2 + 1$. With more proxy nodes, the number of key fragments increases. As a result, the computational overhead also increases. All four schemes exhibit linear growth as user numbers increase, demonstrating good scalability. As shown in Figure 8. To ensure re-encryption key correctness and immutability, our scheme adds verification fields during generation, incurring slightly higher overhead than the other three schemes. This trade-off enables verifiability and malicious node resilience without affecting linear efficiency. Experiments show it maintains stable performance while enhancing security, with promising prospects.

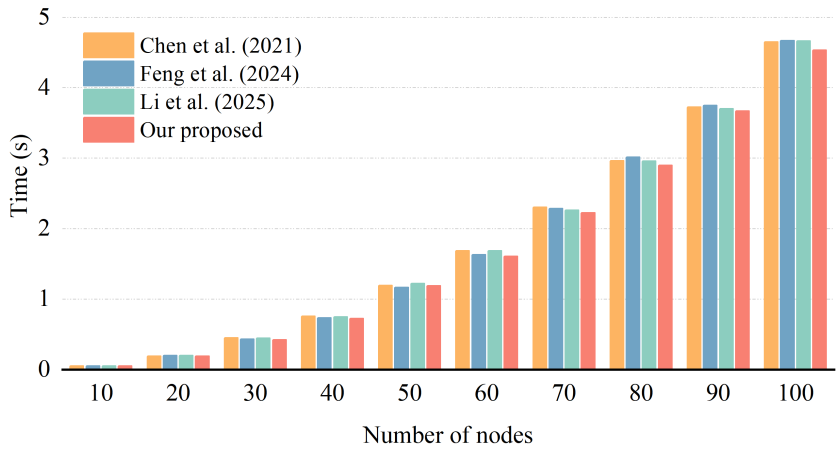

Figure 7. Comparison of computation overhead for *RKGen* without verification [24,27,34].

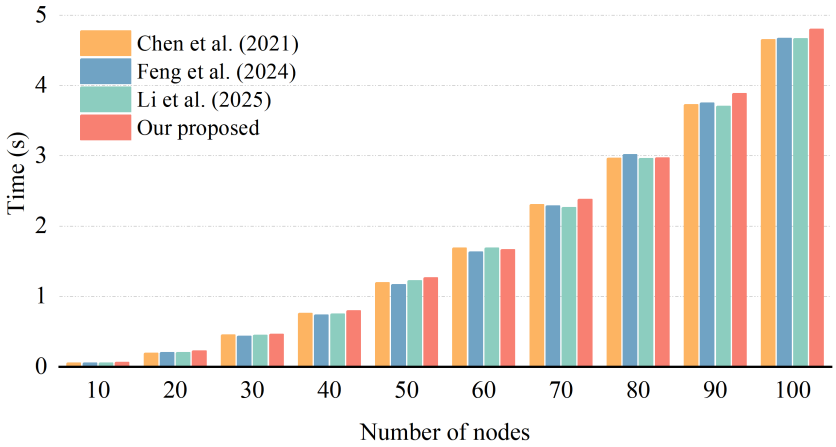

Figure 8. Comparison of computation overhead for *RKGen* with verification [24,27,34].

7. Conclusions

Aiming at the requirements of secure and efficient data sharing in cross-domain environment, this paper proposes a decentralized data sharing scheme based on TPRE. The scheme adopts the committee node governance mode to ensure data confidentiality, realize fine-grained access control and secure ciphertext conversion, and improve the robustness of the system. This effectively resists single-point failure and node collusion attacks. The scheme also uses DID technology to realize decentralized cross-domain identity authentication and authorization. In addition, combined with NIZKPs, the correctness of re-encryption operation is publicly verifiable, which ensures the integrity of ciphertext and the credibility of the process. This scheme achieves IND-CCA2 security and also provides traceability and privacy protection. Performance experiments show that this scheme has good performance simultaneously and is expected to be applied in scenarios such as medical care and the industrial Internet of Things, enhancing the security of cross-domain data sharing.

Author Contributions: Conceptualization, B.W.; methodology, B.W. and Q.Y.; software, J.Z.; validation, Q.Y.; formal analysis, B.W.; investigation, B.W.; resources, Q.Y.; data curation, S.J.; writing—original draft preparation, B.W.; writing—review and editing, Q.Y.; visualization, J.Z.; supervision, Q.Y.; project administration, S.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data generated or analyzed during this study are available on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cooper, R.B. Information technology development creativity: A case study of attempted radical change. *Mis Q.* **2000**, *24*, 245–276. [CrossRef]
2. Yaqoob, I.; Salah, K.; Jayaraman, R.; Al-Hammadi, Y. Blockchain for healthcare data management: Opportunities, challenges, and future recommendations. *Neural Comput. Appl.* **2022**, *34*, 11475–11490. [CrossRef]
3. Remolina Leon, N. Open finance: Regulatory challenges of the evolution of data sharing arrangements in the financial sector. *Bank. Financ. Law Rev.* **2023**, *40*, 35. [CrossRef]

4. Badirova, A.; Dabbaghi, S.; Moghaddam, F.F.; Wieder, P.; Yahyapour, R. A survey on identity and access management for cross-domain dynamic users: Issues, solutions, and challenges. *IEEE Access* **2023**, *11*, 61660–61679. [CrossRef]
5. Nguyen, T.L.; Nguyen, L.; Hoang, T.; Bandara, D.; Wang, Q.; Lu, Q.; Xu, X.; Zhu, L.; Chen, S. Blockchain-empowered trustworthy data sharing: Fundamentals, applications, and challenges. *ACM Comput. Surv.* **2025**, *57*, 1–36. [CrossRef]
6. Wang, M.; Rui, L.; Yang, Y.; Gao, Z.; Chen, X. A blockchain-based multi-CA cross-domain authentication scheme in decentralized autonomous network. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 2664–2676. [CrossRef]
7. Ali, G.; Ahmad, N.; Cao, Y.; Khan, S.; Cruickshank, H.; Qazi, E.A.; Ali, A. xDBAuth: Blockchain based cross domain authentication and authorization framework for Internet of Things. *IEEE Access* **2020**, *8*, 58800–58816. [CrossRef]
8. Döttling, N.; Nishimaki, R. Universal proxy re-encryption. In *Proceedings of the IACR International Conference on Public-Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 512–542.
9. Keshta, I.; Aoudni, Y.; Sandhu, M.; Singh, A.; Xalikovich, P.A.; Rizwan, A.; Soni, M.; Lalar, S. Blockchain aware proxy re-encryption algorithm-based data sharing scheme. *Phys. Commun.* **2023**, *58*, 102048. [CrossRef]
10. Raghav, A.N.; Verma, K.; Venkatesan, S.; Verma, S. Proactive threshold-proxy re-encryption scheme for secure data sharing on cloud: Raghav et al. *J. Supercomput.* **2023**, *79*, 14117–14145. [CrossRef]
11. Liu, J.; Liu, Z.; Sun, C.; Zhuang, J. A data transmission approach based on ant colony optimization and threshold proxy re-encryption in WSNs. *J. Artif. Intell. Technol.* **2022**, *2*, 23–31. [CrossRef]
12. Reed, D.; Sporny, M.; Longley, D.; Allen, C.; Grant, R.; Sabadello, M.; Holt, J. Decentralized identifiers (dids) v1. 0. *Draft Community Group Rep.* **2020**. Available online: https://www.w3.org/TR/did-1.0/ (accessed on 8 January 2026).
13. Golosova, J.; Romanovs, A. The advantages and disadvantages of the blockchain technology. In *Proceedings of the 2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*; IEEE: New York, NY, USA, 2018; pp. 1–6.
14. Green, M.; Ateniese, G. Identity-based proxy re-encryption. In *Proceedings of the International Conference on Applied Cryptography and Network Security*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 288–306.
15. Yao, S.; Dayot, R.V.J.; Ra, I.H.; Xu, L.; Mei, Z.; Shi, J. An identity-based proxy re-encryption scheme with single-hop conditional delegation and multi-hop ciphertext evolution for secure cloud data sharing. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3833–3848. [CrossRef]
16. Luo, S.; Hu, J.; Chen, Z. Ciphertext policy attribute-based proxy re-encryption. In *Proceedings of the International Conference on Information and Communications Security*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 401–415.
17. Ge, C.; Susilo, W.; Baek, J.; Liu, Z.; Xia, J.; Fang, L. A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2907–2919. [CrossRef]
18. Pei, H.; Yang, P.; Li, W.; Du, M.; Hu, Z. Proxy re-encryption for secure data sharing with blockchain in internet of medical things. *Comput. Netw.* **2024**, *245*, 110373. [CrossRef]
19. Agyekum, K.O.B.O.; Xia, Q.; Sifah, E.B.; Cobblah, C.N.A.; Xia, H.; Gao, J. A proxy re-encryption approach to secure data sharing in the internet of things based on blockchain. *IEEE Syst. J.* **2021**, *16*, 1685–1696. [CrossRef]
20. Maiti, S.; Misra, S. P2B: Privacy preserving identity-based broadcast proxy re-encryption. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5610–5617. [CrossRef]
21. Ge, C.; Susilo, W.; Liu, Z.; Baek, J.; Luo, X.; Fang, L. Attribute-based proxy re-encryption with direct revocation mechanism for data sharing in clouds. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 949–960. [CrossRef]
22. Doshi, N. An enhanced approach for CP-ABE with proxy re-encryption in IoT paradigm. *Jordanian J. Comput. Inf. Technol.* **2022**, *8*, 232–241. [CrossRef]
23. Lynch, G.S. *Single Point of Failure*; Wiley Online Library: Hoboken, NJ, USA, 2009.
24. Chen, Y.; Hu, B.; Yu, H.; Duan, Z.; Huang, J. A threshold proxy re-encryption scheme for secure IoT data sharing based on blockchain. *Electronics* **2021**, *10*, 2359. [CrossRef]
25. Chen, X.; Liu, Y.; Li, Y.; Lin, C. Threshold proxy re-encryption and its application in blockchain. In *Proceedings of the International Conference on Cloud Computing and Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 16–25.
26. Zhao, F.; Weng, J.; Xie, W.; Li, M.; Weng, J. HRA-secure attribute-based threshold proxy re-encryption from lattices. *Inf. Sci.* **2024**, *655*, 119900. [CrossRef]
27. Feng, J.; Li, Y.; Wang, T.; Liu, S. A Certificateless Threshold Proxy Re-Encrypted Data Sharing Scheme With Cloud-Chain Collaboration in Industrial Internet Environments. *IEEE Internet Things J.* **2024**, *11*, 33247–33268. [CrossRef]
28. Beimel, A. Secure Schemes for Secret Sharing and Key Distribution. Ph.D. Thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
29. Fiat, A.; Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1986; pp. 186–194.
30. Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In Proceedings of the 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), Los Angeles, CA, USA, 12–14 October 1987; pp. 427–438.
31. Frey, D. Recent research on selective exposure to information. *Adv. Exp. Soc. Psychol.* **1986**, *19*, 41–80.

32. Boneh, D.; Lynn, B.; Shacham, H. Short signatures from the Weil pairing. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 514–532.
33. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
34. Li, W.; Xie, Z.; Liu, S.; Zhou, Y.; Zhang, H. Controllable Access Control in Permissioned Blockchains via Controllable Threshold Proxy Re-Encryption. *IEEE Trans. Inf. Forensics Secur.* **2025**, *20*, 9625–9640. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.