Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.047975

ARTICLE

# Blockchain-Based Key Management Scheme Using Rational Secret Sharing

**Xingfan Zhao[1], Changgen Peng[1,2,*], Weijie Tan[2] and Kun Niu[1]**

[1]State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, 550025, China

[2]Guizhou Big Data Academy, Guizhou University, Guiyang, 550025, China

*Corresponding Author: Changgen Peng. Email: cgpeng@gzu.edu.cn

Received: 23 November 2023 Accepted: 23 January 2024 Published: 25 April 2024

**ABSTRACT**

Traditional blockchain key management schemes store private keys in the same location, which can easily lead to security issues such as a single point of failure. Therefore, decentralized threshold key management schemes have become a research focus for blockchain private key protection. The security of private keys for blockchain user wallet is highly related to user identity authentication and digital asset security. The threshold blockchain private key management schemes based on verifiable secret sharing have made some progress, but these schemes do not consider participants' self-interested behavior, and require trusted nodes to keep private key fragments, resulting in a narrow application scope and low deployment efficiency, which cannot meet the needs of personal wallet private key escrow and recovery in public blockchains. We design a private key management scheme based on rational secret sharing that considers the self-interest of participants in secret sharing protocols, and constrains the behavior of rational participants through reasonable mechanism design, making it more suitable in distributed scenarios such as the public blockchain. The proposed scheme achieves the escrow and recovery of personal wallet private keys without the participation of trusted nodes, and simulate its implementation on smart contracts. Compared to other existing threshold wallet solutions and key management schemes based on password-protected secret sharing (PPSS), the proposed scheme has a wide range of applications, verifiable private key recovery, low communication overhead, higher computational efficiency when users perform one-time multi-key escrow, no need for trusted nodes, and personal rational constraints and anti-collusion attack capabilities.

**KEYWORDS**

Blockchain; smart contract; rational secret sharing; key management

## 1 Introduction

Key Escrow [1] refers to the storage of decryption keys in a key escrow system (also known as a key recovery system), and in some cases, authorizes third parties to access these keys, which is a key management mechanism for statutory escrow agency. Government and law enforcement agencies' needs to access encrypted data and recover commercial data have made key escrow a popular research topic in the 1990s [2]. Key Escrow faces two challenges: the trust level of the key escrow agent and the ability to protect the keys [3]. It is easy for a single key escrow party to collaborate with illegal organizations to obtain confidential data, facing widespread user doubts. A single key escrow node also faces the risk of a single point of failure. Therefore, split key escrow has gradually become a new direction for the development of key escrow [4]. In this stage, Shamir [5] proposed partial key escrow for DES keys, Lenstra et al. [6] proposed a time-constrained key escrow system, and Micali et al. [7] proposed shared random functions and key escrow schemes. However, based on Shamir [8] and Blakley [9], independently proposed the concept of key dispersed management, using threshold secret sharing as the representative threshold cryptography as the theoretical basis, comprehensively considering the security and implementation difficulty of the scheme, studying threshold key escrow schemes has become the main trend of key split escrow. Therefore, the key to designing a threshold key escrow scheme that meets actual needs lies in studying and developing corresponding threshold secret sharing technology.

Shamir's threshold secret sharing and other early-stage secret sharing schemes assumed that the secret distributor and participants were honest without considering deceptive behavior. In response, Chor et al. [10] proposed the concept of verifiable secret sharing (VSS), which added a verification algorithm to the traditional secret sharing scheme. However, this scheme required an interactive process, resulting in low efficiency. Bai et al. [11] proposed a verifiable quantum secret sharing scheme based on d-dimensional Greenberger-Horne-Zeilinger (GHZ) state and analyzed its security against three main quantum attacks: interception retransmission attack, entanglement measurement attack, and honest participant attack. Gentry et al. [12] proposed a non-interactive public verifiable secret sharing (PVSS) scheme for large-scale distributed networks, which can be extended to have hundreds or even thousands of participants to support secure computing in large-scale distributed systems. However, these VSS schemes only consider honest participants and suspected fraudulent participants, without considering the motivations and benefits of each participant. For large-scale institutions like governments or enterprises, key escrow participants are usually bound by political orders, business contracts, and other forms of compulsion, which do not need to consider the self-interested behavior of each participant in the threshold key escrow scheme. Whereas, for online distributed systems like blockchain that target a large number of network users, the traditional threshold key escrow scheme does not have such constraints, resulting in participants who do not upload or upload false key shares still being able to obtain reconstructed keys.

Halpern et al. [13] considered the selfish behavior of participants and introduced Game theory to secret sharing and secure multi-party computation. They proposed a rational cryptographic protocol represented by rational secret sharing and rational secure multi-party computation for the first time. However, traditional rational secret sharing schemes rely on synchronous communication environments and cannot be applied to distributed systems such as blockchains with asynchronous communication. Kol et al. [14] used cryptographic techniques such as oblivious transfer and secure multi-party computation to construct the first rational secret sharing scheme suitable for asynchronous communication. In addition, Zhang et al. [15] introduced rational cryptography into quantum secret sharing research and proposed a rational quantum secret sharing scheme based on GHZ states, which was implemented on the IBM quantum computer [16].

The wide application of blockchain and smart contracts in the Internet of Things has enabled the fulfillment of resource sharing and automated transactions without trusted centers [17]. The public key cryptography technology widely used in blockchain is highly related to the security of user private keys, user asset security, data security, and identity certification security. As there is no trusted third party in the blockchain system, individuals need to locally store their private keys. Once the device where the private key is stored is lost, data is damaged, or even hacked by hackers, it will lead to the loss or leakage of user data and assets. In this context, blockchain private key escrow solutions based on threshold key management technology have become a research hotspot for blockchain private key protection. However, the currently proposed key management schemes for blockchain private key recovery still adopt traditional VSS technology, without considering the rational self-interest behavior of the escrow users [18]; the successfully implemented rational secret sharing schemes on blockchain require the secret sharing participants to deliver deposits in advance [19]. These schemes do not meet the individual rational constraint in mechanism design, limiting the application and promotion of corresponding blockchain threshold key escrow design. The main contributions of this paper are as follows:

We proposed a threshold multi-key escrow management scheme that satisfied individual rational constraints and incentive compatibility: We first designed a reputation mechanism for participants during the key reconstruction phase with the process of uploading and verifying private key fragments and reporting. Then, according to the ratio of contribution to promote the cooperation and behavioral norms of blockchain users, we also proposed a benefit distribution mechanism that relied on a fair process of cooperative gaming.

We constructed the whole multi-key escrow and reconstruction process based on verifiable random functions (VRF) and analyzed and demonstrated the collusion resistance, security verification ability, incentive compatibility and individual rational constraints of the threshold key escrow scheme through utility function analysis. We also compared it with existing blockchain recoverable key management schemes.

We used the smart contracts to complete a multi-wallet key escrow simulation, demonstrating the ability of the scheme to manage wallet keys in Ethereum. We analyzed the performance and costs of the scheme, providing possibilities for achieving a distributed rational key management system for blockchain.

## 2 Related Work

### *2.1 Blockchain and Smart Contract*

Nakamoto et al. [20] proposed Bitcoin, a virtual encrypted currency, and the blockchain transaction structure using Proof of Work and timestamp construction in 2008, which marked the birth of decentralized blockchain technology. Blockchain combines with cryptographic technology ensures traceability, immutability, non-repudiation, and non-counterfeiting of transactions, supporting data security sharing and large-scale collaborative computing, protecting user identity and confidential data privacy, and is suitable for high-privacy and secure distributed application scenarios. Traceability refers to the recording of transaction changes in a blockchain in chronological order, immutability and non-repudiation refer to the inability to modify and deny data written into a blockchain, and non-counterfeiting refers to the inability to forge transactions that can be verified by miners and the entire transaction change record. Blockchain uses hash functions, digital signatures, and distributed consensus fault tolerance to increase the difficulty and cost of attackers falsifying, forging, and denying data operations, enhancing data security.

Szabo proposed the smart contract, which is a computerized transaction protocol for executing contract terms [21]. In the context of blockchain, smart contracts serve as scripts on blockchain and are stored on the blockchain. As smart contracts are located on the blockchain, they have their unique addresses. We trigger their execution by sending transactions to their addresses. Subsequently, smart contracts will be executed independently and automatically on every node in the network according to the data contained in the triggered transaction. Smart contracts on blockchain possess uniqueness, transparency, predictability, and monitoring and tracking capabilities, and have fostered the concept of a decentralized autonomous organization, allowing blockchain to achieve general computing among users without a trusted central node's supervision [17]. Raj et al. [22] proposed a blockchain smart contract scheme for supply chain management, which replaces the trust intermediary with smart contracts, shortens the payment process, reduces the communication cost, and improves the supply chain transaction efficiency.

## *2.2 Research on the Security of Blockchain Wallet and Threshold Key Escrow*

Blockchain digital wallets provide users with a convenient way to manage private and public keys, while also enabling them to complete digital asset transfers and transactions, and record details of all transactions. The security of the wallet directly relates to the user's asset security. The comparison of various blockchain key management schemes is shown in Table 1. The existing common wallet solution types include software wallets [23], hardware wallets [24], and custodial wallets [25], but they all have security risks in terms of centralized private key storage. To address these issues, decentralized secure threshold wallet schemes have emerged, which can split private keys into multiple parts and store them on different nodes, thereby achieving distributed storage and protection of private keys, while also allowing private keys to be restored. Therefore, this secure threshold wallet scheme is considered an effective protection plan.

**Table 1:** Comparison of various blockchain key management schemes

| Scheme type | Disadvantages |
|---|---|
| Software wallets [23] | The private key is stored locally, with low security and the risk of a single point of failure. |
| Hardware wallets [24] | Poor portability and the risk of a single point of failure. |
| Custodial wallets [25] | Reliance on trusted third parties and the risk of a single point of failure. |
| Threshold signature wallets [26] | Personal wallet private key escrow and recovery are not supported. |
| Threshold escrow wallets [18] | Require trusted nodes or servers. |

A threshold secret sharing-based threshold signature wallet scheme such as [26] does not support the escrow and recovery of personal user wallet private keys, while a threshold wallet management scheme that meets the needs of personal key escrow and recovery, such as [18], is only applicable to consortium blockchain applications, which requires trusted central nodes and identity authentication mechanisms. In addition, the aforementioned threshold wallet schemes have not taken into account the rational self-interest behavior of participants, resulting in high costs and limited applicability, which cannot be satisfied to the safe escrow needs of public blockchain user wallet keys such as Ethereum. Therefore, designing a private key escrow smart contract based on rational secret sharing is beneficial for achieving the escrow and recovery of public blockchain user private keys in an environment without trusted nodes.

## 3 Preliminaries

### 3.1 Bilinear Mapping

*Definition 1: Let $G_1$ and $G_2$ be the additive cyclic group and multiplicative cyclic group of a prime number p, respectively, let g be a generator of G. The mapping $e\colon G_1 \times G_1 \rightarrow G_2$ is bilinear if it satisfies the following properties:*

*1) Bilinearity: For any $m, n \in G_1$ and $a, b \in Z_p{}^*$, exists $e\left(m^a, n^b\right) = e\left(m, n\right)^{ab}$.*

*2) Non-degeneracy: $e\left(g, g\right) \neq 1$.*

*3) Computability: There exists an efficient algorithm to compute $e\left(m, n\right)$ for any $m, n \in G_1$.*

*Definition 2: Elliptic Curve Discrete Logarithm Problem (DCELP): Given an elliptic curve E defined over a finite field $F_p$, with a base point $P \in E\left(F_q\right)$ of order n. $Q \in E\left(F_q\right)$ where $Q = IP, I \in n$, it is difficult to compute $I$.*

*Definition 3: Computational Diffie-Hellman (CDH) Given $aP, bP, cP \in G_1$, it is difficult to compute $abP$ for any $a, b \in Z_P{}^*$.*

*Definition 4: Bilinear Diffie-Hellman Problem Given $aP, bP, cP \in G_1$, it is difficult to compute $e\left(P, P\right)^{abc}$ for any $a, b, c \in Z_P{}^*$. This problem relies on the difficulty of the CDH problem in $G_1$.*

*Definition 5: Decisional Bilinear Diffie-Hellman Inversion Assumption, also known as q-DBDHI problem. An adversary with polynomial time capability cannot distinguish between $\left(g, g^x, \ldots, g^{\left(x^q\right)}, e\left(g, g\right)^{\frac{1}{x}}\right)$ and $\left(g, g^x, \ldots, g^{\left(x^q\right)}, \Gamma\right)$ with a non-negligible advantage, where $x \in Z_P{}^*$, $\Gamma \in G_2$.*

### 3.2 Game Theory Related Knowledge

#### *3.2.1 Incentive Compatibility*

Incentive compatibility refers to the consistency between individual incentive mechanisms and overall benefits in a game. It includes Bayesian incentive compatibility and dominant strategy incentive compatibility. It should be noted that Bayesian incentive compatibility is a weak concept in games, while dominant strategy incentive compatibility is a strong concept. If a game is dominant strategy incentive compatible, then it must be Bayesian incentive compatible.

In mechanism design, Bayesian incentive compatibility is considered a sufficient and necessary condition for participants to speak the truth. A game is Bayesian incentive compatible if and only if, for any player *i*, any possible private type $\theta_i$, and any other players' strategies $\sigma_{-i}$, the following condition is satisfied:

$$\forall\theta_i, \forall\sigma_i, \forall\sigma_i'\colon U_i\left(\sigma_i, \sigma_{-i}, \theta_i\right) \geq U_i\left(\sigma_i', \sigma_{-i}, \theta_i\right) \tag{1}$$

where $U_i\left(\sigma_i, \sigma_{-i}, \theta_i\right)$ represents player *i*'s utility when their private type is $\theta_i$, they choose strategy $\sigma_i$ (the true strategy), and the other players choose strategies $\sigma_{-i}$.

In a game, if each player adopts their dominant strategy regardless of the other players' actions, the game is dominant strategy incentive compatible. Mathematically, a game is dominant strategy incentive compatible if and only if for any player *i*, any other player's strategy $\sigma_{-i}$, the following condition is satisfied:

$$\forall\sigma_i, \forall\sigma_i'\colon U_i\left(\sigma_i, \sigma_{-i}\right) \geq U_i\left(\sigma_i', \sigma_{-i}\right) \tag{2}$$

where $U_i\left(\sigma_i, \sigma_{-i}\right)$ represents player *i*'s utility when all players adopt strategy $\sigma_i, \sigma_{-i}$.

### 3.2.2 *Individual Rationality*

In game theory, individual rationality refers to each player choosing strategies that are beneficial to them in pursuit of their own maximum interests. This is also known as the assumption of rational participants. In mechanism design, individual rationality means designing a game mechanism to ensure that each participant has the motivation to participate and will not obtain a worse result due to participation.

Mathematically, individual rationality satisfies the following conditions:

For each participant $i$, individual rationality requires that their returns after participating in the mechanism are not less than their worst returns from leaving the mechanism, i.e.,

$$\forall \sigma_i, \forall \sigma_{-i}: U_i(\sigma_i, \sigma_{-i}) \geq U_i(\varnothing) \tag{3}$$

where $U_i(\varnothing)$ represents the returns of participant $i$ when they do not participate in the mechanism.

### 3.2.3 *Fairness of Cooperative Game and Shapley Value Principle*

Fairness is a concept to measure the justice of a game. In Game theory, fairness usually refers to whether a game satisfies some fair principles, such as the symmetry principle in a zero-sum game and the Shapley value principle in a cooperative game.

In a cooperative game, the Shapley value principle means that each player should get the corresponding reward for his contribution to the game. Specifically, the Shapley value refers to the average contribution of a player to all possible coalitions. If a player's Shapley value is lower than other players, this situation is also unfair.

### 3.2.4 *Rational Fairness*

In Game Theory, rational fairness refers to the participants in a game mechanism pursuing their own interests (rationality) while also obtaining fair results. Rational fairness requires participants to consider not only their own interests but also the interests of other participants when choosing strategies, in order to achieve a fair balance.

Rational fairness should satisfy the following conditions:

For each participant, rational fairness requires that the chosen strategy is an optimal response, i.e.,

$$\widehat{\sigma_i^*} \in BR_i(s_{-i}) \tag{4}$$

### 3.2.5 *Resisting Collusion Attack*

In Game Theory, a Collusion Attack refers to participants secretly collaborating or cooperating to devise strategies for their own benefits while disregarding the interests of other participants. This collusion may distort the results of the game, sacrificing fairness and balance. Resisting Collusion Attack refers to a feature of strategic or mechanism design that aims to prevent participants from forming collusion or cooperation that could harm the fairness or balance of the game.

Reference [27] proposed a definition of a computable anti-collusion equilibrium: in a game $\Gamma = \left(\{A_i\}_{i=1}^n, \{u_i\}_{i=1}^n\right)$, let $T$ represent a collusion group and $k$ be a secure parameter of a cryptographic protocol. Assuming that for each collusion group, the non-collusive strategy is $a_T$, the collusive strategy is $a'_T$, and the non-collusion group's strategy is $a_{-i}$, we call the strategic combination $a = (a_1, \ldots, a_n) \in A$ a computable anti-collusion equilibrium if for $P_i \in T$, there exists a negligible function $\varepsilon(k)$ such that the game remains in

$$u_i\left(a'_T(k), a_{-T}(k)\right) \leq u_i\left(a_T(k), a_{-T}(k)\right) + \varepsilon(k) \tag{5}$$

### *3.3 Verifiable Random Function*

Verifiable Random Function (VRF) [28] is a random function with verification capabilities. It can generate an output and allow a verifier to verify that the output was generated from a specific input and key while maintaining the randomness and unpredictability of the output. Reference [29] proposed an efficient verifiable random function based on bilinear pairs. The scheme includes the following components:

a) Public-private key generation module $Gen\left(1^k\right)$: Input $1^k$, output $SK = s, PK = g^s$.

b) Encryption module $F_{SK}(x)$: Input $SK, x$, output $y = F_{SK}(x) = e(g, g)^{\frac{1}{x+SK}}$.

c) Evidence module $\pi_{SK}(x)$: Input $SK, x$, output $\pi = \pi_{SK}(x) = g^{\frac{1}{x+SK}}$.

d) Verification module $V_{PK}(x, y, \pi)$: Determine whether $y$ is the output corresponding to $x$, input $(PK, x, y, \pi)$, verify whether $e(g \cdot pk, \pi) = e(g, g)$ and $y = e(g, \pi)$ are both true, and output 1 if both are true, otherwise output 0.

## 4 Multi-Key Management Scheme Based on Rational Secret Sharing

Based on the anti-collusion rational secret sharing [27], and taking into account the practical situation of blockchain key escrow, the current and long-term benefits of secret-sharing participants, and the advantages of blockchain smart contracts, a multi-private key management scheme based on rational secret sharing is proposed as follows.

### *4.1 System Parameters*

The parameters involved in the proposed scheme are shown in Table 2.

**Table 2:** The parameters of the proposed key management scheme

| Notation | Description |
|---|---|
| $P = \{P_1, \ldots, P_n\}$ | $n$ participants |
| $s_0, s_1, \ldots, s_{p-1}$ | Contains $p$ keys in ciphertext form, each with a length of 256 bits |
| $Rv_i$ | The reputation value of participant $i$ |
| $Cv_i$ | The contribution value of participant $i$ |
| $h(.)$ | Collision-resistant hash function, SHA-256 hash function is used in this paper |
| $\|\|$ | Concatenating operation |
| $\oplus$ | XOR operation |
| $V_j^{r*}$ | The XOR value of key $j$ |
| $F_q$ | The finite field with $q$ elements, where $q$ is a large prime number. The system operates on the finite field, i.e., all parameters in the system are elements of the finite field |
| $M$ | The cost of a private key escrow contract with $n$ participants |
| $M_1$ | The total reward amount for successful reconstruction of the secret by the participants |

(Continued)

**Table 2 (continued)**

| Notation | Description |
|---|---|
| $M_2$ | Advance payment for smart contract expenses |
| $M_{2(r)}$ | The remaining value of the advance payment for the contract's round $r$ of refactoring process upon completion |
| $L$ | The maximum cost value of the contract during each round of refactoring |
| $\lambda$ | Determine the parameters of the geometric distribution based on the estimated profit of the participants |
| $r^*$ | Security parameter generation based on geometric distribution |
| $Add_i$ | The blockchain user address of Participant $i$ |
| $m_r$ | The number of participants in round $r$ of the key reconstruction phase |
| $sk_i$ | The personal private key obtained by Participant $i$ from the key escrow user for reconstructing |
| $pk_i$ | The public key used to verify the information uploaded by participant $i$ during the key reconstruction phase |

## 4.2 Key Generation Phase

This algorithm is executed by the system administrator to generate public key and private key of $P_i$. After executing this algorithm, the administrator will exit this program permanently. Especially, the setup algorithm inputs security parameter $1^k$ and generates the public key $pk_i$ and the private key $sk_i$ by using Shamir's (t, n)-threshold secret sharing scheme for $P_i$. Moreover, blockchain administrators use multiple servers provided by the blockchain platform to secretly obtain randomly generated two-dimensional point coordinates, reconstruct Lagrange polynomials, and generate private keys by substituting additional random values. Its purpose is to avoid the risk of a single point of failure during the user's local private key generation process.

## 4.3 Key Escrow Negotiation Phase

As shown in Fig. 1, the key escrow user sets a threshold value $t$ according to its own needs, determines the number of participants $n$ to sign the key escrow agreement based on the active degree of the blockchain nodes and $p$ encrypted keys $s_0, s_1, \ldots, s_{p-1}$, then negotiates the cost of the escrow contract $M = M_1 + M_2$, determines the parameters of the geometric distribution $\lambda$ based on the estimated benefits of the participants (see Section 6.3 for details) and generates a secure parameter $r^* \in Z^+$ ($Z^+$ is a positive integer set) based on this geometric distribution. The key escrow user learns the addresses of all participants and establishes secure channels with each participant. Send the private key $sk_i$ $(i = 1, 2, \ldots, n)$ to the participant $i$ through a secure channel and upload the public data to the smart contract: a) $pk_i$ $(i = 1, 2, \ldots, n)$, b) $M = M_1 + M_2$.

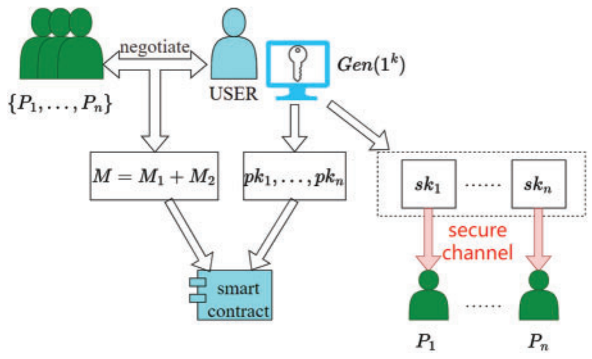

**Figure 1:** Key escrow negotiation process

### *4.4 Key Sharing Phase*

*Step 1*: Key Escrow User uses the two-dimensional coordinate points $\left(h\left(Add_1||r^*\right), F_{sk_1}\left(r^*\right)\right), (h(Add_2||r^*), F_{sk_2}\left(r^*\right)), \ldots, \left(h\left(Add_n||r^*\right), F_{sk_n}\left(r^*\right)\right)$ and $(0,s_0), (1,s_1), \ldots, \left(p-1, s_{p-1}\right)$ determines a $n+p-1$ degree polynomial as follows:

$$G(x) = \sum_{i=1}^{n} F_{sk_i}\left(r^*\right) \prod_{j=1,j\neq i}^{n} \frac{x - h\left(Add_j||r^*\right)}{h\left(Add_i||r^*\right) - h\left(Add_j||r^*\right)}$$

$$\prod_{j=0}^{p-1} \frac{x-j}{h\left(Add_i||r^*\right) - j} + \sum_{i=0}^{p-1} s_i \prod_{j=0,j\neq i}^{p-1} \frac{x-j}{i-j} \prod_{j=1}^{n} \frac{x - h\left(Add_j||r^*\right)}{i - h\left(Add_j||r^*\right)} \bmod q$$

$$= a_0^{r*} + a_1^{r*}x + a_2^{r*}x^2 + \cdots a_{n+p-1}^{r*}x^{n+p-1} \tag{6}$$

Let $V_0^{r*} = s_0 \oplus G(0), V_1^{r*} = s_1 \oplus G(1), \ldots, V_{p-1}^{r*} = s_{p-1} \oplus G(p-1)$

*Step 2*: To avoid duplicate numerical pairs causing failure during the key reconstruction phase, the key escrow user selects $n+p-t$ smallest integers $d_1, d_2, \ldots, d_{n+p-t}$ from $[p, q-1] - \{h(Add_i||r) \mid i = 1,2,\ldots,n; r = 1,2,\ldots,r^*\}$.

*Step 3*: Upload public data to the smart contract: a) $(d_1, G(d_1)), (d_2, G(d_2)), \ldots, \left(d_{n+p-t}, G\left(d_{n+p-t}\right)\right)$, b) $V_j^{r*} = s_j \oplus G(j)\ (j = 0,1,\ldots,p-1)$ and c) $h\left(a_k^{r*}\right)\ (k = 0,1,\ldots,n+p-1)$.

*Step 4*: The key escrow user deletes $sk_i\ (i = 1,2,\ldots,n)$, $r^*$, $s_0, s_1, \ldots, s_{p-1}$ and only retains the public information: $pk_i\ (i = 1,2,\ldots,n)$, $(d_1, G(d_1)), (d_2, G(d_2)), \ldots, \left(d_{n+p-t}, G\left(d_{n+p-t}\right)\right)$, $V_j^{r*} = s_j \oplus G(j)\ (j = 0,1,\ldots,p-1)$ and $h\left(a_k^{r*}\right)\ (k = 0,1,\ldots,n+p-1)$.

The process of key sharing phase is shown in Fig. 2.

### *4.5 Key Reconstruction Phase*

*Step 1*: The contract indicates that round $r\ (r = 1,2,\ldots,r^*)$ begins (the initial value $r$ is 1), and $i\ (i = 1,2,\ldots,m_r, t \leq m_r \leq n)$ participants must input $y_i^r = F_{sk_i}(r) = e(g,g)^{\frac{1}{r+sk_i}}$, $\pi_i^r = \pi_{sk_i}(x) = g^{\frac{1}{r+sk_i}}$ within the prescribed upload time $T_1$ into the smart contract. Repeated uploads or $y_i^r, \pi_i^r$ with a reputation value $Rv_i < 0$ are considered invalid inputs.

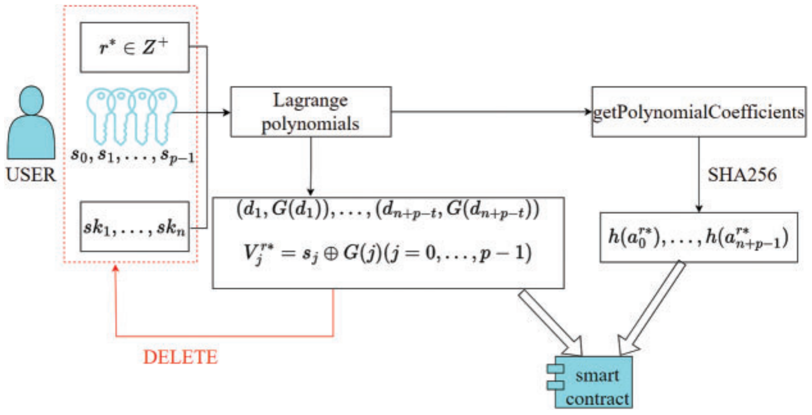
$r^* \in Z^+$
USER
$s_0, s_1, \ldots, s_{p-1}$
$sk_1, \ldots, sk_n$
Lagrange polynomials
getPolynomialCoefficients
SHA256
$(d_1, G(d_1)), \ldots, (d_{n+p-t}, G(d_{n+p-t}))$
$V_j^{r*} = s_j \oplus G(j)(j = 0, \ldots, p-1)$
$h(a_0^{r*}), \ldots, h(a_{n+p-1}^{r*})$
DELETE
smart contract

**Figure 2:** Key sharing process

*Step 2*: During the verification time $T_2$, all participants can make their own judgment about whether $e\left(g^r \cdot pk_i, \pi_i^r\right) = e(g,g)$ and $y_i^r = e\left(g, \pi_i^r\right)$ both are valid or not. During this period, all participants can report anyone. If someone reports, the reported person will be marked as $fl$ and sorted by order. For the same reported person, the reporter with the highest reputation value (or if the reputation values are the same, the one who reported earlier) will be marked as $ld$ and proceed to *Step 3*. If no one reports, proceed to *Step 4* directly.

*Step 3*: Performs automatic contract verification of the reported account in order of submission. If the verification result is 0, the reputation value of the reported account will be $Rv_{fl} - 1$, and the contribution value will be $Cv_{ld} - 1$. At the same time, the reporter will be rewarded with a reputation value of $Rv_{ld} + 1$ and a contribution value of $Cv_{ld} + 1$. If the reporter $ld$ has not uploaded $y_{ld}^r$ and $\pi_{ld}^r$ within $T_1$, they can resubmit after the *Step 3* contract automatic verification is completed and return to *Step 2*. If the verification result is 1, the reputation value of the reported account will remain unchanged, and the contribution value will be $Cv_{fl} + 1$. At the same time, the reporter will be penalized with a reputation value of $Rv_{ld} - 1$ and a contribution value of $Cv_{ld} - 1$.

*Step 4*: Within $T_3$, the key escrow user calls the local resource to verify the remaining unreported participants one by one. If the participant $i$'s verification result is 0, the user will report feedback to the contract. After being verified by the contract, $Cv_i - 1$, $m_r - 1$. If the result is 1, it will continue to verify the participant $i + 1$. Complete the verification of all participants and $m_r \geq t$, then proceed to *Step 5*.

*Step 5*: The key escrow user and participants use $\left(h\left(Add_1||r\right), F_{sk_1}(r)\right), \left(h\left(Add_2||r\right), F_{sk_2}(r)\right), \ldots,$ $\left(h\left(Add_{m_r}||r\right), F_{sk_{m_r}}(r)\right)$ and $\left(d_1, G\left(d_1\right)\right), \left(d_2, G\left(d_2\right)\right), \ldots, \left(d_{n+p-t}, G\left(d_{n+p-t}\right)\right)$ to reconstruct the following polynomial:

$$P(x) = \sum_{u=1}^{m_r} F_{sk_u}(r) \prod_{j=1, j\neq u}^{m_r} \frac{x - h\left(Add_j||r\right)}{h\left(Add_u||r\right) - h\left(Add_j||r\right)}$$

$$\prod_{k=1}^{n+p-m_r} \frac{x - d_k}{h\left(Add_u||r\right) - d_k} + \sum_{i=1}^{n+p-m_r} G\left(d_i\right) \prod_{k=1, k\neq i}^{n+p-m_r} \frac{x - d_k}{d_i - d_k} \prod_{u=1}^{m_r} \frac{x - h\left(Add_u||r\right)}{d_i - h\left(Add_u||r\right)} \bmod q$$

$$= b_0^r + b_1^r x + b_2^r x^2 + \cdots b_{n+p-1}^r x^{n+p-1}, \tag{7}$$

Then judge whether $h\left(b_k^r\right) = h\left(a_k^{r*}\right)(k = 0, 1, \ldots, n + p - 1)$ is valid. If no one declares the reconstruction is correct (i.e., declares it to be invalid or remains silent), then proceed to *Step 7*. If at least the key escrow user declares the reconstruction is correct, proceed to *Step 6*.

*Step 6*: The contract automatically calculates and determines whether $h\left(b_k^r\right) = h\left(a_k^{r*}\right)(k = 0, 1, \ldots, n + p - 1)$ is valid. If it is valid, the secrets $s_i = P(i) \oplus V_i^{r*}(i = 0, 1, \ldots, p - 1)$ reconstruction are completed, the contract assigns each participant with the corresponding contribution value $A_i = \frac{Cv_i}{\sum_{u=1}^{n} Cv_u} \cdot M_1$, and $M_{2(r)}$ is returned to the key escrow user, terminating the contract. If it is not valid, the invalid participants $j$ will be executed $Rv_j + 1$ and proceed to *Step 7*.

*Step 7*: Check if $M_{2(r)}$ in the current contract is not less than the maximum cost $L$ of the next round. If it is, proceed to *Step 8*; if not, notify the key escrow user to make up the payment (not less than $L - M_{2(r)}$) within $T_5$. If the payment is made promptly, proceed to *Step 8*. If not, the contract will be assigned to each participant with the corresponding contribution value $A_i = \frac{Cv_i}{\sum_{u=1}^{n} Cv_u} \cdot M_1$, and $M_{2(r)}$ will be returned to the key escrow. The contract will be terminated.

*Step 8*: The contract executes $r + 1$, and return to *Step 1*.

The process of the key reconstruction phase is shown in Fig. 3. The algorithm pseudocode mainly executed by the smart contract is shown in Fig. 4.

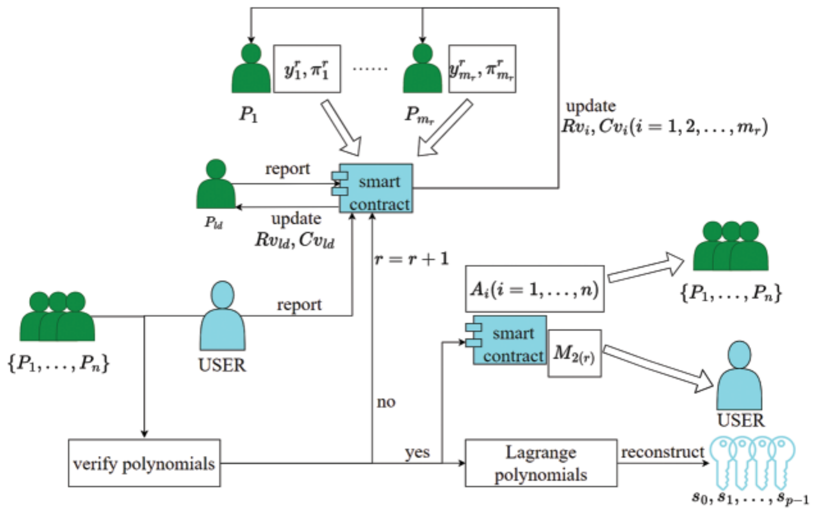

**Figure 3:** Key reconstruction process

**Algorithm 1** getPolynomialCoefficients

```
function getPolynomialCoefficients(Point[], prime)
    n ← points.length;
    if n ≤ 0 then
        return "No points are added yet";
    coefficients[] ← new uint256[n];
    for i ← 0 to n do
        numerator ← 1;
        denominator ← 1;
        for j ← 0 to n do
            if j ≠ i then
                numerator ← mulmod(numerator, submod(0, points[j].x), prime);
                denominator ← mulmod(denominator, submod(points[i].x, points[j].x), prime);
        coefficients[i] ← mulmod(points[i].y, mulmod(numerator, inverseMod(denominator, prime), prime), prime);
    return coefficients;
```

**Algorithm 2** modular arithmetic

```
function submod(uint256 a, uint256 b)
    return (a + prime - b) % prime;

function inverseMod(uint256 a, uint256 m)
    if a = 0 then
        return "Cannot calculate inverse for zero";
    m0 ← m;
    t ← 0;
    q ← 0;
    x0 ← 0;
    x1 ← 1;
    while a > 1 do
        q ← a / m;
        t ← m;
        m ← a % m;
        a ← t;
        t ← x0;
        x0 ← x1 - q * x0;
        x1 ← t;
    if x1 < 0 then
        x1 ← x1 + m0;
    return uint256(x1);

function mulmod(uint256 a, uint256 b, uint256 m)
    return (a * b) % m;
```

**Figure 4:** Pseudocode for the main algorithms in the smart contract

## 5 Scheme Analysis

### 5.1 Correctness Analysis

Participants and smart contracts can verify the accuracy of the input $y_i^r = F_{sk_i}(r) = e(g,g)^{\frac{1}{r+sk_i}}$ and $\pi_i^r = \pi_{sk_i}(x) = g^{\frac{1}{r+sk_i}}$ by checking whether $e\left(g^r \cdot pk_i, \pi_i^r\right) = e(g,g)$ and $y_i^r = e\left(g, \pi_i^r\right)$ are both true.

Proof:

$$e\left(g^r \cdot pk_i, \pi_i^r\right) = e\left(g^r \cdot g^{sk_i}, g^{\frac{1}{r+sk_i}}\right) = e\left(g^{r+sk_i}, g^{\frac{1}{r+sk_i}}\right) = e(g,g)^{(r+sk_i)\frac{1}{r+sk_i}} = e(g,g)$$

$$y_i^r = e(g,g)^{\frac{1}{r+sk_i}} = e\left(g, g^{\frac{1}{r+sk_i}}\right) = e\left(g, \pi_i^r\right)$$

### 5.2 Smart Contract Participant Utility Analysis

For *Step 1*, the contract setting does not allow participants to upload duplicate data to avoid "free-riding" behavior in a round of polynomial reconstruction, which would improperly increase the contribution value $Cv$ and allow participants to obtain rewards that do not belong to them. Not allowing participants satisfied $Rv < 0$ to upload data reflects punishment for malicious participants (those who often engage in dishonest behavior).

For *Step 2* to *Step 4*, participants $ld$ and participants $fl$ can be viewed as the following extended game (a complete information dynamic game, which means participants know each other's payoffs, the available actions, and the order in which moves are made):

Participant set $N\{ld, fl\}$, $ld$'s strategy space $\left\{\text{“}report\text{”}, \text{“}silence\text{”}\right\}$, $fl$'s strategy space $\left\{\text{“}true\text{”}, \text{“}false\text{”}\right\}$. The corresponding utility functions are:

$$\begin{cases} U_{ld}\left(\sigma_{ld} = \text{"}report\text{"}, \sigma_{fl} = \text{"}false\text{"}\right) = \{Cv_{ld} + 1, Rv_{ld} + 1\} = a \\ U_{ld}\left(\sigma_{ld} = \text{"}report\text{"}, \sigma_{fl} = \text{"}true\text{"}\right) = \{Cv_{ld} - 1, Rv_{ld} - 1\} = e \\ U_{ld}\left(\sigma_{ld} = \text{"}silence\text{"}, \sigma_{fl} = \text{"}false\text{"}\right) = \{Cv_{ld}, Rv_{ld}\} = c \\ U_{ld}\left(\sigma_{ld} = \text{"}silence\text{"}, \sigma_{fl} = \text{"}true\text{"}\right) = \{Cv_{ld}, Rv_{ld}\} = c \\ U_{fl}\left(\sigma_{ld} = \text{"}report\text{"}, \sigma_{fl} = \text{"}false\text{"}\right) = \left\{Cv_{fl} - 1, Rv_{fl} - 1\right\} = e \\ U_{fl}\left(\sigma_{ld} = \text{"}report\text{"}, \sigma_{fl} = \text{"}true\text{"}\right) = \left\{Cv_{fl} + 1, Rv_{fl}\right\} = b \\ U_{fl}\left(\sigma_{ld} = \text{"}silence\text{"}, \sigma_{fl} = \text{"}false\text{"}\right) = \left\{Cv_{fl} - 1, Rv_{fl}\right\} = d \\ U_{fl}\left(\sigma_{ld} = \text{"}silence\text{"}, \sigma_{fl} = \text{"}true\text{"}\right) = \left\{Cv_{fl}, Rv_{fl}\right\} = c \end{cases}$$

The game tree for two players is shown in Fig. 5 ($a > b > c > d > e$):

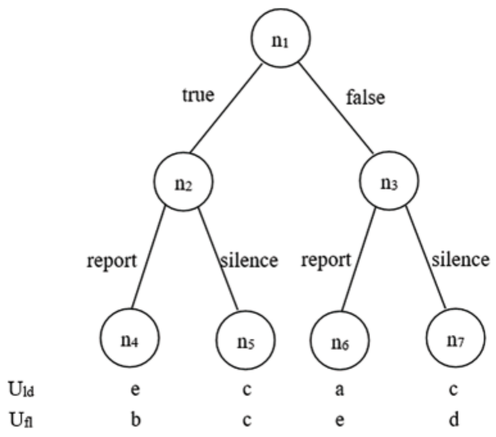

**Figure 5:** Participants' game in key reconstruction phase

It is easy to find that the game has two optimal choices (true, silence) and (false, report) for participant $ld$. For the participant $fl$, is a strictly dominant strategy $\sigma_{fl} = \text{"}true\text{"}$, and from the perspective of individual rationality, the participant is more inclined to honestly upload their own data. When malicious participants upload false data, the contract can effectively use the reporting mechanism to punish malicious participants promptly. Ultimately, a unique pure strategy Nash equilibrium in the sequential game could be found: $\left(\sigma_{fl} = \text{"}true\text{"}, \sigma_{ld} = \text{"}silence\text{"}\right)$, honestly uploading data in the data uploading stage and not making false reports in the verification stage.

For *Step 5* to *Step 8*, there is actually no effective contribution value gain among participants, but only a reputation value reward is given to honest declaration behavior when the key escrow user declares deception to the contract. In principle, the key escrow user has no motivation to deceive, because the key escrow user needs to pay the contract verification overhead, and declaring success only increases the contract polynomial reconstruction overhead by 1 time when the polynomial is not reconstructed successfully. If the key escrow user does not declare reconstruction success in $T_4$ when $r = r^*$, only the amount belonging to the user $M_{2(r)}$ will be consumed by the contract until it is not more than one round of the maximum contract overhead value $L$ to stop.

For the entire key escrow protocol, the actual benefits for each participant are $U_i = A_i = \frac{Cv_i}{\sum_{u=1}^{n} Cv_u} \cdot M_1$.

### 5.3 *Analysis of Smart Contract Fairness*

#### 5.3.1 *Fairness Analysis of Cooperative Game*

In this key escrow contract scheme, the Shapley value of the participant $i$ is $Shv_i = \sum_{r=1}^{r^*} \overline{Cv_{ir}} = \sum_{r=1}^{r^*} \left( \frac{1 \cdot m_r}{m_r} - \frac{1 \cdot m_r - Cv_{ir}}{m_r} \right) \frac{m_r}{\sum_{r=1}^{r^*} m_r} = \frac{Cv_i}{\sum_{r=1}^{r^*} m_r} = \frac{Cv_i}{\sum_{u=1}^{n} Cv_u}$, which is proportional to its revenue $A_i = \frac{Cv_i}{\sum_{u=1}^{n} Cv_u} \cdot M_1$, satisfying the fairness of the cooperative game.

#### 5.3.2 *Rational Fairness Analysis*

In this scheme, everyone's choice strategy $\widehat{\sigma_i^*} = \{"true", ("false", "report")\} \in BR_i(s_{-i})$ satisfies the requirement of rational fairness.

### 5.4 *Mechanism Design Analysis*

#### 5.4.1 *Fairness Analysis of Cooperative Game*

In the key escrow contract scheme, $\forall \sigma_i, \forall \sigma_i'$: $U_i \left( \sigma_i = \widehat{\sigma_i^*}, \sigma_{-i} \right) \geq U_i \left( \sigma_i', \sigma_{-i} \right)$, which means the scheme satisfies dominant-strategy incentive compatibility and also satisfies Bayesian incentive compatibility.

#### 5.4.2 *Personal Rational Analysis*

In the key escrow contract scheme, it was found in the utility analysis in Section 5.2 that $U_i = A_i = \frac{Cv_i}{\sum_{u=1}^{n} Cv_u} \cdot M_1 \geq 0 = U_i \{\varnothing\}$, the condition for personal rationality in mechanism design is satisfied.

## 6 Security Analysis

### 6.1 *Unforgeability of VRF*

***Proof***: Assume there exists an algorithm $A$ that can distinguish between a random element of $G_2$ and $y = F_{SK}(x) = e(g,g)^{\frac{1}{x+SK}}$ with $\varepsilon$ advantage (i.e., $\frac{1}{2} + \varepsilon$ probability of success). Then the advantage of the algorithm $B$ constructed using this algorithm $A$ to solve the DBDHI problem should be $\frac{\varepsilon}{2}$ (i.e., $\frac{1+\varepsilon}{2}$ probability of successful solution). Given $\left( g, g^{\alpha}, \ldots, g^{(\alpha^q)}, \Gamma \right)$, $B$ attempts to distinguish whether $\Gamma$ is a random element of $G_2$ or $e(g,g)^{\frac{1}{\alpha}}$. If $\Gamma$ is $e(g,g)^{\frac{1}{\alpha}}$, output 0, otherwise output 1.

First, the challenger sets $G_1$ and $G_2$, determines the bilinear transformation $e$ and generator $g$. Then, the challenger randomly selects $\xi \in \{0,1\}$ without revealing it to $B$. If $\xi = 0$, let $\Gamma = e(g,g)^{\frac{1}{\alpha}}$; if $\xi = 1$, then let $\Gamma \in G_2$.

*Initial phase*: Simulate the algorithm $B$ by running $A$, select a fake VRF value for some input $x_0$, let $\chi = \alpha - x_0$, define a function $m(o) = \sum_{j=0}^{q-1} c_j o^j$, $m'(o) = \sum_{j=0}^{q-2} r_j o^j$, calculate $n = g^{m(\chi)}$, $PK = n^{\chi}$ and $SK = \chi$, and send the public key $PK$ to the attacker $A$.

*Phase 1*: $A$ ask for the VRF value of $x_i \, (x_i \neq x_0)$, the oracle will return $\pi_{SK}(x_i) = g^{\frac{1}{x_i+\chi}}$ and $F_{SK}(x_i) = e(n,n)^{\frac{1}{x_i+\chi}}$ to $A$.

*Challenge phase*: $A$ sends two equal-length values $x_0, x_1$ to $B$. $B$ randomly select $\varphi \in \{0, 1\}$, and generate $\pi_{SK}\left(x_\varphi\right) = g^{\frac{1}{x\varphi+\chi}}$ and $F_{SK}\left(x_\varphi\right) = e(n, n)^{\frac{1}{x\varphi+\chi}}$.

Let $\Gamma^* = \Gamma^{(r^2)} e(g, g)^{\frac{m(\chi)^2-r^2}{\alpha}}$, if $\Gamma = e(g, g)^{\frac{1}{\alpha}}$, then $\Gamma^* = e(n, n)^{\frac{1}{\alpha}}$. $B$ Send $\pi_\varphi, y_\varphi, \Gamma^*$ to $A$. When $\xi = 0$, $\Gamma$ and $\Gamma^*$ have the same distribution; when $\xi = 1$, $\Gamma$ and $\Gamma^*$ are both random numbers.

*Phase 2*: Repeat the process of Phase 1.

*Guessing phase*: $A$ provides a guess value $\varphi'$. If $\varphi' = \varphi$, $B$ outputs $\xi' = 0$. If $\varphi' \neq \varphi$, $B$ outputs $\xi' = 1$. When $\xi = 1$, $A$ obtains random number information with $\Pr[\varphi' \neq \varphi | \xi = 1] = \frac{1}{2}$ and $B$ outputs $\xi' = 1$, $\Pr[\xi' = \xi | \xi = 1] = \frac{1}{2}$. When $\xi = 0$, $A$ obtains valid information with an advantage of $\varepsilon$, so $\Pr[\varphi' = \varphi | \xi = 0] = \frac{1}{2} + \varepsilon$, $B$ outputs $\xi' = 0$, $\Pr[\xi' = \xi | \xi = 0] = \frac{1}{2} + \varepsilon$. The probability of algorithm $B$ cracking q-DBDHI is: $\Pr[\xi' = \xi] = \Pr[\xi = 0]\Pr[\xi' = \xi | \xi = 0] + \Pr[\xi = 1]\Pr[\xi' = \xi | \xi = 1] = \frac{1+\varepsilon}{2}$, i.e., the advantage of algorithm is $\frac{\varepsilon}{2}$. However, there is currently no algorithm with an unignorable advantage in cracking q-DBDHI, so the advantage $\varepsilon$ of the assumed algorithm $A$ is negligible, i.e., the output value of VRF cannot be forged in the q-DBDHI problem. Therefore, using VRF for verification can detect the authenticity and correctness of the data to be verified.

### 6.2 *Threshold Security: No More Than $t - 1$ Participants Cannot Obtain Information about $p$ Encrypted Keys*

Public information $(d_1, G(d_1)), \ldots, \left(d_{n+p-t}, G\left(d_{n+p-t}\right)\right)$ consists of $n + p - t$ numerical pairs, and no more than $n + p - 1$ numerical pairs can be determined when no more than $t - 1$ participants cooperate, making it impossible to determine a polynomial of $n+p-1$ degree as in Eq. (6) and obtain information about the corresponding $p$ encrypted keys.

### 6.3 *Resisting Collusion Attacks*

Based on the threshold security analysis, any colluding party of the colluding group $T \subset [n], |T| \leq t - 1$ cannot determine whether the current round is the $r^*$ round through collusion. If the colluding party wants to obtain the encrypted keys information at this time, they can only guess, with $\beta^T$ the probability of guessing correctly, corresponding to the benefit $U_i^-$; and $1 - \beta^T$ the probability of guessing incorrectly, corresponding to the benefit $U_i^+$; the expected benefit of colluding party $P_i$ for guessing is $E\left(U_i^{T guess}\right) = \beta^T U_i^+ + \left(1 - \beta^T\right) U_i^-$. Suppose the probability of attacking exactly at round $r^*$ is $\lambda^T$, and the benefit of attacking exactly at round $r^*$ is still $U_i^+$, otherwise the benefit of colluding party $P_i$ is $E\left(U_i^{T guess}\right)$. Therefore, the expected benefit of colluding party $P_i$ is $E\left(U_i^T\right) = \lambda^T U_i^+ + \left(1 - \lambda^T\right) E\left(U_i^{T guess}\right)$. If colluding members follow the protocol, the benefit of $P_i$ at this time is $U_i$, so when $U_i > E\left(U_i^T\right)$ (i.e., $U_i > \lambda^T U_i^+ + \left(1 - \lambda^T\right) E\left(U_i^{T guess}\right)$), colluding members have no motivation to deviate from the protocol.

For this private key management protocol, the probability of guessing the plaintext of the private key of the blockchain private wallet is negligible, i.e., $E\left(U_i^{T guess}\right)$ can be approximated to 0. This protocol only needs to ensure $U_i > \lambda^T U_i^+$. In the protocol, $U_i = A_i = \frac{Cv_i}{\sum_{u=1}^{n} Cv_u} \cdot M_1 \geq \frac{M_1}{n}$, if the key escrow user is willing to pay no more than $R$ to the colluding organization through signing a new smart contract for key shares trading when threatened by a collusion attack, then $U_i^+ = \frac{R}{|T|} < \frac{R}{t}$. Therefore, $\lambda^T < \frac{tM_1}{nR}$, in the corresponding protocol, $\lambda \leq \lambda^T$ can meet the requirement of resisting collusion attacks.

## 7 Performance Analysis

### 7.1 Communication Cost

During key escrow agreement, the user uploads all public key data ($512n$ bits), hash values of polynomial coefficients ($256\,(n+p)$ bits), the XOR values of all keys ($256p$ bits), and coordinates of $n+p-t$ two-dimensional points ($512\,(n+p-t)$ bits) to the blockchain. The total bits are $1280n + 1024p - 512t$. For each round, each participant only needs to upload the encrypted information $y_i^r = F_{sk_i}(r) = e(g,g)^{\frac{1}{r+sk_i}}$ and verification information $\pi_i^r = \pi_{sk_i}(x) = g^{\frac{1}{r+sk_i}}$, which are 256 bits and 320 bits. Therefore, the communication overhead of each round of communication between participants and the smart contract is 576 bits. As a result, the total communication cost of our proposed scheme is $576\sum_{1}^{r^*} m_r + 1280n + 1024p - 512t$ bits.

### 7.2 User Communication Overhead

As shown in Table 3, during the key distribution phase, whether it is Ogata's scheme [30], Ra's scheme [31], or the proposed scheme in this paper, it is necessary to transmit the corresponding key shares to $n$ server/participant nodes through secure channels. When the server/participant nodes are honest, Ogata's scheme [30] only needs to communicate and interact during the reconstruction phase without the verification stage of server information. Ra's scheme [31] needs to communicate with each server through an interactive verification method and determine whether the corresponding node is malicious. When there are $e$ malicious nodes among the server/participant nodes, the former cannot correctly reconstruct the corresponding escrow key, while the latter needs to communicate with these $e$ malicious nodes for an additional round during the verification stage. For the proposed scheme in this paper, whether there are malicious nodes or not, in each round of key escrow user node verification, it only needs to obtain the verification information uploaded by each participant from the chain, that is, the communication overhead per round of verification is $O(1)$, and the communication overhead of $r^*$ rounds of verification is $O(r^*) = O\left(\frac{1}{\lambda}\right)$. After $r^*$ rounds of iteration, the user reconstructs successfully locally and communicates with the smart contract deployed on the blockchain once, that is, the communication overhead of the reconstruction phase is $O(1)$. However, for the proposed scheme, the overall communication complexity is $O\left(n^2\right)$ due to the participants broadcasting communication interactions during the reconstruction phase.

**Table 3:** User communication overhead comparison

| Phases | Ogata [30] | Ra et al. [31] | Proposed scheme |
|---|---|---|---|
| Distribution phase | $O(n)$ | $O(n)$ | $O(n)$ |
| Verification phase (Honest nodes) | – | $O(n)$ | $O(r^*)$ |
| Reconstruction phase (Honest nodes) | $O(n)$ | $O(n)$ | $O(1)$ |
| Verification phase (Appearance of malicious nodes) | – | $O(n) + O(e)$ | $O(r^*)$ |
| Reconstruction phase (The malicious node exists) | Fail to reconstruct | $O(n)$ | $O(1)$ |

### 7.3 *Feature Comparison*

As shown in Table 4, the current password-protected secret sharing (PPSS)-based key recovery management schemes for blockchain can meet the key recovery needs of blockchain with semi-honest nodes, such as the consortium blockchain [30,31]. However, these schemes require interactive verification between users and servers and often require synchronous communication. At the same time, they do not consider the rational self-interested behavior of blockchain, and cannot realize key recovery in more blockchain application scenarios such as the public blockchain. Our scheme uses rational secret sharing and the non-interactive verification method Elliptic Curve VRF (ECVRF) to meet all the requirements in the table, thus realizing the key recovery needs of blockchain users in the asynchronous communication environment of the public blockchain.

**Table 4:** Feature comparison

| Schemes | Ogata [30] | Ra et al. [31] | Proposed scheme |
|---|---|---|---|
| Decentralized approach | YES | YES | YES |
| Without honest nodes | Semi-honest | Semi-honest | YES |
| Non-interactive verification | NO | NO | YES |
| Preventing malicious behavior | NO | YES | YES |
| Preventing illegal key recovery | YES | YES | YES |
| Individual rationality | NO | NO | YES |
| Nash equilibrium for sequential games | NO | NO | YES |
| Collusion resistance | YES | YES | YES |
| Fair reconstruction rewards | NOT OFFER | NOT OFFER | YES |

### 7.4 *Computational Efficiency Analysis*

Based on the calculation efficiency analysis in Ra et al. [31] and the key distribution and reconstruction process proposed in the proposed scheme, we compare the computational complexity of Ogata's scheme [30], Ra's scheme [31], and the proposed scheme, as shown in Table 5.

**Table 5:** Computational complexity comparison

| Phases | Ogata [30] | Ra et al. [31] | Proposed scheme |
|---|---|---|---|
| Distribution | $nS$ | $nS + k$ | $(n + p)\,S$ |
| Recovery | $(n + 3k)\,S$ | $(n + 4k)\,S$ | $r^* \left[(n + p)\,S + nT\right]$ |

The computational complexity of Shamir's secret sharing is denoted as $S$, the time complexity of generating random numbers is denoted as $k$, the time complexity of performing one ECVRF execution is denoted as $T$, and the number of executions of the proposed scheme during secret reconstruction is denoted as $r^*$.

Typically, from the perspective of Byzantine fault tolerance, $t \in \left(\frac{2}{3}n, n\right]$, $E\left(r^*\right) = \frac{1}{\lambda} \geq \frac{1}{\lambda T} > \frac{nR}{tM_1} > \frac{nR}{\frac{2}{3}nM_1} = \frac{3R}{2M_1}$. We consider the cost of the managed key service as $M_1$, and the cost of the premium managed key service as $R$, and from a practical point of view, it should satisfy $M_1 < R \leq 2M_1$, so

$E(r^*) = 3$ is more reasonable. The computational complexity of Shamir's secret sharing is within 0.218 s [32], let $r^* = 3$, and the following is the computational time overhead diagram:

As shown in Fig. 6, in the key distribution process, when the user only hosts one key, the computational time overhead of the proposed scheme is higher than Ogata's scheme [30] and Ra's scheme [31]. When the user hosts two keys at once, only in the case of $n = 2$, the computational time overhead of the proposed scheme is the same as Ogata's scheme [30], and in other cases, it is lower than Ogata's scheme [30] and Ra's scheme [31]. When the user hosts three keys at once, the computational time overhead of the proposed scheme is lower than Ogata's scheme [30] and Ra's scheme [31].

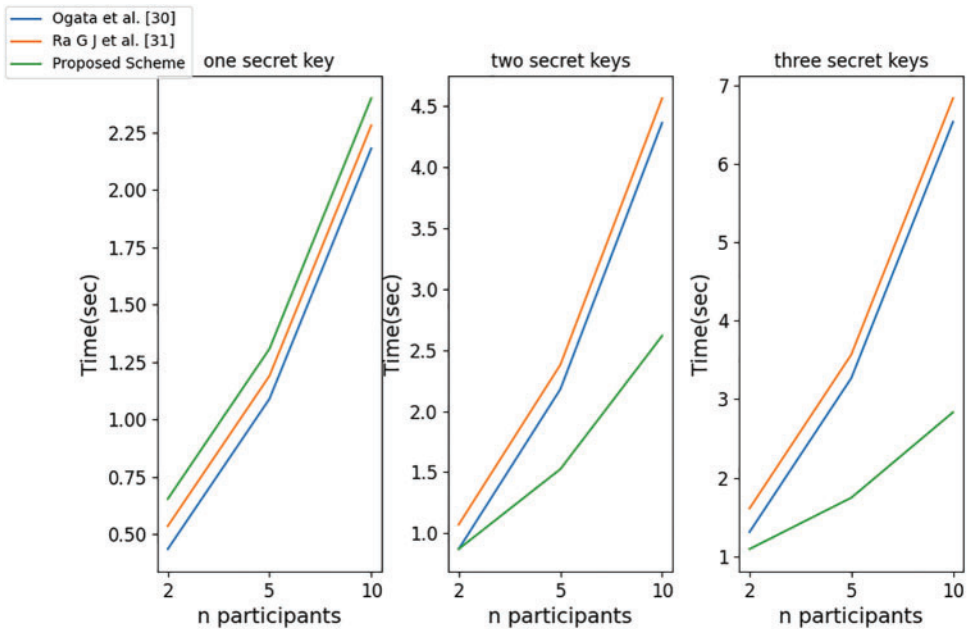

**Figure 6:** Computational efficiency comparison during distribution

As shown in Fig. 7, when the user only hosts one key in the key recovery process, the computational time overhead of the proposed scheme is higher than Ogata's scheme [30] and Ra's scheme [31]. When the user hosts five keys at once, in the case of $n \geq 10$, the computational time overhead of the proposed scheme is lower than Ogata's scheme [30] and Ra's scheme [31]. When the user hosts seven keys at once, in the case of $n \geq 5$, the computational time overhead of the proposed scheme is lower than Ogata's scheme [30] and Ra's scheme [31].

We conducted computational cost simulations on a personal computer with a 12th Gen Intel (R) Core (TM) i7-12700H 2.70 GHz CPU and 16.0 GB RAM, using JavaScript language. The number of participants in the proposed scheme is set to be $n = 1000$ and the number $p$ of private keys that need to be escrowed is denoted as $p = 5$. The computational cost for the negotiation and key sharing stage was 403881 ms, and the computational cost for the key reconstruction stage was 1248453 ms.

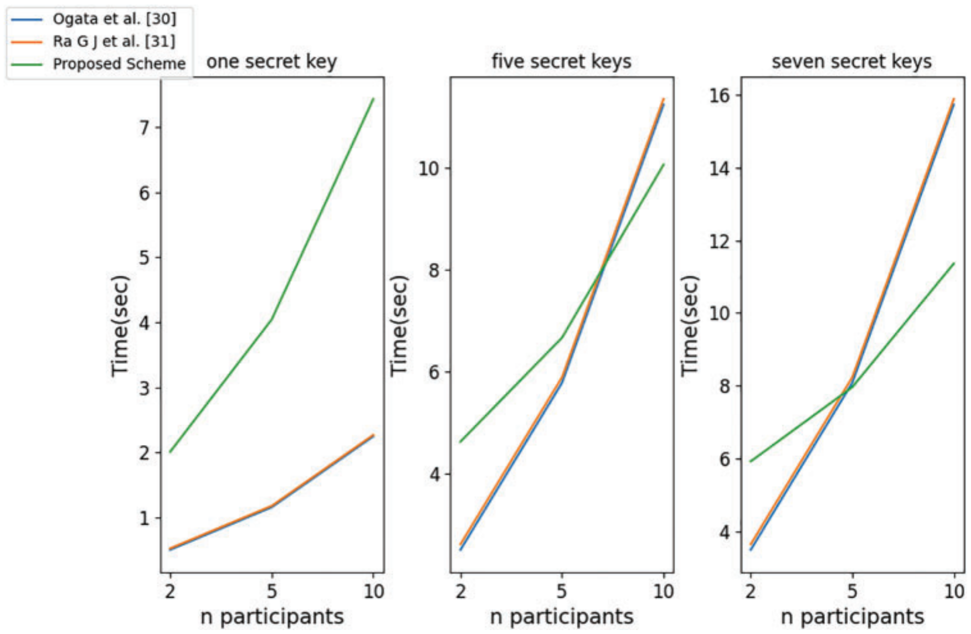

**Figure 7:** Computational efficiency comparison during recovery

### *7.5 Ether Consumption*

Table 6 shows the functions and corresponding Ether costs involved in the smart contract during the refactoring phase of the proposed scheme. However, according to the utility analysis in Section 5.2, participants will comply with the protocol and upload real data without false reporting. In principle, the contract will only execute a polynomial refactoring and hash value verification once, and will not execute the relevant functions of ECVRF (i.e., serial 2 and 3).

**Table 6:** Ether cost

| Serial | Function | Cost in Ether |
|---|---|---|
| 1 | Deploy | 0.047784 |
| 2 | decodeProof | 0.0010178 |
| 3 | verify | 0.058908 |
| 4 | addPoint | 0.0014430 |
| 5 | getPolynomialCoefficients | 0.0027637 (n = 3), 0.0050395 (n = 5), 0.011919 (n = 10), 0.39951 (n = 100), 3.7142 (n = 1000) |
| 6 | SHA256 | 0.000061654 |
| 7 | sethash | 0.000647947 |
| 8 | verifyhash | 0.000071862 |

## 8 Conclusion

We proposed a blockchain private key management scheme based on rational secret sharing, which can achieve one-time escrow and recovery of multiple wallet private keys for blockchain private user. This scheme can be applied to the secure backup scenario of digital assets such as blockchain user wallets and encrypted data. The proposed scheme takes into account the self-interested behavior of participants and satisfies individual rational constraints and incentive compatibility. It promotes the cooperation and behavioral norms of blockchain users by designing a reputation mechanism and a revenue distribution mechanism based on contribution value proportion. In addition, the scheme also designed the whole process of multi-key escrow and reconstruction based on verifiable random functions and demonstrated the collusion resistance, security verification ability, communication overhead, computational overhead, incentive compatibility and individual rational constraints through participants' utility function analysis. Finally, the scheme was simulated on Ethereum smart contracts to prove its feasibility on the public blockchain.

In the future work, in order to make the proposed scheme a long-term blockchain key management protocol, we will consider the evolutionary game of blockchain users, adjust the reputation incentive design in this scheme and improve the efficiency of blockchain private key escrow and recovery. In addition, we will consider extending the management of the blockchain user's private key from the wallet to more application scenarios of blockchain key management.

**Acknowledgement:** I would like to express my deepest gratitude to all those who have supported and guided me throughout the course of this research.

**Funding Statement:** This work was supported in part by the State's Key Project of Research and Development Plan under Grant 2022YFB2701400, in part by the National Natural Science Foundation of China under Grants 62272124 and 62361010, in part by the Science and Technology Planning Project of Guizhou Province under Grant [2020]5017, in part by the Research Project of Guizhou University for Talent Introduction under Grant [2020]61, in part by the Cultivation Project of Guizhou University under Grant [2019]56, in part by the Open Fund of Key Laboratory of Advanced Manufacturing Technology, Ministry of Education under Grant GZUAMT2021KF[01], the Science and Technology Program of Guizhou Province (No. [2023]371).

**Author Contributions:** The authors confirm contribution to the paper as follows: study conception and design: Xingfan Zhao and Changgen Peng; supervision: Changgen Peng; data collection: Kun Niu; analysis and interpretation of results: Weijie Tan; draft manuscript preparation: Xingfan Zhao. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** There is no available data and materials.

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

## References

[1] D. E. Denning and D. K. Branstad, "A taxonomy for key escrow encryption systems," *Commun. ACM*, vol. 39, no. 3, pp. 34–40, 1996. doi: 10.1145/227234.227239.
[2] H. Abelson *et al.*, "The risks of key recovery, key escrow, and trusted third-party encryption," *World Wide Web J.*, vol. 2, no. 3, pp. 241–257, 1997.

[3] C. Boyd, X. Boyen, C. Carr, and T. Haines, “Key recovery: Inert and public,” in *Paradigms in Cryptology—Mycrypt 2016. Malicious and Exploratory Cryptology*. Cham: Springer International Publishing, 2017, pp. 111–126.
[4] M. Bellare and S. Goldwasser, “Verifiable partial key escrow,” in *Proc. 4th ACM Conf. Comput. Commun. Secur.*, 1997, pp. 78–91.
[5] A. Shamir, “Partial key escrow: A new approach to software key escrow,” in *Key Escrow Conf.*, 1995.
[6] A. K. Lenstra, P. Winkler, and Y. Yacobi, “A key escrow system with warrant bounds,” in *Annual Int. Cryptol. Conf.*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 197–207.
[7] S. Micali and R. Sidney, “A simple method for generating and sharing pseudo-random functions, with applications to clipper-like key escrow systems,” in *Annual Int. Cryptol. Conf.*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 185–196.
[8] A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979. doi: 10.1145/359168.359176.
[9] G. R. Blakley, “Safeguarding cryptographic keys,” in *Managing Requirements Knowl., Int. Workshop on, IEEE Comput. Soc.*, 1979, pp. 313.
[10] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, “Verifiable secret sharing and achieving simultaneity in the presence of faults,” in *26th Annual Symp. Foundations of Comput. Sci.*, Portland, OR, USA, 1985, pp. 383–395.
[11] C. M. Bai, S. Zhang, and L. Liu, “Verifiable quantum secret sharing scheme using d-dimensional GHZ state,” *Int. J. Theor. Phys.*, vol. 60, no. 10, pp. 3993–4005, 2021. doi: 10.1007/s10773-021-04955-1.
[12] C. Gentry, S. Halevi, and V. Lyubashevsky, “Practical non-interactive publicly verifiable secret sharing with thousands of parties,” in *Annual Int. Conf. Theory Appl. Cryptographic Tech.*, 2022, pp. 458–487.
[13] J. Halpern and V. Teague, “Rational secret sharing and multiparty computation,” in *Proc. Thirty-Sixth Annual ACM Symp. Theory Comput.*, 2004, pp. 623–632.
[14] G. Kol and M. Naor, “Cryptography and game theory: Designing protocols for exchanging information,” in *Theory Cryptography Conf.*, Berlin, Heidelberg: Springer, 2008, pp. 320–339.
[15] X. Zhang, L. Wang, S. Lin, N. Wang, and L. Hong, “Rational quantum secret sharing scheme based on GHZ state,” *Quantum Inf. Process.*, vol. 22, no. 2, pp. 91, 2023. doi: 10.1007/s11128-022-03739-8.
[16] D. Joy, M. Sabir, B. K. Behera, and P. K. Panigrahi, “Implementation of quantum secret sharing and quantum binary voting protocol in the IBM quantum computer,” *Quantum Inf. Process.*, vol. 19, no. 1, pp. 1–20, 2020. doi: 10.1007/s11128-019-2531-z.
[17] K. Christidis and M. Devetsikiotis, “Blockchains and smart contracts for the internet of things,” *IEEE Access.*, vol. 4, pp. 2292–2303, 2016. doi: 10.1109/ACCESS.2016.2566339.
[18] G. Li, L. You, G. Hu, and L. Hu, “Recoverable private key scheme for consortium blockchain based on verifiable secret sharing,” *KSII. Trans. Internet Inf. Syst. (TIIS)*, vol. 15, no. 8, pp. 2865–2878, 2021.
[19] Z. Chen, Y. Tian, and C. Peng, “An incentive-compatible rational secret sharing scheme using blockchain and smart contract,” *Sci. China Inf. Sci.*, vol. 64, no. 10, pp. 1–21, 2021. doi: 10.1007/s11432-019-2858-8.
[20] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *White Paper*, 2008. Accessed: Sep. 17, 2019. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[21] N. Szabo, “Smart contracts,” 1994. Accessed: Dec. 20, 2018. [Online]. Available: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html
[22] P. V. R. P. Raj, S. K. Jauhar, M. Ramkumar, and S. Pratap, “Procurement, traceability and advance cash credit payment transactions in supply chain using blockchain smart contracts,” *Comput. Ind. Eng.*, vol. 167, no. 8, pp. 108038, 2022. doi: 10.1016/j.cie.2022.108038.
[23] A. R. Thota, P. Upadhyay, S. Kulkarni, P. Selvam, and B. Viswanathan, “Software wallet based secure participation in Hyperledger Fabric networks,” in *2020 Int. Conf. Commun. Syst. Netw. (COMSNETS)*, 2020, pp. 1–6.

[24] D. Park, M. Choi, G. Kim, D. Bae, H. Kim and S. Hong, “Stealing keys from hardware wallets: A single trace side-channel attack on elliptic curve scalar multiplication without profiling,” *IEEE Access*, vol. 11, pp. 44578–44589, 2023.

[25] K. Chalkias, P. Chatzigiannis, and Y. Ji, “Broken proofs of solvency in blockchain custodial wallets and exchanges,” in *Lecture Notes in Computer Science*, Cham: Springer, vol. 13412, 2023.

[26] R. Gennaro, S. Goldfeder, and A. Narayanan, “Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security,” in *Appl. Cryptography Netw. Secur.: 14th Int. Conf.,* Guildford, UK, Springer International Publishing, Jun. 2016, pp. 156–174.

[27] E. Zhang, Q. Sun, and Y. Liu, “Collusion-free rational multi-secret sharing scheme,” *Comput. Sci.*, vol. 42, no. 10, pp. 164–169, 2015.

[28] S. Micali, M. Rabin, and S. Vadhan, “Verifiable random functions,” in *40th Annual Symp. Foundations Comput. Sci. (Cat. No. 99CB37039)*, New York, NY, USA, 1999, pp. 120–130.

[29] Y. Dodis and A. Yampolskiy, “A verifiable random function with short proofs and keys,” in *Int. Workshop Public Key Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 416–431.

[30] W. Ogata, “Improvement of IT-secure password-protected secret sharing,” in *30th Symp. Cryptography Inf. Secur. (SCIS 2013)*, 2013.

[31] G. J. Ra, C. H. Roh, and I. Y. Lee, “A key recovery system based on password-protected secret sharing in a permissioned blockchain,” *Comput. Mater. Conti.*, vol. 65, no. 1, pp. 153–170, 2020. doi: 10.32604/cmc.2020.011293.

[32] S. Sugianto, S. Suharjito, and N. Surantha, “Comparison of secret splitting, secret sharing and recursive threshold visual cryptography for security of handwritten images,” *TELKOMNIKA. (Telecommun. Comput. Electron. Cont.)*, vol. 16, no. 1, pp. 323–333, 2018. doi: 10.12928/telkomnika.v16i1.6632.