Publicly Verifiable Threshold Proxy Re-encryption and Its Application in Data Rights Confirmation

Tao Liu, Liang Zhang*, Haibin Kan and Jiheng Zhang

***Abstract*—Proxy re-encryption (PRE) has been regarded as an effective cryptographic primitive in data sharing systems with distributed proxies. However, no literature considers the honesty of data owners, which is critical in the age of big data. In this paper, we fill the gap by introducing a new proxy re-encryption scheme, called publicly verifiable threshold PRE (PVTPRE). Briefly speaking, we innovatively apply a slightly modified publicly verifiable secret sharing (PVSS) scheme to distribute the re-encryption keys to multiple proxies. Consequently, we achieve publicly verifiability of data owners non-interactively. Then, the correctness of data users in decryption and public verifiability of proxies in re-encryption are guaranteed seamlessly through execution of the PVSS reconstruction algorithms. We further prove that PVTPRE satisfies *IND-CPA* security. Besides, we put forward a privacy-preserving data rights confirmation framework by providing clear principles for data ownership and usage, based on the PVTPRE scheme and blockchain. Blockchain plays the role of data bank and smart contract engine, providing reliable storage and verification for all framework. To our knowledge, we are the first to systematically investigate data rights confirmation considering privacy as well as public verifiability, addressing the growing need for robust mechanisms to protect data rights and ensure transparency. Finally, we conduct comprehensive experiments to illustrate the correctness, feasibility and effectiveness. The experimental results show that our PVTPRE outperforms other PREs in many aspects.**

***Index Terms*—Threshold proxy re-encryption, Publicly verifiable, Data rights confirmation, Blockchain, PVSS**

I. Introduction

Data sharing [1]–[3] plays a vital role in fostering collaboration among businesses, researchers, and developers, driving new discoveries, technologies, and solutions. Proxy re-encryption (PRE) [4]–[6] is a significant extension of public key encryption that enables a proxy to convert delegator's ciphertext into a new ciphertext for a delegatee. This powerful cryptographic primitive finds extensive applications in secure data sharing, including in fields such as medical data collaboration [7]–[9], the Internet of Things [10], cloud computing [11], [12] and data trading [13].

Tao Liu and Liang Zhang are with School of Cyberspace Security (School of Cryptology), Hainan University, Haikou, 570228, China. Liang Zhang is also with Department of Industrial Engineering and Decision Analytics, Hong Kong University of Science and Technology. (e-mail: jb8181632@gmail.com; briliasm@gmail.com)

Haibin Kan is with School of Computer Science, Fudan University, Shanghai 200433, China, Shanghai Engineering Research Center of Blockchain, Shanghai 200433, China and Yiwu Research Institute of Fudan University, Yiwu City 322000, China. (e-mail: hbkan@fudan.edu.cn)

Jiheng Zhang is with Department of Industrial Engineering and Decision Analytics, Hong Kong University of Science and Technology. (e-mail: jiheng@ust.hk)

*Corresponding author

In PRE schemes, the honesty of the proxy plays a crucial role in practical applications. Luo et al. [11] highlight that in traditional PRE schemes, the proxy may disclose the re-encryption key, leading to potential abuse of the re-encryption key. To mitigate this issue, they propose a public traceable PRE scheme. However, the traceability algorithm requires the re-encryption key as input, which must be made publicly available during the tracing process.

Moreover, ensuring the correctness of the re-encrypted ciphertext produced by the proxy is also critical in PRE schemes. In other words, detecting malicious activities by the proxy during the re-encryption process is a key consideration. Ohata et al. [5] are the first to address this issue and introduce the concept of "re-encryption verifiability". Specifically, they propose a verifiable PRE scheme based on the re-splittable threshold public key encryption scheme [14], which achieves re-encryption verifiability. Some other PRE schemes [15] achieve verifiability by relying on a "claim" algorithm, which requires disclosing the plaintext and the re-encryption key. This approach necessitates an additional communication round to determine whether the proxy is faulty.

Many PRE schemes rely on a single semi-trusted proxy to perform the re-encryption process. However, this approach is vulnerable to a single point of failure and is not suitable for distributed systems. To address this limitation, multi-proxy or threshold re-encryption schemes have been introduced and extensively studied [6], [16]–[18]. In (t, n)-threshold PRE schemes, the re-encrypted ciphertext is generated by collaboration of at least t out of n proxies. In terms of computational complexity, many threshold PRE schemes [16]–[18] require bilinear pairings.

Furthermore, none of previous literatures considers the honesty of the delegator (i.e., the data owner). Previous PRE schemes [5], [6], [10]–[12], [15], [19]–[21] rely on transmitting re-encryption keys through private and secure channels. In real-world scenarios, proxies must ensure that the received keys can be correctly used for re-encryption, and the delegatee (i.e., the data user) needs to verify that the decrypted data matches their expectations. In the event of a data sharing failure, it is essential to determine which entity is responsible for the issue. Therefore, both the encryption and re-encryption keys should be verifiable or accountable.

We address the aforementioned challenges by proposing a publicly verifiable threshold proxy re-encryption (PVTPRE) scheme based on publicly verifiable secret sharing (PVSS) [22]–[24]. PVSS is a variant of secret sharing scheme [25] that enables anyone to verify the correctness of the shares publicly. By designating the delegator as the

PVSS dealer and the proxies as PVSS shareholders, we can elegantly construct the PVTPRE scheme. Consequently, re-encryption keys can be sent using public channels and whether the delegator has generated re-encryption keys can be checked publicly. The underlying PVSS protocol we employ is DHPVSS [22]. Compared to other PVSS schemes, DHPVSS utilizes the dealer's private key when generating the encrypted secret shares, a feature that is crucial for the security of PRE schemes.

Based on the proposed PVTPRE and blockchain [26], we present a data rights confirmation framework. Due to its decentralization, security, transparency, and immutability, blockchain serves as both the data bank and the smart contract engine. The data bank is designed for data storage, management, and governance. Specifically, we implement the data bank by integrating IPFS with Ethereum. On one hand, the outputs of PVTPRE algorithms are uploaded to the blockchain and automatically verified by smart contracts. On the other hand, the ownership, control, and usage rights of data are recorded on the blockchain to ensure the transparency and traceability of data rights confirmation. The proposed framework also accounts for potential threats posed by all participating entities, ensuring its robustness in practical applications.

Summarily, our contributions are as follows.

- By innovatively applying the PVSS scheme to construct threshold PRE, we introduce the concept of PVTPRE and provide a concrete construction using the DHPVSS scheme. Additionally, we formally prove the security requirements of the scheme.
- The PVTPRE not only supports multiple proxies with threshold tolerance, but also guarantees verifiability of the proxies when performing re-encryption.
- For the first time, the honesty of the delegator (data owner) is incorporated into our PRE scheme. Specifically, the delegator in our PVTPRE is held accountable for generating the re-encryption keys for the proxies.
- We integrate the PVTPRE and blockchain into a data rights confirmation framework, enabling data management in a transparent yet verifiable manner. Furthermore, as a complement to PVTPRE, we provide a dispute resolution method, allowing the data owner to be held accountable during data encryption.
- We provide a comprehensive comparison with related works in terms of complexity and evaluate the computational cost based on the Ethereum-compatible BN128 curve. Experimental results demonstrate that the pairing-free PVTPRE scheme achieves plausible performance.

II. Related Works

PRE is proposed by Blaze et al. [4] and has been extensively studied by the academic community for decades. Numerous PRE schemes with varying properties have been proposed to meet the diverse demands of different applications. In the following section, we focus on describing the characteristics of recent PRE schemes, with particular emphasis on their verifiability.

Traceable proxy re-encryption. Given the risk of re-encryption key abuse during the delegation process in PRE—whether due to malicious behavior by the proxy or collusion between the proxy and the delegatee—the concept of traceable proxy re-encryption [27] scheme has been introduced. While this primitive does not completely eliminate the risk of re-encryption key abuse, it provides a deterrent to potential abusers. Guo et al. [28] propose a generic construction of traceable PRE, embedding collusion-resistant codes into the re-encryption key to enable traceability of any abuse. Recently, Luo et al. [11] introduced a public trace-and-revoke PRE scheme, claiming that their approach can identify abusers of the re-encryption key and revoke their decryption capabilities. This work opens a new avenue of research for addressing the problem of re-encryption key abuse.

Autonomous path proxy re-encryption. Blaze et al. [4] classify proxy re-encryption (PRE) schemes based on the number of allowed transformations. If a PRE scheme permits the ciphertext to be transformed multiple times (e.g., from Alice to Bob, from Bob to Carol, and so on), it is considered multi-hop; otherwise, it is single-hop. In a multi-hop scheme, only the first proxy is determined by the delegator. To ensure that the entire proxy process remains under the delegator's control, the concept of autonomous path proxy re-encryption (AP-PRE) [19] is introduced. Lin et al. [29] observe that the access policy used in AP-PRE schemes, which allows the delegator to predefine the entire decryption path, supports only linked paths and does not allow proxies in the path to generate authorized branches for accessing other data. This limitation may lead to excessively long linked paths in practice. To address this issue, Lin et al. [29] propose a generalized autonomous path proxy re-encryption scheme (ABP-PRE) that supports branching functionality.

Threshold proxy re-encryption. Recently, several threshold PRE schemes have been proposed and applied to specific scenarios. For instance, Feng et al. [16] introduce a certificateless threshold PRE scheme and apply it to data sharing in cloud-chain collaboration within industrial IoT environments. Patil et al. [17] propose a new threshold PRE scheme designed for networks with resource-constrained environments. However, all of these schemes require pairing operations. More recently, Zhao et al. [30] propose a threshold PRE scheme called AB-TPRE, which is based on key-policy attribute-based encryption and the (t, n)-threshold secret sharing scheme [25]. In their scheme, the re-encryption key is split into n shares, with each proxy receiving one share to generate a re-encrypted ciphertext share. Nunez et al. [6] also draw on Shamir's Secret Sharing [25] to propose a threshold PRE scheme, Umbral, that does not require pairing operations. This scheme utilizes secret distribution and secret reconstruction techniques during the re-encryption key generation and decryption phases, respectively. In this paper, the construction of our threshold PRE scheme is inspired by their approaches, and our PRE is also pairing-free.

Verifiable proxy re-encryption. Recently, some researches have focused on the public verifiability property of PRE [5], [6], [15]. Ohata et al. [5] propose a verifiable PRE scheme based on the re-splittable threshold public key encryption scheme [14], achieving re-encryption verifiability. However,

in their scheme, only the delegatee is permitted to verify the correctness of the re-encrypted ciphertext, thereby detecting and suppressing malicious behavior by the proxy. Other external users cannot perform such verification, which may lead to situations where the delegatee maliciously accuses the proxy. In the threshold PRE scheme Umbral, proposed by Nunez et al. [6], corresponding zero-knowledge proofs are generated during the re-encryption process to ensure the correctness of the re-encrypted ciphertext. This guarantees the public verifiability of the re-encryption operation, meaning any verifier can check the correctness of the re-encrypted ciphertext. Ge et al. [15] propose a verifiable PRE scheme, VF-CP-ABPRE, based on ciphertext-policy attribute-based encryption [31]. They leverage the technique of commitment and message-lock encryption (MLE) [32] to guarantee the correctness of the re-encrypted ciphertext, which can be publicly verified. Although the aforementioned researches have mentioned the properties of verifiability or public verifiability, they are limited to the re-encrypted ciphertext. These works consider only the verifiability of the re-encrypted ciphertext and overlook the verifiability of the re-encryption key.

III. Preliminaries

A. Shamir SS on Groups

A (t, n)-threshold Shamir Secret Sharing (SS) scheme [25] (where $0 < t < n$) enables a dealer to share a secret $s \in \mathbb{Z}_p$ among n shareholders. Any set of at least t shares, provided by the shareholders, can reconstruct the secret $s \in \mathbb{Z}_p$. The Shamir SS on Groups scheme [22] allows a dealer to share a secret $S = g^s$, where S is an element of a group $\mathbb{G}$ with generator g. The scheme can be described using two algorithms as follows:

- $(\{A_i\}_{i\in[n]}, m(X)) \leftarrow \mathsf{GS.Share}(pp_{sh}, S)$. In the algorithm, the dealer first selects a random $(t-1)$-degree polynomial $m(X)$ with $\{\alpha_i\}$ are the coefficients, satisfying $m(\alpha_0) = 0$. The dealer then computes the share $A_i = S \cdot g^{m(\alpha_i)}$ and securely sends to shareholder i.
- $S \leftarrow \mathsf{GS.Recon}(pp_{sh}, \mathcal{T}, \{A_i\}_{i\in\mathcal{T}})$. Given a set of shareholders $\mathcal{T}$, each shareholder $i \in \mathcal{T}$ first computes the Lagrange interpolation coefficients $\lambda_{i,\mathcal{T}} = \Pi_{j\in\mathcal{T}, j\neq i} \frac{\alpha_0-\alpha_j}{\alpha_i-\alpha_j}$, where $\mathcal{T} \subseteq [n]$ and $|\mathcal{T}| \geq t$. The dealer's secret is recovered as $S = \Pi_{i\in\mathcal{T}} A_i^{\lambda_{i,\mathcal{T}}}$.

B. Discrete Logarithm Equality (DLEQ)

The discrete logarithm equality (DLEQ) problem involves proving the equality of two discrete logarithms for two group elements. Given two generators a and b of an elliptic curve group, and a secret value $x \in \mathbb{Z}_p$, where $A = a^x$ and $B = b^x$. The goal is to prove that $log_a A = log_b B$ without revealing x. A non-interactive zero knowledge (NIZK) proof of DLEQ can be constructed by leveraging Σ-protocol [33] and Fiat-Shamir heuristic [34]. The protocol can be described using two algorithms as follows:

- $\pi_x \leftarrow \mathsf{DLEQ.Proof}(x; a, A, b, B)$. The algorithm enables a prover to generate an NIZK proof π_x to prove that the discrete logarithms of A and B are equal.
- $(0/1) \leftarrow \mathsf{DLEQ.Verify}(a, A, b, B, \pi_x)$. This algorithm allows any external verifier to verify that the prover knows the discrete logarithm x without revealing it.

C. Publicly verifiable Secret Sharing

Compared to traditional secret sharing schemes [25] (where $0 < t < n$), the PVSS scheme can not only enable a dealer to share secrets among n shareholders, but also allow any external verifier to verify the validity of all the secret shares from both the dealer and shareholders. The PVSS proposed by Cascudo et al. [22] can be described as below:

- $pp \leftarrow \mathsf{PVSS.Setup}(1^\lambda, t, n)$. Given security parameters λ, t and n, a list of public parameters $pp = (\mathbb{G}, g, t, n, \alpha_0, \{(\alpha_i, v_i)\}_{i\in[n]})$ are generated, where $\mathbb{G}$ is a a group with generator g, $\alpha_0, \alpha_1, ..., \alpha_n \in \mathbb{Z}_p$ are randomly selected values and $v_i = \Pi_{j\in[n], j\neq i}(\alpha_i - \alpha_j)^{-1}$.
- $(\{C_i\}, \pi_{sh}) \leftarrow \mathsf{PVSS.Share}(pp, pk_D, sk_D, \{pk_i\}, S)$. The algorithm takes public parameters pp, the dealer's key pair (pk_D, sk_D), n shareholders' public keys and the secret $S(= g^s$, where $s \in \mathbb{Z}_p)$ as inputs. It generates n encrypted shares for the shareholders and a DLEQ proof π_{sh}. Note that private keys sk_D and $\{sk_i\}_{i\in[n]}$ are randomly selected values from $\mathbb{Z}_p$ and public keys $pk_D = g^{sk_D}$ and $\{pk_i = g^{sk_i}\}_{i\in[n]}$.
- $(0/1) \leftarrow \mathsf{PVSS.Verify}(pp, pk_D, \{(pk_i, C_i)\}_{i\in[n]}, \pi_{sh})$. The algorithm allows any external verifier uses pp, pk_D, $\{(pk_i, C_i)\}_{i\in[n]}$ and π_{sh} to verify the validity of the encrypted shares from the dealer.
- $(A_i, \pi_{Deci}) \leftarrow \mathsf{PVSS.PreRecon}(pp, pk_D, pk_i, sk_i, C_i)$. Each shareholder i to decrypt his encrypted share C_i and generate the corresponding zero knowledge proof π_{Deci}.
- $(0/1) \leftarrow \mathsf{PVSS.VerifyDec}(pp, pk_D, pk_i, C_i, A_i, \pi_{Deci})$. The algorithm allows any external verifier to verify the validity of the decrypted share from shareholder i.
- $S \leftarrow \mathsf{PVSS.Recon}(pp, \{A_i\}_{i\in\mathcal{T}})$. This algorithm reconstructs the orginal secret S on group $\mathbb{G}$ using at least t shares by calling $\mathsf{GS.Recon}$.

PVSS schemes generally satisfy the following properties:

- ***Correctness*** Correctness guarantees that if the dealer and at least t shareholders follow the protocol, the reconstructed secret will be identical to the original secret.
- ***Public Verifiability*** Public Verifiability allows any external verifier to validate the correctness of all the secret shares from both the dealer and shareholders.
- ***IND1-Secrecy*** IND1-Secrecy [35] ensures that any set of fewer than t shareholders cannot gain any information about the original secret.

D. Ethereum Blockchain

Ethereum [36] is a decentralized, open-source blockchain platform that enables developers to build and deploy decentralized applications (dApps). It provides a programmable environment through smart contracts, which are self-executing contracts. Transactions in smart contract execution consumes "Gas", a unit that measures the computational resources required for an operation or storage cost. Gas fees ensure

efficient use of resources. The ability to verify operations publicly makes Ethereum an ideal tool for building decentralized systems.

IV. Publicly Verifiable Threshold Proxy Re-encryption (PVTPRE)

A. Definition

The entities in PVTPRE include the delegator A, the delegatee B and the set of proxies $\{i\}_{i\in[n]}$. The PVTPRE scheme consists of the following algorithms:

- $\mathsf{PRE.Setup}(1^\lambda, t, n, l)$: The algorithm takes λ, t, n, l as inputs and outputs the public parameters par and the global parameter s. The parameter s is privately held by the delegator A.
- $\mathsf{PRE.KeyGen}(par)$: On input of the public parameters par, all entities $\{j\}_{j\in\{A,B,i\}}$ use this algorithm to generate a public/private key pair (pk_j, sk_j).
- $\mathsf{PRE.ReKeyGen}(pk_B, sk_A, pk_A, \{pk_i\}_{i\in[n]}, s)$: This algorithm takes all entities' public keys, the delegator's private key and the global parameter s as inputs to generate the re-encryption keys $\{ckFrag_i\}_{i\in[n]}$ for all proxies along with proof π_{sh}.
- $\mathsf{PRE.ReKeyVerify}(pk_A, pk_B, \{ckFrag_i, pk_i\}_{i\in[n]}, \pi_{sh})$: On input of all public keys, the re-encryption keys and the proof, this algorithm verifies the correctness of the re-encryption keys. If the algorithm outputs "1", the re-encryption keys are correct; otherwise, they are incorrect.
- $\mathsf{PRE.Enc}_2(pk_A, M, s)$: On input of A's public key, s, and data M, the algorithm outputs the second level ciphertext C that can be re-encrypted into the first level ciphertext C'.
- $\mathsf{PRE.Enc}_1(pk_B, pk_A, sk_A, M, s)$: The algorithm takes the public keys of A and B, A's private key, M and s to generate the first level ciphertext C' that cannot be re-encrypted.
- $\mathsf{PRE.ReEnc}(pk_A, \{ckFrag_i, pk_i, sk_i\}_{i\in[n]}, C)$: Given A's public key, all proxies' key pairs, the re-encryption keys and the second level ciphertext, the re-encryption algorithm outputs the first level ciphertext C' along with the proofs $\{\pi_{rei}\}_{i\in[n]}$.
- $\mathsf{PRE.ReEncVerify}(ckFrag_i, C'_{2i}, \pi_{rei}, pk_i, pk_A)$: The algorithm takes as input the re-encryption keys, the first-level ciphertext, the proxies' public keys, A's public key, and the proofs $\{\pi_{rei}\}$. It outputs "1" to indicate that the first level ciphertext is valid, or "0" to indicate that it's invalid.
- $\mathsf{PRE.Dec}_2(sk_A, C)$: On input A's private key and second level ciphertext, the algorithm outputs the data M.
- $\mathsf{PRE.Dec}_1(pk_A, sk_B, C', \mathcal{T})$: On input public key of A and the private key of B, the first level ciphertext and the set of proxies $\mathcal{T}$, the algorithm outputs the data M.

The PVTPRE satisfies the following properties:

- **Correctness:** The delgator and the delegatee can correctly decrypt the second level and first level ciphertext, respectively.
- **Public verifiability:** The re-encryption keys and the re-encrypted ciphertext are publicly verifiable.
- **Fault tolerance:** A delegatee colluding with at most $t-1$ malicious proxies cannot recover the plaintext.
- **Secrecy:** The PVTPRE satisfies ***IND-CPA*** security and its definition is given in Appendix B, following the definitions in [19], [20]. Besides, colluding proxies cannot recover the plaintext.

B. Construction

Figure 1 illustrates the concrete construction of the proposed PVTPRE scheme. The construction mainly utilizes PVSS as a cryptographic primitive. Below, we provide detailed explanations of the implementation of the primary algorithms: PRE.Setup, PRE.ReKeyGen, PRE.ReKeyVerify, PRE.ReEnc, and $\mathsf{PRE.Dec}_1$.

The $\mathsf{PRE.Setup}(1^\lambda, t, n, l)$ algorithm takes as inputs λ, t, n, l to generate the public parameters $par = (pp, \mathcal{H}, \mathsf{KDF})$ and the global parameter s. The $pp = (\mathbb{G}, g, t, n, \alpha_0, \{(\alpha_i, v_i)\}_{i\in[n]})$ is generated by PVSS.Setup. The key derivation function $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^l$ is modeled as a random oracle. $\mathcal{H}$ is a random oracle that maps inputs to a polynomial with degree less than $n-t-2$. Moreover, the global parameter s is randomly selected from $\mathbb{Z}_p$, is one-time use only, and is privately owned by the delegator A. In the subsequent algorithms, we omit the public parameters par from the input.

In the $\mathsf{PRE.ReKeyGen}(pk_B, sk_A, pk_A, \{pk_i\}_{i\in[n]}, s)$ algorithm, delegator A generates re-encryption keys $\{ckFrag_i\}$ for the designated proxies, along with the corresponding NIZK proof π_{sh} using PVSS.Share algorithm. Note that we have made a slight modification to the PVSS.Share algorithm: the public key pk_B of the delegatee is incorporated into the share encryption process, as depicted in Figure 1. This modification prevents proxies from recovering the key seed from $\{C'_{2i}\}$, thereby ensuring the confidentiality of the delegator's data. When considering $pk_B \cdot pk_i$ as a whole, which represents the public key of shareholder i in the DHPVSS scheme [22], the modification is essentially equivalent to the original DHPVSS scheme and does not impact its security.

In the $\mathsf{PRE.ReKeyVerify}(pk_A, pk_B, \{ckFrag_i, pk_i\}_{i\in[n]}, \pi_{sh})$ algorithm, the PVSS.Verify algorithm is used to verify the validity of the re-encryption keys generated by the delegator A. When invoking the PVSS.Verify algorithm, $pk_B \cdot pk_i$ can similarly be treated as a whole.

In the $\mathsf{PRE.ReEnc}(pk_A, \{ckFrag_i, pk_i, sk_i\}_{i\in[n]}, C)$ algorithm, the delegator's public key, the re-encryption keys, proxies' private keys and the second level ciphertext are used to generate the first level ciphertext $C' = (C_1, \{C'_{2i}\}_{i\in[n]})$, along with the corresponding NIZK proofs $\{\pi_{rei}\}_{i\in[n]}$, which ensure the validity of the first-level ciphertext. This process corresponds to the PVSS.PreRecon algorithm.

The $\mathsf{PRE.ReEncVerify}(ckFrag_i, C'_{2i}, \pi_{rei}, pk_i, pk_A)$ algorithm executes the PVSS.VerifyDec algorithm to verify the validity of the first level ciphertext, i.e., re-encrypted ciphertext.

In the $\mathsf{PRE.Dec}_1(pk_A, sk_B, C', \mathcal{T})$ algorithm, the first level C' is first parsed as $(C_1, \{C'_{2i}\}_{i\in[n]})$. Then, the subset $\{C'_{2i}\}_{i\in\mathcal{T}}$ are decrypted using the delegatee's private key to

Functionality The proposed publicly verifiable threshold PRE scheme

$\underline{(par, s) \leftarrow \mathsf{PRE.Setup}(1^\lambda, t, n, l):}$

$pp \leftarrow \mathsf{PVSS.Setup}(1^\lambda, t, n)$, where $pp = (\mathbb{G}, g, t, n, \alpha_0, \{(\alpha_i, v_i)\}_{i\in[n]})$

$par \leftarrow (pp, \mathcal{H}, \mathsf{KDF})$

$s \xleftarrow{R} \mathbb{Z}_p$ (s is private to the delegator)

$\underline{(pk_j, sk_j) \leftarrow \mathsf{PRE.KeyGen}():}$

$sk_j \xleftarrow{R} \mathbb{Z}_p$, $pk_j \leftarrow g^{sk_j}$

$\underline{(\{ckFrag_i\}_{i\in[n]}, \pi_{sh}) \leftarrow \mathsf{PRE.ReKeyGen}(pk_B, sk_A, pk_A, \{pk_i\}_{i\in[n]}, s):}$

$$\mathsf{PVSS.Share}\begin{cases} pp_{sh} \leftarrow pp \setminus \{v_i\}_{i\in[n]}, \\ (\{kFrag_i\}_{i\in[n]}, m(X)) \leftarrow \mathsf{GS.Share}(pp, g^s), \\ \forall i \in [n],\ ckFrag_i \leftarrow (pk_B \cdot pk_i)^{sk_A} \cdot kFrag_i, \\ m^* \leftarrow \mathcal{H}(pk_A, pk_B, \{(pk_i, ckFrag_i)\}_{i\in[n]}), \\ V \leftarrow \Pi_{i=1}^n ckFrag_i^{v_i \cdot m^*(\alpha_i)},\ U \leftarrow \Pi_{i=1}^n (pk_B \cdot pk_i)^{v_i \cdot m^*(\alpha_i)}, \\ \pi_{sh} \leftarrow \mathsf{DLEQ.Proof}(sk_A; g, pk_A, U, V) \end{cases}$$

$\underline{(0/1) \leftarrow \mathsf{PRE.ReKeyVerify}(pk_A, pk_B, \{ckFrag_i, pk_i\}_{i\in[n]}, \pi_{sh}):}$

$$\mathsf{PVSS.Verify}\begin{cases} m^* \leftarrow \mathcal{H}(pk_A, pk_B, \{(pk_i, ckFrag_i)\}_{i\in[n]}), \\ V \leftarrow \Pi_{i=1}^n ckFrag_i^{v_i \cdot m^*(\alpha_i)},\ U \leftarrow \Pi_{i=1}^n (pk_B \cdot pk_i)^{v_i \cdot m^*(\alpha_i)}, \\ (0/1) \leftarrow \mathsf{DLEQ.Verify}(g, pk_A, U, V, \pi_{sh}) \end{cases}$$

$\underline{C \leftarrow \mathsf{PRE.Enc}_2(pk_A, M, s):}$

$K \leftarrow \mathsf{KDF}(g^s)$, $C_1 \leftarrow \mathsf{AES.Enc}(M, K)$

$C \leftarrow (C_1, C_2 = pk_A^s)$

$\underline{C' \leftarrow \mathsf{PRE.Enc}_1(pk_B, sk_A, M, s):}$

$K \leftarrow \mathsf{KDF}(g^s)$, $C_1 \leftarrow \mathsf{AES.Enc}(M, K)$

$(\{kFrag_i\}_{i\in[n]}, m(X)) \leftarrow \mathsf{GS.Share}(pp, g^s)$

$\forall i \in [n], C'_{2i} \leftarrow pk_B^{sk_A} \cdot kFrag_i$

$C' \leftarrow (C_1, \{C'_{2i}\}_{i\in[n]})$

$\underline{(C', \{\pi_{rei}\}_{i\in[n]}) \leftarrow \mathsf{PRE.ReEnc}(pk_A, \{ckFrag_i, pk_i, sk_i\}_{i\in[n]}, C):}$

Parse C as (C_1, C_2)

$$\forall i \in [n],\ \mathsf{PVSS.PreRecon}\begin{cases} C'_{2i} \leftarrow ckFrag_i / pk_A^{sk_i}, \\ \pi_{rei} \leftarrow \mathsf{DLEQ.Proof}(sk_i; g, pk_i, pk_A, ckFrag_i / C'_{2i}) \end{cases}$$

$C' \leftarrow (C_1, \{C'_{2i}\}_{i\in[n]})$

$\underline{(0/1) \leftarrow \mathsf{PRE.ReEncVerify}(ckFrag_i, C'_{2i}, \pi_{rei}, pk_i, pk_A):}$

$\mathsf{PVSS.VerifyDec}\{\ (0/1) \leftarrow \mathsf{DLEQ.Verify}(g, pk_i, pk_A, ckFrag_i / C'_{2i}, \pi_{rei})$

$\underline{M \leftarrow \mathsf{PRE.Dec}_2(sk_A, C):}$

Parse C as (C_1, C_2)

$K \leftarrow \mathsf{KDF}(C_2^{1/sk_A})$, $M \leftarrow \mathsf{AES.Dec}(C_1, K)$

$\underline{M \leftarrow \mathsf{PRE.Dec}_1(pk_A, sk_B, C', \mathcal{T}):}$

Parse C' as $(C_1, \{C'_{2i}\})$

$\forall i \in \mathcal{T}, kFrag_i \leftarrow C'_{2i} / pk_A^{sk_B}$

$$\mathsf{PVSS.Recon}\begin{cases} pp_{sh} \leftarrow pp \setminus \{v_i\}_{i\in[n]}, \\ PreK \leftarrow \mathsf{GS.Recon}(pp_{sh}, \mathcal{T}, \{ckFrag_i\}_{i\in\mathcal{T}}) \end{cases}$$

$K \leftarrow \mathsf{KDF}(PreK)$, $M \leftarrow \mathsf{AES.Dec}(C_1, K)$

Fig. 1: Construction of the proposed publicly verifiable threshold PRE scheme

generate the shares $\{kFrag_i\}_{i\in\mathcal{T}}$ of the key seed. These t shares are subsequently used to recover the key seed via the PVSS.Recon algorithm. After the key seed is recovered, the same operations as in the $\mathsf{PRE.Dec}_2$ algorithm are performed to decrypt C_1 and obtain the plaintext M.

C. Security Analysis

Correctness: The correctness of the decryption algorithm can be explained as follows:

- Given C is the second level ciphertext, decryption of C by leveraging $\mathsf{PRE.Dec}_2$ is correct, due to Equations (1) and (2).

$$\mathsf{KDF}(C_2^{1/sk_A}) = \mathsf{KDF}(g^s) = K \tag{1}$$

$$\mathsf{AES.Dec}(\mathsf{AES.Enc}(M, K), K) = M \tag{2}$$

- Given C' is the first level ciphertext (i.e. the re-encrypted ciphertext), decryption of C' by leveraging $\mathsf{PRE.Dec}_1$ is correct, due to Equations (3), (4) and (2).

$$C'_{2i}/pk_A^{sk_B} = (kFrag_i \cdot pk_B^{sk_A})/pk_A^{sk_B} = kFrag_i \tag{3}$$

$$\mathsf{GS.Recon}(\{kFrag_i \in \mathsf{GS.Share}(g^s)\}_{i\in\mathcal{T}}) = g^s \tag{4}$$

Public verifiability: In our PRE scheme, the public verifiability is primarily reflected in two aspects: first, the public verifiability of re-encryption keys, and second, the public verifiablity of the re-encryption ciphertext (i.e., the first level ciphertext).

Theorem 1: The re-encryption keys in our PRE scheme are publicly verifiable. Namely, any external verifier with publicly known information can verify the correctness of the re-encryption keys generated by the delegator.

$Proof$: The public verifiability property of the re-encryption keys essentially inherits from the DHPVSS [22]. In our PRE scheme, the PRE.ReKeyGen algorithm used by delegator to generate the re-encryption keys corresponds to the PVSS.Share algorithm, and the generated re-encryption keys are associated with the encryption shares. Since the encryption shares in the DHPVSS scheme have been proven to possess the property of public verifiability, the re-encryption keys generated in our PRE scheme also exhibit public verifiability.

Theorem 2: The first level ciphertext in our PRE scheme is publicly verifiable. Namely, any external verifier with publicly known information can verify the correctness of the re-encryption ciphertext generated by proxies.

$Proof$: Simultaneously with the generation of the first level ciphertext, we require each proxy i to generate a non-interactive zero-knowledge proof π_{rei} concerning the re-encryption ciphertext, in order to ensure the correctness of the re-encryption ciphertext. More precisely, if $\mathsf{PRE.ReEncVerify}(ckFrag_i, C'_{2i}, \pi_{rei}, pk_i, pk_A) = 1$ then π_{rei} is a valid proof of knowledge of discrete logarithm equality for g, pk_i, pk_A, $ckFrag_i/C'_{2i}$. Therefore, the proxy i's private key sk_i such that $pk_i = g^{sk_i}$ and $ckFrag_i/C'_{2i} = pk_A^{sk_i}$ can be extracted from π_{rei}. Therefore $C'_{2i} = ckFrag_i/pk_A^{sk_i}$, and so $\mathsf{PRE.ReEnc}(pk_A, \{ckFrag_i, sk_i\}_{i\in[n]}, C) = (\{C'_{2i}\}_{i\in[n]}, \cdot)$ for any randomness input to this algorithm.

Fault tolerance: This property can be proved by Theorem 6 in Appendix C.

Secrecy: The proof of ***IND-CPA*** security and resistance to collusion attacks can be found in Appendix C.

D. Complexity

We compare the computational complexity between our PVTPRE with previous schemes in Table I. In the table, n represents the number of proxies, t denotes the threshold value. m is the size of the set of user attributes with access rights, and r refers to the total number of attributes in scheme [15] based on attribute-based encryption. The results show that many algorithms in PVTPRE has lower computational overhead compared to Umbral, which achieve similar functionalities, with both schemes being paring-free. Moreover, only the proposed PVTPRE allows for the verification of the validity of re-encryption keys.

V. Application in Data Rights Confirmation

In this section, we present the framework for data rights confirmation, based on the proposed PVTPRE scheme. The framework enables the realization of the three rights (ownership, control and usage) separation model. Figure 2 provides a high-level overview of the proposed framework, leveraging the PVTPRE algorithms.

As illustrated in Figure 2, our framework primarily consists of four entities: data owner A, data user B, authorizers $\{i\}$, and data bank. Under the three rights separation model, the rights and responsibilities of each entity are defined as follows:

- **Data owner** A: Data owner A, the PRE delegator, is the producer and owner of the data. A retains full ownership

TABLE I: Computation complexity

Ref.	Multiple proxies	$\mathsf{PRE.Enc}_2$		PRE.ReKeyGen		PRE.ReKeyVerify		PRE.ReEnc		PRE.ReEncVerify		$\mathsf{PRE.Dec}_1$	
		Exp.	Pair.	Exp.	Pair.	Sup.	Exp.	Exp.	Pair.	Sup.	Exp.	Exp.	Pair.
[37]	✗	4	1	1	-	✗	-	1	1	✗	-	1	1
[29]	✗	4	1	4	1	✗	-	-	1	✗	-	2	2
[10]	✗	2	1	3	-	✗	-	-	1	✗	-	1	1
[38]	✗	4	-	5	-	✗	-	1	1	✗	-	2	3
[15]	✗	$3r$+5	4	m+$3r$+8	2	✗	-	r	$2r$+2	✓	3	r+3	$2r$+1
[5]	✗	1	-	4	-	✗	-	4	-	✓	2	6	-
[16]	✓	4	1	n+1	-	✗	-	$3t$	$2t$	✗	-	1	-
Umbral [6]	✓	3	-	$2n$+3	-	✗	-	$5n$	-	✓	$6n$	$2t$+3	-
Our	✓	2	-	$4n$+2	-	✓	$2n$+4	$3n$	-	✓	$4n$	t+1	-

rights over the data and has the authority to decide who may access and use the data generated by him.

- **Data user** B: Data user B, the PRE delegatee, can submit a request to gain usage rights for A's data.
- **Authorizers** $\{i\}$: The set of authorizers, PRE proxies, forms a decentralized committee. These authorizers are fault-tolerant in data control, as they are responsible for performing data re-encryption.
- **Data Bank**: Data bank is composed of two primary components: the blockchain and a distributed file system. The blockchain is responsible for data registration and rights certification, ensuring transparency and traceability throughout the rights confirmation process. The distributed file system, e.g., IPFS, is tasked with securely and efficiently storing data files.

In the following description, we omit details regarding the initialization, such as the setup of public parameters and the key pair generation.

A. Ownership of Dara owner

Without loss of generality, we assume that the data owner A has already generated some data through certain activities, which we treat as a data file.

(1) To manage the data safely and efficiently, A first encrypts the data file using the $\mathsf{PRE.Enc}_2$ algorithm, producing the corresponding second level ciphertext file $C = (C_1, C_2)$. C_1 is then stored in the distributed file system, which returns the hash of C_1 to the owner A.

(2) Subsequently, A generates metadata for the plaintext data file and uploads it to the data bank. The structure and descriptions of the fields in metadata are detailed in Table II, similar to Zhang et al. [39].

TABLE II: Metadata to describe the ownership

Field	Type	Description
ID	string	identifier of the data owner
hash	string	the hash of C_1 within the second level ciphertext stored in data bank
size	int256	the size of the data file
description	string	a brief description of the data content
timestamp	uint	the time of uploading the metadata from the local

Upon receiving the Metadata uploaded by the data owner A, the smart contract on blockchain first checks the existence of the Metadata on blockchain by the hash field value hash of Metadata. If the Metadata has already been recorded on blockchain, it indicates that A is attempting to illegitimately claim ownership of the data, and this malicious behavior will be suppressed. Conversely, if it is not found on the blockchain, it confirms that data owner A is indeed the producer of the corresponding data, and the smart contract records the Metadata on blockchain. At this point, the data ownership confirmation process for data owner A is formally concluded.

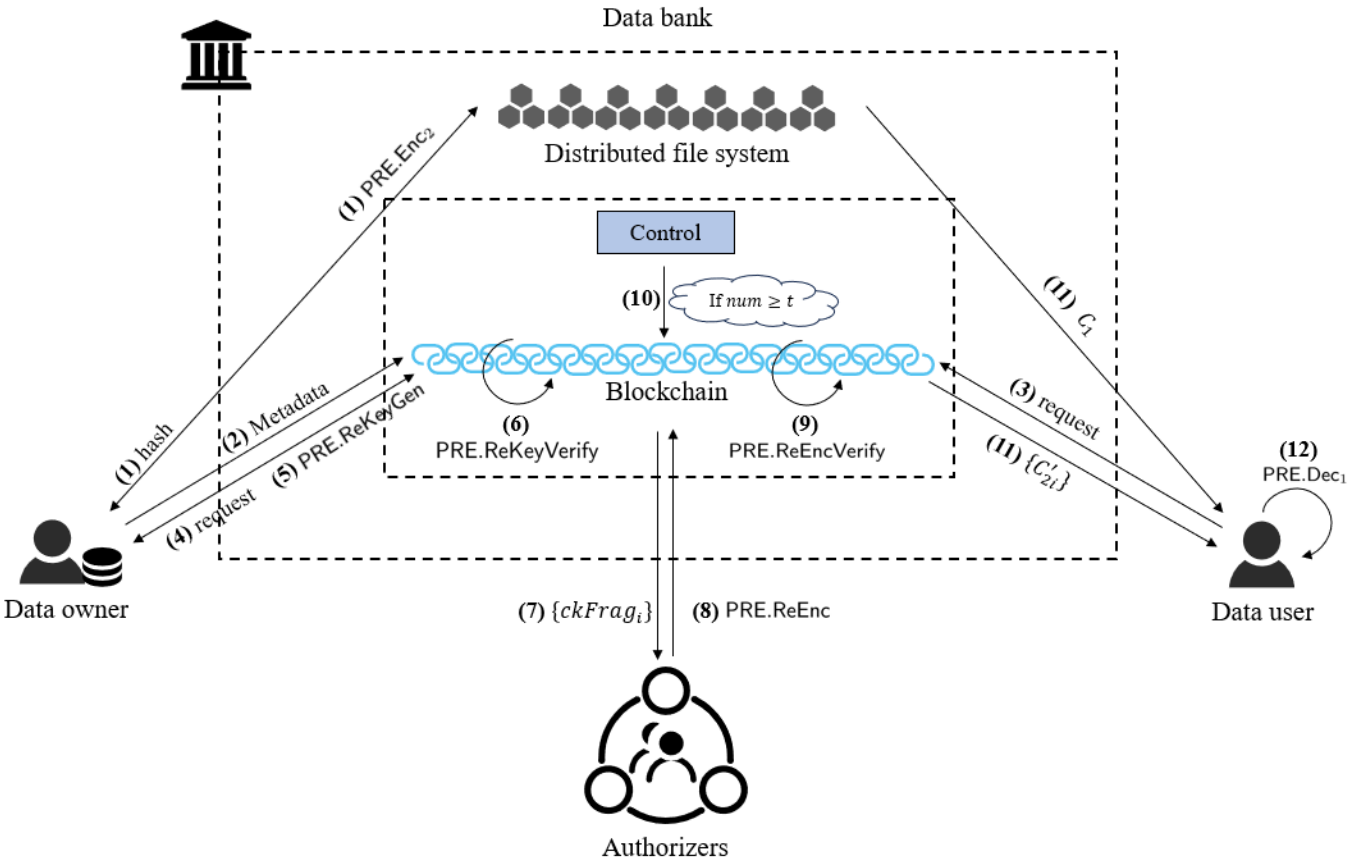

Fig. 2: Overview of our data rights confirmation framework

B. Data control by Authorizers

In our setting, the n authorizers collectively form a decentralized committee, each sharing equal control rights over the data owned by A. The control over the owner's data is implied through the execution of the re-encryption algorithm. We now continue with the introduction of the data rights confirmation framework, as illustrated in Figure 2.

(3) Suppose B sends a data usage request to blockchain. The request includes the identities of data owner and data user, the hash of C_1, and current timestamp.
(4) Data owner A then extracts data user's public key pk_B from the request.
(5) A leverages the PRE.ReKeyGen algorithm to generate the re-encryption keys $\{ckFrag_i\}$ along with the NIZK proof π_{sh} and uploads them to blockchain.
(6) The smart contract invokes the PRE.ReKeyVerify algorithm to verify the proof π_{sh}. If the result returns "1", it indicates that the delegator A has honestly generated the re-encryption keys.
(7) Each authorizer i downloads the corresponding re-encryption key $ckFrag_i$ from blockchain.
(8) Authorizer i then leverages the PRE.ReEnc algorithm to generate C'_{2i} of the first level ciphertext $C' = (C_1, \{C'_{2i}\})$ and the corresponding NIZK proof π_{rei}, which are uploaded to blockchain.
(9) The uploaded re-encrypted ciphertext is verified by the smart contract using the PRE.ReEncVerify algorithm. If the verification result for π_{rei} is "1", it indicates that authorizer i agrees to grant B the right to use the data. Otherwise, authorizer i is regarded as faulty.
(10) Denote the number of correct proofs $\{\pi_{rei}\}$ as num. The smart contract collects at least $t(\leq num)$ correct proofs and records the corresponding data control history according to Table III.

TABLE III: The table to describe data Control

Field	Type	Description
user's ID	string	the identity of the data user granted the right of use for certain data
authorizers' ID	strings	the identities of the authorizers who agree to grant the right of use
hash	string	the hash of C_1 within the second level ciphertext stored in data bank
states	bool	whether $num \geq t$ is satisfied
timestamp	uint	the time of the notarization posted

C. Usage for Data user

If a threshold of authorizers have uploaded correct re-encrypted ciphertext, the user B can access A's data with following steps.

(11) B downloads $C' = (C_1, \{C'_{2i}\})$ from data bank.
(12) B decrypts C' leveraging the $\mathsf{PRE.Dec}_1$ algorithm with his private key to access the desired data.

We now consider the case where the decrypted data is incorrect, i.e., the data owner is malicious during the execution of the $\mathsf{PRE.Enc}_2$ algorithm. In this case, the data user B can submit a dispute to blockchain, accusing the data owner A of malicious behavior. The dispute $pk_A^{sk_B}$, along with $\{\pi_{dis}\}$, is constructed by Equation (5), which eliminate the possibility of false accusation.

$$\{\pi_{dis}\} \leftarrow \mathsf{DLEQ.Proof}(sk_B; g, pk_B, pk_A, pk_A^{sk_B}) \quad (5)$$

The legitimacy of the dispute can be verified by Equation (6). Additionally, anyone can compute $kFrag_i$ and recover the $PreK$, as the user does, to confirm that the correct data M cannot be obtained, thereby proving that the data owner is malicious in executing the $\mathsf{PRE.Enc}_2$ algorithm.

$$\mathsf{DLEQ.Verify}(g, pk_B, pk_A, pk_A^{sk_B}, \pi_{dis}) \stackrel{?}{=} 1. \quad (6)$$

VI. Evaluation

We evaluate the proposed PVTPRER scheme and the data rights confirmation framework. Table IV describes the specific environment used of our experiments. The source code is available on Github[1].

TABLE IV: Experiment environment

Virtual Machine	VMware Workstation
CPU	Intel (R) Core (TM) i7-11800H @2.30GHz
Operating System	Ubuntu 22.04.5 LTS
Programming Language	Golang, Solidity
Compiler	Go 1.22.0, Solc 0.8.20
Elliptic Curve	BN128
Blockchain	Ethereum test network
Distributed File System	IPFS

First, we introduce the performance of the primary algorithms involved in the proposed PVTPRE scheme, namely, $\mathsf{PRE.Enc}_2$, PRE.ReKeyGen, PRE.ReEnc, PRE.ReEncVerify, $\mathsf{PRE.Dec}_1$, and PRE.ReKeyVerify. Additionally, we compare the results of these experiments with Umbral [6]. In the implementation of $\mathsf{PRE.Enc}_2$ and $\mathsf{PRE.Dec}_1$, the symmetric encryption used in both schemes is realized with the AES-256 encryption algorithm in GCM mode.

Figure 3 illustrates the computation cost of $\mathsf{PRE.Enc}_2$ algorithm as the size of the data increases. It can be seen that the computation cost of $\mathsf{PRE.Enc}_2$ increases linearly with the size. For a data size of 100 MB, it costs about 33.4 ms in our scheme while Umbral incurs a cost of about 34.5 ms.

Figure 4, Figure 5, Figure 6, and Figure 7 depict the computation cost of PRE.ReKeyGen, PRE.ReEnc, PRE.ReEncVerify, and $\mathsf{PRE.Dec}_1$, respectively, as the number of proxies (n) increases. It can be observed that computational overhead of all these algorithms increases linearly with the number of proxies n. Furthermore, all operations can be accomplished within 100 ms given $n = 100$, demonstrating their practicality. Except the PRE.ReKeyGen algorithm, our PVTPRE significantly outperforms Umbral. The PRE.ReKeyGen algorithm incurs higher computational overhead due to the additional NIZK proofs that need be generated. The NIZK proofs are adopted in the PRE.ReKeyVerify algorithm, the cost of which is shown in Figure 8. The experimental result is align with the complexity analysis presented in Table I.

[1]https://github.com/AppCrypto/PVTPRE

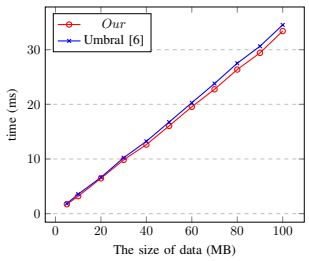

Fig. 3: Computation cost of PRE.Enc$_2$

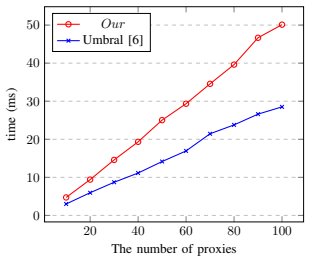

Fig. 4: Computation cost of PRE.ReKeyGen

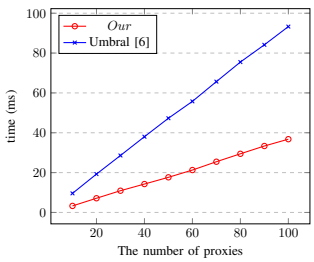

Fig. 5: Computation cost of PRE.ReEnc

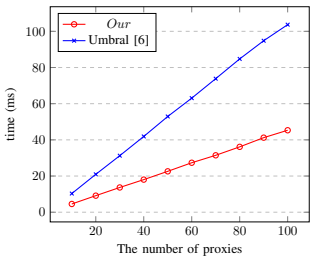

Fig. 6: Computation cost of PRE.ReEncVerify off-chain

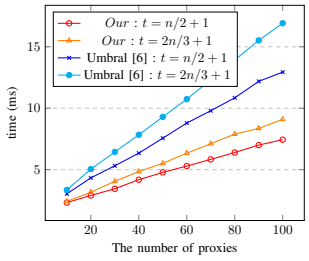

Fig. 7: Computation cost of PRE.Dec$_1$

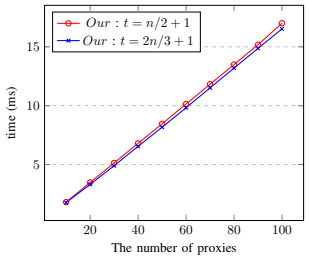

Fig. 8: Computation cost of PRE.ReKeyVerify off-chain

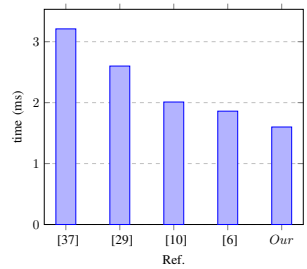

Fig. 9: Computation cost of PRE.Enc$_2$

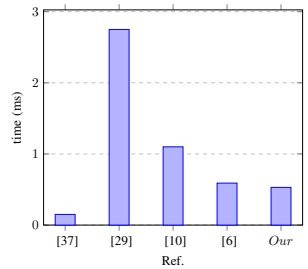

Fig. 10: Computation cost of PRE.ReKeyGen

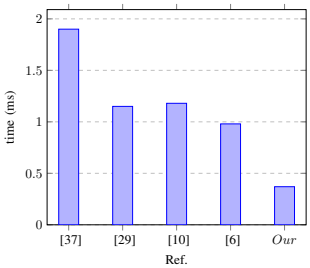

Fig. 11: Computation cost of PRE.ReEnc

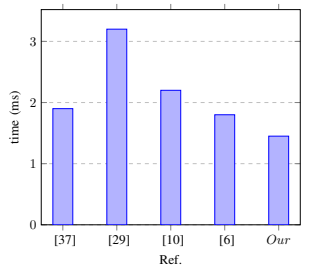

Fig. 12: Computation cost of PRE.Dec$_1$

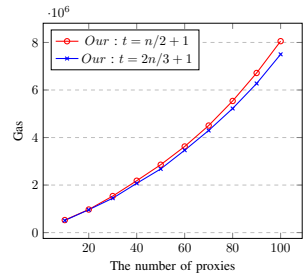

Fig. 13: Gas cost of PRE.ReKeyVerify on-chain

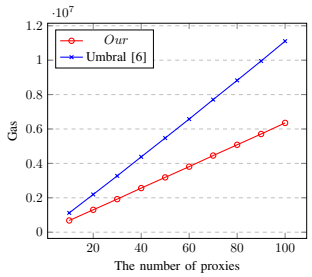

Fig. 14: Gas cost of PRE.ReEncVerify on-chain

For the PRE.Dec$_1$ algorithm, the computation cost is dependent on the threshold t. Therefore, in our experiments, in addition to varying the number of proxies n, we measure the algorithm's time overhead by setting different threshold values t, including $t = n/2 + 1$ and $t = 2n/3 + 1$. As shown in Figure 7, the computation cost is lower when $t = n/2 + 1$ compared to when $t = 2n/3+1$. Furthermore, it is obvious that under the same conditions, the computation cost of PVTPRE is lower that of Umbral.

Furthermore, to better highlight the computational complexity advantages of our PRE scheme, we conduct a broader set of comparative experiments with recent related works, includes [6], [10], [29], [37]. In these experiments, we primarily compare PRE.Enc$_2$, PRE.ReKeyGen, PRE.ReEnc, and PRE.Dec$_1$ algorithms, with the results presented in Figure 9, Figure 10, Figure11, and Figure 12, respectively. Note that [29], [37], and [10] are single-proxy PRE schemes, with [29] being multi-hop PRE schemes. To conduct fair comparison, we set the number of proxies n and the threshold t to "1" in both PVTPRE and Umbral, and the number of hops to "1" in [29].

The underlying elliptic curve BN128 is an asymmetric, with the overhead of exponentiation on $\mathbb{G}_2$ being significantly higher than that on $\mathbb{G}_1$. Figure 9, Figure 11, and Figure 12 demonstrate that the overhead of PRE.Enc$_2$, PRE.ReEnc, and PRE.Dec$_1$ algorithms in our scheme are the lowest, owing to the absence of pairing operations, as shown in Table I. [37] demonstrates the best performance for PRE.ReKeyGen algorithm, as shown in Figure 10. This is because it only involves a single exponentiation operation on group $\mathbb{G}_1$. In contrast, [29] requires multiple operations on $\mathbb{G}_2$ in its PRE.ReKeyGen algorithm, leading to the highest cost.

The PVTPRE is applied within the data rights confirmation framework, where blockchain is leveraged to execute verification algorithms, specifically PRE.ReKeyVerify and PRE.ReEncVerify. Figure 13 and Figure 14 show the on-chain gas consumption of the PRE.ReKeyVerify and PRE.ReEncVerify algorithms, respectively. The gas cost of the re-encryption keys verification algorithm PRE.ReKeyVerify on-chain exhibits super-linear growth as the number of proxies increases. This is because the verification process involves n iterations, with each iteration requiring the computation of a

polynomial of degree $n-t-2$. In contrast, the off-chain verification time cost increases linearly with the number of proxies, as shown in Figure 8. This is due to the fact that the computational overhead of polynomial evaluations over finite fields in Golang is negligible compared to the cost of group-based operations.

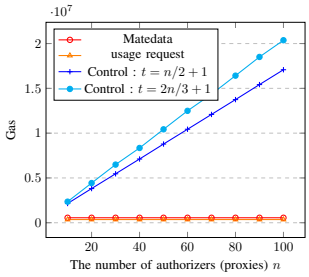

Fig. 15: Gas cost of storage in data control

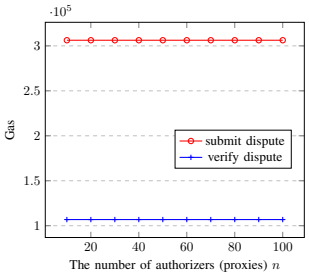

Fig. 16: Gas cost regarding disputes

Additionally, we evaluate the gas consumption required to implement the traceability of data ownership, rights of control and use, within the data rights confirmation framework. This includes storing the Metadata and Control information on the blockchain. We also assess the gas consumption required for data user to submit the usage request. As shown in Figure 15, the gas costs for storing Metadata and submitting data usage request are both constant, at 555,168 and 353,926, respectively. This is due to the fixed, constant size of the fields contained in these two segments. The gas cost for storing data Control information increases linearly with the number of authorizers (proxies). This is because it includes the ID of all authorizers who have agreed to grant the data rights to the data user, as detailed in Table III.

Finally, we evaluate the gas costs for submitting and verifying the dispute. As shown in Figure 16, the gas costs for these two operations are constant, at 306,261 and 106,831, respectively.

VII. Conclusion

In this paper, we propose a novel threshold proxy re-encryption scheme based on publicly verifiable secret sharing, referred to as publicly verifiable threshold PRE (PVTPRE). The public verifiability of our PRE scheme is demonstrated not only in the first-level ciphertext but also in the re-encryption keys, a feature absent in prior research. We also analyze the security of PVTPRE, including correctness, public verifiability, fault tolerance, and secrecy. Furthermore, based on the proposed PVTPRE, we design a non-interactive data rights confirmation framework, which is well-suited in transparent and accountable data management. Finally, we evaluate the performance of our PVTPRE scheme through comparative experiments conducted both on-chain and off-chain. The experimental results validate the feasibility and effectiveness of our PVTPRE scheme.

References

[1] Eltayieb, N., Elhabob, R., Abdelgader, A. M., Liao, Y., Li, F., & Zhou, S. Certificateless Proxy Re-encryption with Cryptographic Reverse Firewalls for Secure Cloud Data Sharing. *Future Generation Computer Systems*, 2025, 162, 107478.

[2] Rodrigues, B., Amorim, I., Silva, I., & Mendes, A. Patient-centric health data sovereignty: an approach using Proxy re-encryption. In European Symposium on Research in Computer Security, 2023, pp. 199-215.

[3] Zhang, L., Ou, Z., Hu, C., Kan, H., & Zhang, J. Data sharing in the metaverse with key abuse resistance based on decentralized CP-ABE, *IEEE Transactions on Computers*, 2024, accepted.

[4] Blaze, M., Bleumer, G., & Strauss, M. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, 1998, pp. 127-144.

[5] Ohata, S., Kawai, Y., Matsuda, T., Hanaoka, G., & Matsuura, K. Re-encryption verifiability: How to detect malicious activities of a proxy in proxy re-encryption. In *CT-RSA 2015: The Cryptographer's Track at the RSA Conference 2015*, 2015, 20-24.

[6] Nunez, D. A. V. I. D. Umbral: a threshold proxy re-encryption scheme. *NuCypher Inc and NICS Lab, University of Malaga, Spain.* 2018.

[7] Pei, H., Yang, P., Li, W., Du, M., & Hu, Z. Proxy Re-Encryption for Secure Data Sharing with Blockchain in Internet of Medical Things. *Computer Networks*, 2024, 245, 110373.

[8] Rodrigues, B., Amorim, I., Silva, I., & Mendes, A. Patient-centric health data sovereignty: an approach using Proxy re-encryption. In *European Symposium on Research in Computer Security*, 2023, (pp. 199-215). Cham: Springer Nature Switzerland.

[9] Liu, S., Qin, H., Taniar, D., Liu, W., Li, Y., & Zhang, J. A certificate-based proxy re-encryption plus scheme for secure medical data sharing. *Internet of Things*, 2023, 23, 100836.

[10] Pei, H., Yang, P., Li, W., Du, M., & Hu, Z. Proxy Re-Encryption for Secure Data Sharing with Blockchain in Internet of Medical Things. *Computer Networks*, 2024, 245, 110373.

[11] Luo, F., Wang, H., Susilo, W., Yan, X., & Zheng, X. Public trace-and-revoke proxy re-encryption for secure data sharing in clouds. *IEEE Transactions on Information Forensics and Security*, 2024.

[12] Chen, W. H., Fan, C. I., & Tseng, Y. F. CCA-Secure Key-Aggregate Proxy Re-Encryption for Secure Cloud Storage. arXiv preprint arXiv:2410.08120, 2024.

[13] Long, Y., Peng, C., Chen, Y., Tan, W., & Sun, J. BFFDT: Blockchain-based Fair and Fine-Grained Data Trading Using Proxy Re-Encryption and Verifiable Commitment. *IEEE Internet of Things Journal*, 2024.

[14] Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., & Zhao, Y. Generic construction of chosen ciphertext secure proxy re-encryption. In *Topics in Cryptology–CT-RSA 2012: The Cryptographers' Track at the RSA Conference 2012*, San Francisco, CA, USA, February 27–March 2, 2012. Proceedings (pp. 349-364). Springer Berlin Heidelberg.

[15] 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 Transactions on Dependable and Secure Computing*, 2021, 19(5), 2907-2919.

[16] 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 of Things Journal*, 2024.

[17] Patil, S. M., & Purushothama, B. R. Non-transitive and collusion resistant quorum controlled proxy re-encryption scheme for resource constrained networks. *Journal of Information Security and Applications*, 2020, 50, 102411.

[18] Chen, X., Liu, Y., Li, Y., & Lin, C. Threshold proxy re-encryption and its application in blockchain. In *Cloud Computing and Security: 4th International Conference*, ICCCS 2018, Haikou, China, June 8–10, 2018, Revised Selected Papers, Part IV 4 (pp. 16-25). Springer International Publishing.

[19] Cao Z, Wang H, & Zhao Y. AP-PRE: Autonomous path proxy re-encryption and its applications. *IEEE Transactions on Dependable and Secure Computing*, 2017, 16(5): 833-842.

[20] Weng, J., Chen, M., Yang, Y., Deng, R., Chen, K., & Bao, F. CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. *Science China Information Sciences*, 2010 53, 593-606.

[21] B. Libert, & D. Vergnaud, Unidirectional chosen-ciphertext secure proxy re-encryption, *International Workshop on Public Key Cryptography*, 2008, pp. 360–379.

[22] Cascudo, I., David, B., Garms, L., & Konring, A. YOLO YOSO: Fast and Simple Encryption and Secret Sharing in the YOSO Model. In *ASIACRYPT*, 2022, pp. 651-680.

[23] Cascudo, I., & David, B. SCRAPE: Scalable Randomness Attested by Public Entities. In *ACNS*, 2017, pp. 537-556.

[24] Gentry, C., Halevi, S., & Lyubashevsky, V. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2022, (pp. 458-487). Cham: Springer International Publishing.
[25] Shamir A. How to share a secret. *Communications of the ACM*, 1979, 22(11):612-613.
[26] Buterin, V. Ethereum white paper. 2013.
[27] Libert, B., & Vergnaud, D. Tracing malicious proxies in proxy re-encryption. In *Pairing-Based Cryptography–Pairing 2008: Second International Conference*, Egham, UK, September 1-3, 2008. Proceedings 2 (pp. 332-353). Springer Berlin Heidelberg.
[28] Guo, H., Zhang, Z., Xu, J., & Xia, M. Generic traceable proxy re-encryption and accountable extension in consensus network. In *Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security*, Luxembourg, September 23–27, 2019, Proceedings, Part I 24 (pp. 234-256). Springer International Publishing.
[29] Lin, Z., Zhou, J., Cao, Z., Dong, X., & Choo, K. K. R. Generalized autonomous path proxy re-encryption scheme to support branch functionality. *IEEE Transactions on Information Forensics and Security*, 2023.
[30] Zhao, F., Weng, J., Xie, W., Li, M., & Weng, J. HRA-secure attribute-based threshold proxy re-encryption from lattices. *Information Sciences*, 2024, 655, 119900.
[31] Bethencourt, J., Sahai, A., & Waters, B. Ciphertext-policy attribute-based encryption. In*2007 IEEE symposium on security and privacy (SP'07)*, 2007, May (pp. 321-334). IEEE.
[32] Bellare, M., Keelveedhi, S., & Ristenpart, T. Message-locked encryption and secure deduplication. In *Annual international conference on the theory and applications of cryptographic techniques*, 2013, (pp. 296-312). Berlin, Heidelberg: Springer Berlin Heidelberg.
[33] Chaum, D., & Pedersen, T. P. Wallet databases with observers. In *CRYPTO*, 1992, pp. 89-105.
[34] Fiat, A., & Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1968, pp. 186-194.
[35] Heidarvand, S., & Villar, J. L. Public verifiability from pairings in secret sharing schemes. In *SAC*, 2009, pp. 294-308.
[36] H. Chen, M. Pendleton, L. Njilla, & S. Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys*, 2020, 53(3):1-43.
[37] Guo, H., Zhang, Z., Xu, J., An, N., & Lan, X. Accountable proxy re-encryption for secure data sharing. *IEEE Transactions on Dependable and Secure Computing*, 2018, 18(1), 145-159.
[38] Maiti, S., Misra, S., & Mondal, A. CBP: Coalitional-Game-Based Broadcast Proxy Re-Encryption in IoT. *IEEE Internet of Things Journal*, 2023, 10(17), 15642-15651.
[39] Zhang, L., Kan, H., & Huang, H. Patient-centered cross-enterprise document sharing and dynamic consent framework using consortium blockchain and ciphertext-policy attribute-based encryption. In *Proceedings of the 19th ACM International Conference on Computing Frontiers*, 2022, pp. 58-66.
[40] Bao, F., Deng, R. H., & Zhu, H. Variations of diffie-hellman problem. In *International conference on information and communications security*, 2003, pp. 301-312.

APPENDIX

A. Decision Diffie-Hellman and Divisible Decision Diffie-Hellman Assumptions

Let $\mathbb{G}$ is a cyclic group of prime order p with generator g. Decision Diffie-Hellman (DDH) problem is defined as the task of determining whether the equation $T = g^{ab}$ holds given (g, g^a, g^b, T), where $a, b \in \mathbb{Z}_p$. The DDH assumption is defined as the assertion that no probabilistic polynomial-time (PPT) adversary has a non-negligible advantage in solving the DDH problem.

Divisible Decision Diffie-Hellman (DDDH) [40] problem is defined as the task of determining whether the equation $T = g^{(b/a)}$ holds given (g, g^a, g^b, T), where $a, b \in \mathbb{Z}_p$. The DDDH assumption is defined as the assertion that no PPT adversary has a non-negligible advantage in solving the DDDH problem.

B. Security Definition

To formalize the CPA security notion for PRE schemes, we first introduce the following possible oracles, which together model the capabilities of an adversary $\mathcal{A}$:

- Public key oracle $\mathcal{O}_{pk}(j)$: Takes public parameter j as input, run $\mathsf{PRE.KeyGen}$, and return pk_j to $\mathcal{A}$.
- Private key oracle $\mathcal{O}_{sk}(pk_j)$: Returns the private key sk_j corresponding to pk_j to $\mathcal{A}$.
- Re-encryption key oracle $\mathcal{O}_{rk}(pk_B, pk_A, \{pk_i\})$: Runs $\mathsf{PRE.ReKeyGen}$, and returns $\{ckFrag_i\}_{i\in[n]}$ to $\mathcal{A}$.
- Re-encryption oracle $\mathcal{O}_{re}(pk_A, \{ckFrag_i\}_{i\in[n]}, C)$: Runs $\mathsf{PRE.ReEnc}$, and returns the first level ciphertext C' to $\mathcal{A}$.

Remark 1: Note that in the CPA analysis of our PRE scheme, neither the generation of the first level challenge ciphertext nor the generation of the second level challenge ciphertext involves the secret key of the proxy party. This can be clearly seen from the $\mathsf{PRE.Enc}_1$ and $\mathsf{PRE.Enc}_2$ algorithms. To simplify the proof process, we assume that the public-private key pairs $\{(pk_i, sk_i)\}_{i\in[n]}$ for the proxies have been pre-generated, and that the proxies have not been corrupted by the adversary (since the CPA proof focuses on the security of the ciphertext). Additionally, the public keys of proxies are known to the adversary. Therefore, we will omit the part involving the $\mathcal{O}_{pk}$ and $\mathcal{O}_{sk}$ oracles that generates the key pairs for the proxies in the following discussion. As for the case of proxies corruption, we will discuss it later.

1) Security of second level ciphertext: We define a game between a PPT adversary $\mathcal{A}$, who operates in two phases $find$ and $guess$, and the challenger $\mathcal{B}$. Let the identity of the challenged delegator be denoted as A^*, and the identity of the user outside the proxies (i.e., the delegators A and delegatees B) be denoted by j. The game can be described with the following experiment:

Experiment 1: $\mathrm{Exp}^{\textbf{\textit{IND-2PRE-CPA}}}_{\Gamma,\mathcal{A}}(\lambda, t, n, l)$

1: $par \leftarrow \mathsf{PRE.Setup}(1^\lambda, t, n, l)$
2: $(pk_{A^*}, pk_B, M_0, M_1, st) \leftarrow \mathcal{A}^{\mathcal{O}_{pk},\mathcal{O}_{sk},\mathcal{O}_{rk},\mathcal{O}_{re}}_{find}(par)$, where $|M_0| = |M_1|$, and st is some state information
3: $d \xleftarrow{R} \{0, 1\}$
4: $C^* \leftarrow \mathsf{PRE.Enc}_2(pk_{A^*}, M_d)$
5: $d' \leftarrow \mathcal{A}^{\mathcal{O}_{pk},\mathcal{O}_{sk},\mathcal{O}_{rk},\mathcal{O}_{re}}_{guess}$
6: return d'

During the above experiment, the following requirements should be satisfied:

- $\mathcal{A}$ can not issue the private key query $\mathcal{O}_{sk}(pk_{A^*})$;
- $\mathcal{A}$ can not issue the private key query $\mathcal{O}_{sk}(pk_B)$, and the re-encryption key query $\mathcal{O}_{rk}(pk_B, pk_{A^*}, \{pk_i\}_{i\in[n]})$ at the same time;
- $\mathcal{A}$ can not simultaneously issue the re-encryption query $\mathcal{O}_{re}(pk_{A^*}, \{ckFrag_i\}_{i\in[n]}, C^*)$ and the private key query $\mathcal{O}_{sk}(pk_B)$.

Then, with respect to the above experiment $\mathrm{Exp}^{\textbf{\textit{IND-2PRE-CPA}}}_{\Gamma,\mathcal{A}}(\lambda, t, n, l)$, we define $\mathcal{A}$'s advantage against ***IND-2PRE-CPA*** security as follows:

$$\mathrm{Adv}^{\textbf{\textit{IND-2PRE-CPA}}}_{\Gamma,\mathcal{A}}(\lambda, t, n, l) = |\Pr[d' = d] - 1/2|.$$

*Definition 1 (**IND-2PRE-CPA**):* We say that our proposed PRE scheme Γ is $(t, q_{pk}, q_{sk}, q_{rk}, q_{re}, \epsilon)$-***IND-2PRE-CPA*** secure if there no t-time adversary $\mathcal{A}$ such that $\mathrm{Adv}_{\Gamma,\mathcal{A}}^{\textbf{\textit{IND-2PRE-CPA}}}(\lambda, t, n, l) \geq \epsilon$, where q_{pk}, q_{sk}, q_{rk}, and q_{re} are the numbers of queries to oracles $\mathcal{O}_{pk}, \mathcal{O}_{sk}, \mathcal{O}_{rk}$, and $\mathcal{O}_{re}$, respectively.

2) Security of first level ciphertext:

*Definition 2 (**IND-1PRE-CPA**):* For the first level ciphertext of our scheme, if it is ***IND-CPA*** secure, i.e., the first level ciphertext is indistinguishable against any PPT adversary $\mathcal{A}$, we say that our PRE scheme is ***IND-1PRE-CPA*** secure.

C. Security Proof

The ***IND-2PRE-CPA*** security for our scheme is asserted by the following theorem.

Theorem 3: Under the divisible decision Diffie-Hellman assumption (DDDH assumption) in standard model, our proposed PRE scheme as shown in Figure 1 is ***IND-2PRE-CPA*** secure. The scenario of the ***IND-2PRE-CPA*** proof is depicted in Figure 17.

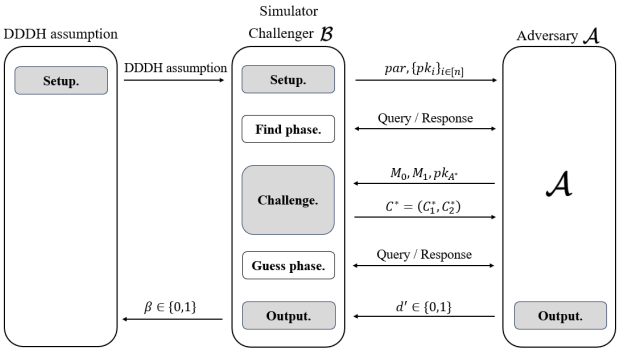

Fig. 17: The scenario of the ***IND-2PRE-CPA*** proof

$Proof$: First, we assume the key derivation function KDF is a pseudorandom function and the symmetric encryption algorithm AES is secure, which ensures that the adversary cannot use known information to obtain valid decryption results. Besides, we assume $\mathcal{H}$ is collision resistant. Then, let $\mathcal{A}$ be an adversary who can break the $(t, q_{pk}, q_{sk}, q_{rk}, q_{re}, \epsilon)$-***IND-2PRE-CPA*** security with non-negligible advantage ϵ, and we can construct a challenger $\mathcal{B}$ using $\mathcal{A}$ to break the DDDH assumption also with non-negligible probability.

We set a properly-distributed tuple $(g, g^a, g^b, T) \in \mathbb{G}^3 \times \mathbb{G}_T$ as challenger $\mathcal{B}$'s input, which is a DDDH instance. If $T = g^{b/a}$, $\mathcal{B}$ tries to output $\beta = 1$; Otherwise, $\mathcal{B}$ tries to output $\beta = 0$. The ***IND-2PRE-CPA*** game between adversary $\mathcal{A}$ and $\mathcal{B}$ can be described as following:

- **Setup.** By *Remark* 1, $\mathcal{B}$ generates the public parameters par and the proxies' key pairs $\{(pk_i, sk_i)\}_{i\in[n]}$ leveraging PRE.Setup algorithm and PRE.KeyGen algorithm, respectively in this phase. And the challenger $\mathcal{B}$ then gives par and $\{pk_i\}_{i\in[n]}$ to adversary $\mathcal{A}$ and maintains the private keys in private. Moreover, $\mathcal{B}$ randomly selects a delegator A^* who will be challenged by adversary $\mathcal{A}$. During the game, $\mathcal{B}$ maintains two tables $\mathcal{PK}$ and $\mathcal{RK}$ to record related information of the public keys and re-encryption keys, respectively. They are initially empty.
- **Find phase.** In this phase, $\mathcal{A}$ can issue queries in the ***IND-2PRE-CPA*** game. And $\mathcal{B}$ answers the queries for $\mathcal{A}$ as follows:
  - Public key query $\mathcal{O}_{pk}(j)$: $\mathcal{B}$ selects a random value $x_j \in \mathbb{Z}_p$, and flips a random coin $c_j \in \{0, 1\}$. Note that $\Pr[c_j = 0] = \Pr[c_j = 1] = \frac{1}{2}$. If $c_j = 0$, $\mathcal{B}$ sets $pk_j = (g)^{x_j}$. Otherwise, $\mathcal{B}$ sets $pk_j = (g^a)^{x_j}$. Then, $\mathcal{B}$ records (pk_j, x_j, c_j) in table $\mathcal{PK}$ and returns pk_j to $\mathcal{A}$.
  - Private key query $\mathcal{O}_{sk}(pk_j)$: $\mathcal{B}$ first accesses the tuple (pk_j, x_j, c_j) from the table $\mathcal{PK}$. If $c_j = 1$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns x_j to $\mathcal{A}$.
  - Re-encryption key query $\mathcal{O}_{rk}(pk_B, pk_A, \{pk_i\})$: The challenger $\mathcal{B}$ first accesses tuples (pk_A, x_A, c_A) and (pk_B, x_B, c_B) from $\mathcal{PK}$, where A and B denote identity of any delegator and delegatee, respectively. Then, $\mathcal{B}$ generates the re-encryption keys for the adversary $\mathcal{A}$ according to the following cases:
    * $c_A = 1 \wedge c_B = 1$: $\mathcal{B}$ aborts.
    * $c_A = 1 \wedge c_B = 0$: It means $sk_A = a \cdot x_A$ and $sk_B = x_B$, $\mathcal{B}$ proceeds as follows:
      · Compute g^s with a random value s.
      · Execute GS.Share to generate the shares of g^s and encrypt them by $pk_A^{(sk_i + sk_B)}$ to get re-encryption keys $\{ckFrag_i\}_{i\in[n]}$.
      · Compute $m^* = \mathcal{H}(pk_A, pk_B, \{pk_i, ckFrag_i\})$, $V = \Pi_{i=1}^n ckFrag_i^{v_i \cdot m^*(\alpha_i)}$, $U = \Pi_{i=1}^n pk_A^{(sk_i + sk_B) \cdot v_i \cdot m^*(\alpha_i)}$.
      · Generate the proof π_{sh} leveraging DLEQ.Proof.

      Then, $\mathcal{B}$ returns $(\{ckFrag_i\}_{i\in[n]}, \pi_{sh})$ to the adversary $\mathcal{A}$ and records the tuple $(\{kFrag_i, ckFrag_i\}_{i\in[n]}, \pi_{sh}, s, A, B)$ in table $\mathcal{RK}$.
    * $c_A = 0$: It means that $sk_A = x_A$. $\mathcal{B}$ runs PRE.ReKeyGen$(pk_B, sk_A, pk_A, \{pk_i\}_{i\in[n]}, s)$ with a random value s and returns $(\{ckFrag_i\}_{i\in[n]}, \pi_{sh})$ to the adversary $\mathcal{A}$. $\mathcal{A}$ can leverage PRE.ReKeyVerify to verify the validity of the re-encryption keys. Then, $\mathcal{B}$ records the tuple $(\{kFrag_i, ckFrag_i\}_{i\in[n]}, \pi_{sh}, s, A, B)$ in table $\mathcal{RK}$.
  - Re-encryption query $\mathcal{O}_{re}(pk_A, \{ckFrag_i, sk_i\}, C)$: $\mathcal{B}$ first checks whether the corresponding tuple $(\{kFrag_i, ckFrag_i\}_{i\in[n]}, \pi_{sh}, s, A, B)$ is in table $\mathcal{RK}$. If not, $\mathcal{B}$ returns "$\perp$" indicating an invalid query. Otherwise, $\mathcal{B}$ parses C as (C_1, C_2) and accesses (pk_A, x_A, c_A) and (pk_B, x_B, c_B) from table $\mathcal{PK}$. $\mathcal{B}$ then generates the re-encryption ciphertext (i.e., the first level ciphertext) for $\mathcal{A}$ according to the following cases:
    * $c_A = 1 \wedge c_B = 1$: $\mathcal{B}$ aborts.
    * $c_A = 1 \wedge c_B = 0$: It means $sk_A = a \cdot x_A$ and $sk_B = x_B$, $\mathcal{B}$ first accesses the tuple $(\{kFrag_i, ckFrag_i\}_{i\in[n]}, \pi_{sh}, s, A, B)$ corresponding to the re-encryption keys $\{ckFrag_i\}$

and computes $C'_{2i} = pk_A^{sk_B} \cdot kFrag_i$, and the NIZK proof π_{rei}. Then, $\mathcal{B}$ returns $C' = (C_1, \{C'_{2i}\}_{i\in[n]})$ and the corresponding NIZK proofs $\{\pi_{rei}\}_{i\in[n]}$ to $\mathcal{A}$. $\mathcal{A}$ can verify the validity of the first level ciphertext leveraging PRE.ReEncVerify.

  - $c_A = 0$: It means $sk_A = x_A$. $\forall i \in [n]$, $\mathcal{B}$ computes $C'_{2i} = ckFrag_i / pk_i^{sk_A}$, and the NIZK proof π_{rei}. Then, $\mathcal{B}$ returns $C' = (C_1, \{C'_{2i}\}_{i\in[n]})$ and the corresponding NIZK proofs $\{\pi_{rei}\}_{i\in[n]}$ to $\mathcal{A}$. $\mathcal{A}$ can verify the validity of the first level ciphertext leveraging PRE.ReEncVerify.

- **Challenge.** When $\mathcal{A}$ judges that the $find$ phase is over, he outputs a public key pk_{A^*} and two messages M_0 and M_1 of equal length. $\mathcal{B}$ responds as follows:
  - Access $(pk_{A^*}, x_{A^*}, c_{A^*})$ from $\mathcal{PK}$. If $c_{A^*} = 0$, $\mathcal{B}$ aborts. Otherwise, it means that $sk_{A^*} = a \cdot x_{A^*}$, and $\mathcal{B}$ proceeds to execute the rest steps.
  - Randomly select $d \in \{0, 1\}$. Set $C_1^* = \mathsf{AES.Enc}(\mathsf{KDF}(T), M_d)$, $C_2^* = (g^b)^{x_{A^*}}$. Obviously, if $T = g^{b/a}$, C^* is indeed a valid challenge ciphertext under public key pk_{A^*}. To see this, setting $s^* = b/a$, we have:
    $C_1^* = \mathsf{AES.Enc}(\mathsf{KDF}(T), M_d) = \mathsf{AES.Enc}(\mathsf{KDF}(g^{s^*}), M_d)$
    $C_2^* = (g^b)^{x_{A^*}} = (g^a)^{x_{A^*}\cdot(b/a)} = (g^a)^{x_{A^*}\cdot s^*} = pk_{A^*}^{s^*}$
    Refer to the statement in [20], when T is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext C^* is independent of d in $\mathcal{A}$'s view.
- **Guess phase.** In this phase, $\mathcal{A}$ can also issue queries in the ***IND-2PRE-CPA*** game and $\mathcal{B}$ answers these queries for $\mathcal{A}$ as in the **Find phase**.
- **Output.** Finally, $\mathcal{A}$ outputs a guess $d' \in \{0, 1\}$. If $d' = d$, $\mathcal{B}$ outputs 1, in which case it indicates that $T = g^{b/a}$, and (g, g^a, g^b, T) is a DDDH tuple; Otherwise, $\mathcal{B}$ outputs 0 indicating that T is a random element in $\mathbb{G}_T$.

Next, we begin to analyze the simulation. Obviously, the simulations of oracle $\mathcal{O}_{pk}$ is perfect since the event of $\mathcal{B}$ aborting during the simulation does not occur. We denote the event of $\mathcal{B}$ aborting during the simulation of oracles $\mathcal{O}_{sk}$, $\mathcal{O}_{rk}$, $\mathcal{O}_{re}$ or in **Challenge** phase by **Abort**. When $c_A = 1$, $\mathcal{B}$ aborts during the query to the private key oracle $\mathcal{O}_{sk}$, and when $c_{A^*} = 0$, $\mathcal{B}$ aborts during **Challenge** phase. So, the probability of the event of $\mathcal{B}$ aborting during the simulation of oracle $\mathcal{O}_{sk}$ or in **Challenge** phase is $\frac{1}{2}$. When $c_A = 1 \wedge c_B = 1$, $\mathcal{B}$ aborts during the query to the re-encryption key query $\mathcal{O}_{rk}$ and the re-encryption query $\mathcal{O}_{re}$. So, the probability of the event of $\mathcal{B}$ aborting during the simulation of oracles $\mathcal{O}_{rk}$ or $\mathcal{O}_{re}$ is $\frac{1}{4}$. Consequently, we have $\Pr[\neg\textbf{Abort}] = (\frac{1}{2})^{q_{sk}+1} \cdot (\frac{3}{4})^{q_{rk}+q_{re}}$. Let $\epsilon_{\mathcal{B}}$ denote the advantage of $\mathcal{B}$ to break the DDDH assumption, we have $\epsilon_{\mathcal{B}} \geq \epsilon \cdot (\frac{1}{2})^{q_{sk}+1} \cdot (\frac{3}{4})^{q_{rk}+q_{re}}$, where ϵ is the non-negligible probability with which $\mathcal{A}$ can break the ***IND-2PRE-CPA*** of our PRE scheme.

Thus, Theorem 3 is proved.

The ***IND-1PRE-CPA*** security for our scheme is asserted by the following theorem.

Theorem 4: Under the DDDH and the DDH assumptions, our proposed PRE scheme as shown in Figure 1 is ***IND-1PRE-CPA*** secure.

$Proof$: For the first level ciphertext C', we can parse it as two parts: C_1, which is included in the second level ciphertext, and $\{C'_{2i}\}_{i\in[n]}$, which is constructed analogously to the output of the PVSS.Share algorithm. From Theorem 3, we know that C_1 is ***IND-CPA*** secure under the DDDH assumption. Now, for the second field of the first level ciphertext $\{C'_{2i}\}_{i\in[n]}$, it is analogous to the decrypted shares held by shareholders in DHPVSS [22], with the key distinction that $\{C'_{2i}\}_{i\in[n]}$ remains encrypted using the delegatee B's key. This ensures that the adversary $\mathcal{A}$ cannot recover the original secret from $\{C'_{2i}\}_{i\in[n]}$. Under the DDH asumption, DHPVSS satisfies ***IND1-Secrecy***, meaning that the adversary cannot distinguish between two different shared secrets before recovering the original secret. Consequently, our PRE scheme inherits this property. Specifically, under the DDH assumption, given $\{C'_{2i}\}_{i\in[n]}$, the adversary $\mathcal{A}$ cannot distinguish between two different shared secrets (since they cannot recover the orginal secret from $\{C'_{2i}\}_{i\in[n]}$), which satisfies the definition of ***IND-CPA***. Thus, Theorem 4 is proved.

Regarding collusion attacks, we consider the following two cases. The first case involves a collusion attack by at least t proxies without the delegatee; the second case is a collusion attack involving at most $t-1$ proxies together with the delegatee.

Theorem 5: Even if an external adversary $\mathcal{A}$ controls more than t proxies to conduct a collusion attack, he cannot obtain the original plaintext data of the delegator A.

$Proof$: First, from Theorem 3 and Theorem 4, we know that our PRE scheme is ***IND-CPA*** secure. Therefore, the only way for an external adversary $\mathcal{A}$ to access the plaintext from the ciphertext is to recover the key seed used in the symmetric encryption. In our scheme, the key seed is shared among n proxies as a secret in the DHPVSS scheme. However, unlike the original DHPVSS scheme, we also use the delegatee's public key, in addition to the delegator's private key (as the dealer) and the proxies' public keys (as the shareholders), when encrypting the shares. This guarantees that proxies will not be able to access the key seed from the re-encryption keys (as encrypted shares). Therefore, even if the number of proxies controlled by the external adversary $\mathcal{A}$ reaches the threshold, he still cannot access the key seed. Theorem 5 is proved.

Theorem 6: Even if an external adversary $\mathcal{A}$ controls at most $t-1$ proxies and the delegatee B to conduct a colluding attack, he cannot obtain the original plaintext data of the delegator A. This property can be referred to as **Fault tolerance**.

$Proof$: Based on the analysis of Theorem 5, it is evident that when the external adversary $\mathcal{A}$ controls the delegatee B and $t-1$ proxies, he can access $t-1$ shares of the key seed. However, due to the ***IND1-Secrecy*** property of PVSS, the adversary $\mathcal{A}$ cannot extract any information about the seed key from these $t-1$ shares. Therefore, Theorem 6 is proved.