Information Sciences 546 (2021) 166–176

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fair hierarchical secret sharing scheme based on smart contract

En Zhang [a,b,*], Ming Li [a,b], Siu-Ming Yiu [c], Jiao Du [d], Jun-Zhe Zhu [a], Gang-Gang Jin [a]

[a] College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China
[b] Engineering Lab of Intelligence Business and Internet of Things of Henan Province, Xinxiang 453007, China
[c] The University of Hong Kong, Hong Kong
[d] College of Mathematics and Information Science, Henan Normal University, Xinxiang 453007, China

ARTICLE INFO

Article history:
Received 10 November 2018
Received in revised form 24 February 2020
Accepted 10 July 2020
Available online 11 August 2020

Keywords:
Secret sharing
Smart contract
Verifiability
Birkhoff interpolation

ABSTRACT

Secret sharing has a number of practical applications in network-based scenarios, such as key transfer protocols, attribute-based encryption and secure multiparty computation. However, existing secret sharing schemes cannot efficiently achieve fairness. They either rely on a trusted third party or require multiple rounds of communication. In this paper, we propose the first decentralized and fair hierarchical threshold secret sharing (HTSS) scheme using blockchain. In the scheme, secret shares are distributed to different levels of parties, and any authorized subset of parties can obtain the secret. We leverage a smart contract to force all participants to commit to the secret shares; otherwise, the committer either reveals his secret share within a certain time frame or pays a fine. Thus, unlike previous HTSS schemes, the participants can reconstruct the secret fairly using Birkhoff interpolation without a trusted party and complete the computation in one round. We formally prove that our scheme is secure. We evaluate the performance of the scheme by implementing our scheme on Ethereum's official test network. Our experiments show that our scheme can run reasonably fast and is practical.

1. Introduction

In the setting of secret sharing schemes, secret shadows, also called secret shares, are distributed among a set of parties, and only authorized parties can reconstruct the secret. Secret sharing has a number of practical applications in linguistic cryptography [1], secure information management [2,3], and image communication [4–6]. The seminal (m, n) secret sharing schemes were first proposed independently by Blakley [7] and Shamir [8] in 1979. Later, many variations of secret sharing schemes were considered [9–21]. A verifiable secret sharing (VSS) scheme was proposed in the work of Chor et al. [9] to achieve security against cheaters. Feldman [10] and Pedersen [11] subsequently proposed non-interactive versions for verifiable secret sharing scheme. To reconstruct multiple secrets at once time, protocols for sharing multiple secrets [12,13] have been explored. Pilaram [15] presented a seminal multi-stage secret sharing scheme which can resist attacks from quantum computers. An outsourcing secret sharing scheme based on homomorphic encryption was introduced by Zhang et al. [14]. Liu et al. [16] proposed a double verification protocol via secret sharing for low-cost RFID tags. Song et al. [17] proposed a new dynamic multiparty quantum direct secret sharing. Goyal and Kumar [18] constructed a non-malleable secret sharing scheme which ensures that the receiver either obtains the original secret or learns an "unrelated" message. Badrinarayanan et al. [19] proposed a new non-malleable secret sharing scheme achieving greater efficiency and a stronger security property.

* Corresponding author at: College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China.
E-mail address: zhangenzdrj@163.com (E. Zhang).

https://doi.org/10.1016/j.ins.2020.07.032
0020-0255/

To resist leak attacks, Fabrice et al. [20] analyzed the local leakage resilience of linear secret sharing schemes and then applied these results to prove the leakage resilience to some secure multiparty computation protocols. Srinivasan et al. [21] designed a compiler that takes a secret sharing scheme for any monotone access structure and proposed a local leakage resilient secret sharing scheme.

1.1. Hierarchical secret sharing

Aiming to handle situations where parties differ in their levels of authority, hierarchical secret sharing schemes have been suggested [22,23]. In these schemes, shares are assigned to participants at different levels in the hierarchy. To reconstruct a secret, participants in certain levels must exist in the subset. For instance, a signature key is distributed to staffs of an organization, and hierarchical secret sharing can make sure that at least one department head must be a participant to reconstruct the key for creating a valid signature. Recently, Farràs et al. [24] introduced ideal HTSS schemes. Later, hierarchical threshold access structures were proposed by Pakniat et al. [25]. Mohamed [26] introduced an HTSS scheme for color images. Traverso et al. [27] proposed an HTSS scheme based on Birkhoff interpolation.

1.2. Fairness in secret sharing

In the setting of secret sharing, one desirable property is fairness, which means that if one participant obtains the secret, the other participant does as well. Harn et al. [28] proposed an approach for secret reconstruction. In 2004, Halpern et al. [29] first introduced rational cryptographic protocols. They pointed out that any approach for reconstructing secrets with a well-recognized upper bound on the running time is unstable, and in the last round, parties will not send anything because they have no incentive to do so. Later, a series of rational protocols were proposed [30,31,12]. A secret sharing protocol based on Bayesian game was introduced by Tian et al. [31]. Zhang et al. [12] proposed a rational multi-secret sharing scheme in point-to-point communication networks. Unfortunately, previous rational secret sharing schemes require multiple rounds which have large overheads. To summarize, existing schemes that can achieve fairness either require a trusted third party or require multiple rounds of communications.

1.3. Bitcoin and fair protocols

Bitcoin is a decentralized digital currency, without relying on trusted third parties, which has recently attracted a lot of attention. By July 2018, bitcoin was the most popular digital cash with a $6300 market capitalization. One benefit of the bitcoin is that no one can take control over the system. Another advantage of bitcoin is anonymity. Users can only be identified by their public keys. It is difficult to find the real person who initiated a transaction [32]. Once a transaction is accepted, it is not reversible. In 2014, Andrychowicz et al. [33,34] introduced the use of bitcoin in multiparty computation to resolve the fairness problem. Bentov et al. [35] proposed a fair protocol based on bitcoin. The underlying technology of bitcoin is blockchain. There are many research papers on blockchain mechanisms [36–39]. Rishab Goyal et al. [40] introduced a cryptographic system that offers a skeleton frame for analyzing and defining the security properties of a proof-of-stake based blockchain. K. Siba et al. [41] presented an evolving blockchain technology that offers technical assistance to a company in digital asset environment by storing each transaction.

1.4. Smart contracts

Smart contracts are supported in some blockchain-based platforms (e.g., Ethereum). By July 2018, Ethereum was the second most popular digital currency, with a $440 market capitalization and has many applications, such as online voting and decentralized governance. Generally, smart contracts are computer programs that act as agreements, and the terms of the agreement can be preprogrammed. Once the contract is deployed, no one can modify the contents, and the agreement will be executed and enforced. Unlike traditional contracts, smart contracts are powerful tools to construct fair protocols. Dong et al. [41] introduced two smart contracts for verifiable cloud computing. Their scheme can resist collusion attacks. If the cloud servers deliver an incorrect result, they will be fined by losing their deposit. A smart contract for a voting protocol was proposed by Mccorry et al. [39]. Wu et al. [42] presented the first decentralized collusion-resistant smart auction contract system, which provides a safe and trusted transaction environment for auction participants. The auction mechanism in the smart contract can effectively prevent bidder collusion and achieve economic robustness. Sergey et al. [43] presented a framework for lightweight verification of Scilla programs that is positioned as an intermediate-level language suitable to serve as a compilation target and as an independent programming framework. Coblenz et al. [44] proposed and justified requirements for a new generation of blockchain software development tools.

1.5. Our contributions

In this work, a novel hierarchical secret sharing scheme based on smart contracts is proposed, as illustrated in Fig. 1. The proposed scheme has several advantages as follows. (1) To achieve fairness, all existing secret sharing schemes either require

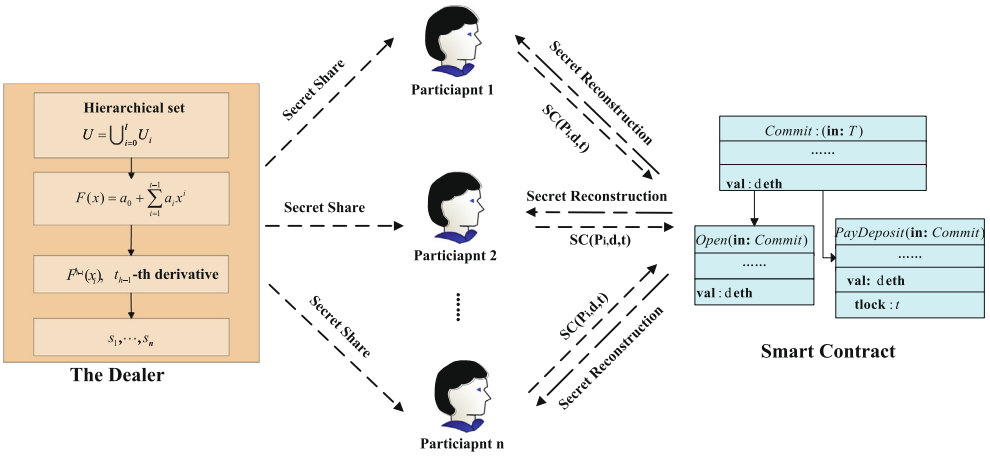

Fig. 1. Hierarchical secret sharing based on Smart Contract.

running multi-rounds or depend on a trusted third party, which is not practical in a real application. In the scheme, we propose the first decentralized and fair HTSS scheme based on a smart contract that does not rely on a trusted third party. Specifically, participants are required to initially pay a deposit, which will be returned to them if they implement the protocol honestly. Otherwise, the deposit will be taken by other participants as compensation. We assume that the deposit paid by each participant is higher than the value of the secret. This allows each participant to have the incentive to implement the protocol honestly. (2) The cost of verification in previous schemes grows dramatically with the number of participants, which is not suitable for weak computational devices. In this work, verification computation can be automatically performed by secret sharing smart contract which can not only validate the correctness of the participants'secret shadows, but also detect the malicious behavior of participants in time. (3) If there is not a trusted third party in the reconstruction phase, participants need to interact to reconstruct secret and the communication complexity is $O(t)$ in previous schemes. However there are many settings in practice where participants usually access their profiles via resource-constrained mobile devices that are not always online, and a non-interactive solution is crucial for such systems to work in practice. In this work, there is no interaction among participants and participants can reconstruct the secret in one round.

The rest of the paper is organized as follows. Section 2 introduces the basic ideas of hierarchical secret sharing and blockchain. We design a smart contract for secret sharing that is suitable for HTSS in Section 3. Section 4 introduces our fair HTSS scheme based on smart contracts. We analyze the scheme and compare our work with previous secret sharing schemes in Section 5. Finally, we conclude our paper in Section 6.

2. Preliminaries

2.1. Hierarchical threshold access structure

Definition 1. Let a set of n parties be denoted by U and let $U = \bigcup_{i=0}^{l} U_i$ consist of levels, where $U_i \bigcap U_j = \phi, 0 \leqslant i < j < l$. Let $\mathbf{m}=\{m_i\}_{i=0}^{l}$ denote the threshold value, where $0 < m_0 < \cdots < m_l$. Then the $(\mathbf{m}, n)$-hierarchical threshold access structure is:

$$\Gamma = \left\{ A \subset U : |A \cap \bigcup_{j=0}^{i} U_j| \geqslant m_i, 0 \leqslant i < l \right\} \tag{1}$$

A scheme that achieves this access structure is a $(\mathbf{m}, n)$-HTSS scheme.

2.2. Birkhoff interpolation

.

Definition 2. (1) Let $X = \{x_1, x_2, \cdots, x_k\}$ denote a series of points in R, where $x_1 < x_2 < \cdots < x_k$;
(2) Let $E = (e_{ij})_{1 \leqslant i \leqslant k, 0 \leqslant j \leqslant l}$ denote a matrix such that $I(E) = \{(i,j) : e_{ij} = 1\}$; and $m = |I(E)|$,
(3) Let $C = \{c_{ij} : (i,j) \in I(E)\}$ be a set of m values.
Then the Birkhoff interpolation is to find a polynomial $P(x) \in R_{m-1}[x]$ satisfying the m equalities

$$P^{(j)}(x_i) = c_{ij} \tag{2}$$

where $R_{m-1}[x]$ is a set of polynomials with degree $m-1$ and $P^{(j)}(\cdot)$ is the (j)-th derivative of $P(x)$. For more details, please refer to [23].

2.3. Blockchain

Blockchain is a distributed ledger that has a number of practical use cases. As the first implementation of blockchain, Bitcoin was first proposed by Satoshi Nakamoto [45] in 2008.

In a real Bitcoin system, each transaction has multiple inputs and outputs.A transaction describes the circulation of an amount v of bitcoin, as illustrated in Fig. 2. Specifically, it is transferred from an address A_{pk} to another address B_{pk}. The transaction can be denoted as $T_x = ((y_1, a_1, \sigma_1), \cdots, (y_l, a_l, \sigma_l), (v_1, \pi_1), \cdots, (v_l, \pi_l), t)$. The triples $(y_1, a_1, \sigma_1), \cdots, (y_l, a_l, \sigma_l)$ are the inputs of T_x, where y_i is a hash value of T_{y_i} that denotes the previous transaction, a_i is an index of the output of T_{y_i} and σ_i is the input, for $i = 1, \cdots, l$. The outputs of a transaction are listed as $(v_1, \pi_1), \cdots, (v_l, \pi_l)$, where π_i is an output. In addition, the transaction has a time lock t, ensuring that the transaction is valid if time t is reached. We denote $((y_1, a_1), \cdots, (y_l, a_l), (v_1, \pi_1), \cdots, (v_l, \pi_l), t)$ as $[T_x]$, which we call the body of T_x.

An Ethereum currency system, which is another important application of blockchain, includes two types of accounts: externally owned accounts and contract accounts. In the system, the currency is called ether, and each user-controlled account is associated with a pair of keys (sk, pk). The address of an account is simply a hash value of the public key pk. The transaction is signed by the sk of the user, and the signature is verified by the pk. Specifically, let a signature on a message m be denoted by $sig_A(m)$ and let $ver_A(m, \sigma)$ denote the verification of a signature. In contrast, a contract account does not have a private key sk. It stores the code of a smart contract that controls the flow of the ethers in the account. Specifically, smart contracts are programmed contracts that automatically move digital currency based on arbitrary preset rules, so that parties transact safely within the agreement of a contract without any trusted third parties. In the real Ethereum system, each transaction has two addresses specifying the receiver and the sender. Parties can initiate a smart contract by creating a transaction in which the receiver is a contract account address. A transaction also contains some gas as the charge of the transaction to encourage miners to contain the code execution in the blockchain.

2.4. Security definition

We now introduce the ideal/hybrid models for the scheme. Let $\prod$ denote the smart contract-based HTSS protocol, *Sim* means that a simulator interacting with ideal model and *A* means that a adversary corrupting participants. In addition, *Sim* gives black-box access to *A*.

Let $Ideal_{Sim}^{\prod}$ be the output of *Sim* after interacting in the ideal model with ideal functionality executing the protocol. Our scheme is run in a hybrid model where participants interact with a trusted party computing secret sharing smart contract. We denote the output of *A* by $Hybrid_{A}^{\prod}$.

Definition 3. A smart contract-based HTSS scheme is secure if for every feasible hybrid model adversary *A*, there exists a feasible ideal model simulator *Sim* such that

$$EXEC_{Hybrid_{A}^{\prod}} \approx EXEC_{Ideal_{Sim}^{\prod}} \tag{3}$$

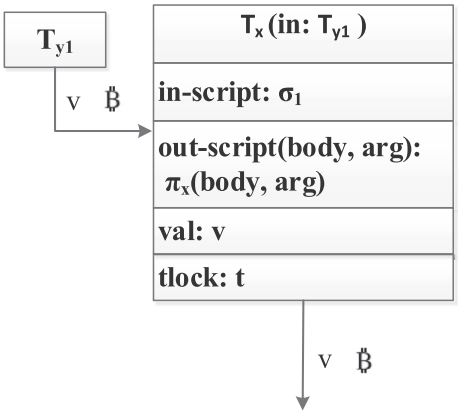

Fig. 2. Transaction T_x.

3. Secret sharing smart contract model

In this section, following the seminal approach [34,39], we propose a secret sharing contract which is suitable for HTSS, as illustrated in Fig.3. A commitment scheme is composed of *commitment phase* and *opening phase* and has two parties: a committer *C* and a receiver *R*. At the *commitment phase*, secret value *s* is committed by C. In the *opening phase*, the commitment is opened by *C* and the secret *s* is revealed. This means that once the commitment phase is over, the value *s* cannot be changed. A commitment scheme for verifiable HTSS can be designed as follows. Let *g* denote a generator of the *qth* order subgroup F_q of F_p^*. To commit a secret share s_i, the committer obtains a pseudoshare s_i' and verifiable information $\alpha_k = g^{a_k}$ from the dealer and sends α_k to the receiver for $k = 0, \cdots, m-1$. Of course, α_k does not leak any information about s_i.

The secret sharing smart contract denoted by $SC(P_i, d, t)$ is suitable for multiple participants; for $i = 1, 2, \cdots, n$, where P_i is a participant, *d* denotes the value of the deposit in Ethereum and *t* denotes the time stamp when the commitment has to be opened by participants. Additionally, $s_i\prime$ is the pseudosecret share, the protocol for which is described below.

Initialization:

(1) For each $i = 1, 2, \cdots, n$, P_i have a pair of keys $(P_i.sk, P_i.pk)$.
(2) P_i knows the pseudosecret share s_i', and all participants obtain the verifiable information $\alpha_k = g^{a_k}$ from the dealer, where $k = 0, \cdots, m-1$.
(3) The ledger contains the transactions $\left\{T_i^j\right\}$ for $1 \leqslant i, j \leqslant n$ and $i \neq j$. The value of the deposit is *d* ether.

The SC.Commit(P_i, d, t):

(1) Every participant commits to the pseudosecret share s_i' as a part of each $Commit(i)$.
(2) The P_i compute the bodies of the transactions $PayDeposit^i$ using appropriate outputs ($Commit(i)$). Then, the participants sign $PayDeposit^i$ and exchange the corresponding signatures.
(3) Each P_i signs the transaction *Commit* and sends the signature to P_j for $1 \leqslant i, j \leqslant n$, and $i \neq j$. P_j signs the transaction *Commit* and broadcasts it.
(4) All participants wait until transaction *Commit* appears on the blockchain.
(5) If *Commit* is not included in the blockchain until time $t - nmax_B$, then P_i immediately quits the protocol, where max_B is the maximum latency time between sending the transaction and being included in the blockchain.

The SC.Open(P_i, d, t):

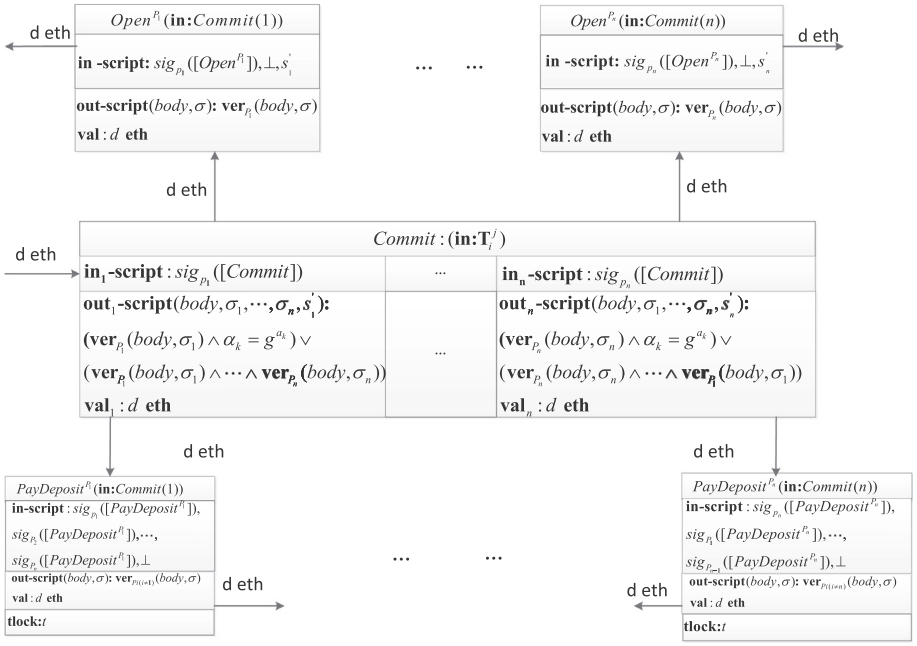

Fig. 3. The Secret Sharing Commitment Model.

(1) All participants broadcast the transactions $Open^i$ ($i \in \{1, \cdots, n\}$), which reveals the pseudosecret shares.
(2) If the transaction $Open^i$ does not appear on the blockchain until time t, then P_j broadcasts $PayDeposit^i$ and gets d ether, for $1 \leqslant i, j \leqslant n$ and $i \neq j$.

4. HTSS based on smart contract

Based on the above secret sharing smart contract, we propose a fair HTSS scheme.
Initialization phase:

(1) Let a set of n parties be denoted by U, and suppose that $U = \bigcup_{h=0}^{l} U_h$ is composed of levels, where m_h is the threshold of level U_h and $h = 0, 1, \cdots, l$. Every participant P_i holds a key pair $(P_i.sk, P_i.pk)$ for $i \in \{1, \cdots, n\}$.
(2) The participants agree on a secret s that they want to jointly reconstruct and on d ether of deposits.
(3) To ensure that the protocol is executed in time, we define a list of timers: m participants must finish registration before the time $t_{finishRegistration}$ and participants must begin the protocol at the time $t_{begainProtocol}$. $t_{finishCommit}$ means that participants who have committed can get back their deposit if not all participants commit to their secret share by $t_{finishCommit}$ and $t_{finishOpen}$ means that participants can claim their deposit if not all participants open their secret shares by $t_{finishOpen}$.

Secret distribution phase:

(1) With the knowledge of $s \oplus r$, the dealer constructs a $(m-1)$th polynomial $F(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{m-1} x^{m-1}$.
(2) The dealer distributes the pseudo secretshare s_i' from $F^{m_{h-1}}(i)$, where $F^{m_{h-1}}(\cdot)$ is the m_{h-1}-th derivative of $F(x)$ and $m_{-1} = 0$.
(3) The dealer distributes the verifiable information $\alpha_k = g^{a_k}$ and random value r to P_i by using private and authenticated channels for $i \in \{1, \cdots, n\}$ and $k = 0, \cdots, m-1$.

Secret recovery phase:

(1) All m participants finish registration before time $t_{finishRegistration}$ and send a deposit of d ether to Ethereum.
(2) The participants execute $SC.Commit$ by $t_{finishCommit}$.
(3) The participants execute $SC.Open$ by $t_{finishOpen}$, which reveals the pseudosecret shares $s_i', i \in \{1, \cdots, n\}$.
(4) If equation $g^{s_i'} = \prod_{k=m_{h-1}}^{m-1} (\alpha_k)^{\frac{k!}{(k-m_{h-1})!} i^{k-m_{h-1}}}$ holds, this means that P_i reveals the correct secret shares. If the equation does not hold, P_j earns d ether from $PayDeposit^i$, for $1 \leqslant i, j \leqslant n$ and $i \neq j$.
(5) At the end of execution, all m participants obtain the pseudosecret shares s_i' and reconstruct the pseudosecret s' by using Birkhoff interpolation for $i \in \{1, \cdots, m\}$. Finally, they decrypt the value s' and obtain the secret $s = s' \oplus r$.

5. Security proof and performance analysis

5.1. Security Proof

Theorem 1. *The fair HTSS protocol based on smart contract is secure against any collusion of $m-1$ malicious participants.*

Proof. Let $\prod$ denote the HTSS protocol based on smart contract. We analyze $\prod$ in a hybrid model where the parties interact with a trusted party computing secret sharing smart contract. We construct a simulator Sim for a nonuniform PPT such that $EXEC^{\prod}_{Hybrid_A} \approx EXEC^{\prod}_{Ideal_{Sim}}$, where A is an adversary allowed to corrupt any $m-1$ participants maliciously and the simulator Sim gives black box access to A. We first construct a simulator and then argue that the simulation is indistinguishable between the ideal and hybrid models.

(1) Simulating honest participants P_i : Sim simulates the view of an eavesdropping adversary A in the hybrid model. Since we use the private and authenticated channels, Sim can emulate all honest clients by using dummy inputs. It is easy to verify that the ideal and hybrid model can not be distinguished. We omit the analysis.
(2) Simulating a corrupted subset of $I \subset \{P_1, \cdots, P_n\}$ with any less than m participants and honest participants $\{P_1, \cdots, P_n\} \setminus I$: We construct a simulator Sim that simulates an adversary A who corrupts participants in I. Simlearns the inputs of the participants in I through an internally emulated copy of the hybrid model; and the description of Sim is as follows.

a. If P_i is corrupt, *Sim* invokes A on the inputs $\{s'_i\}_{i\in I}$, and $\alpha_k = g^{a_k}$ for $k = 0, \cdots, m-1$.

b. *Sim* executes the secret sharing smart contract using the inputs and computes the body of the transactions *Commit*, *Open*, and *PayDeposit*. *Sim* sends the transaction *Commit* to the adversary A. If the transaction *Commit* does not appear on the blockchain until time $t - nmax_B$, the protocol will quit.

c. *Sim* executes *SC.Open* before time $t_{finishOpen}$, which reveals the pseudosecret shares s'_i. If the equation $g^{s'_i} = \prod_{k=m_{h-1}}^{m-1} (\alpha_k)^{\frac{k!}{(k-m_{h-1})!} i^{k-m_{h-1}}}$ does not hold, P_j earns d ether from the $PayDeposit^i$ transactions, for $j \in \{1, \cdots, n\}$ and $i \neq j$. If the equation holds, the adversary A sends the same share that it received.

The view of each corrupted party is drawn from a distribution that is identical to the corresponding distribution in the hybrid model. In the Ethereum currency system, if the adversary wants to obtain the money corresponding to these transactions, he has to construct a new blockchain that violates irreversibility. At the end of execution, since the Birkhoff interpolation polynomial can be uniquely determined by any m points, the $m-1$ shares cannot provide any crucial information about the secret. It is easy to verify that

$$EXEC^{\prod}_{Hybrid,A} \approx EXEC^{\prod}_{Ideal,Sim} \tag{4}$$

Theorem 2. *If the blockchain is irreversible, the HTSS protocol based on smart contracts can satisfy fairness for all participants.*

Proof. (1) If the participants are honest, they send the correct pseudoshare s'_i to the blockchain and then get their deposit back. Therefore, all the participants learn the pseudoshares s'_i and reconstruct the pseudosecret s' by using Birkhoff interpolation. Finally, they decrypt the value s' and obtain the secret $s = s' \oplus r$.

(2) If participant P_i is dishonest, he may send $\tilde{s'_i}$ to the blockchain. The other participants can verify the correctness of the revealed share $\tilde{s'_i}$

$$\begin{aligned} g^{\tilde{s'_i}} &= g^{f^{m_{h-1}}(i)} \\ &= g^{\left(a_0 + a_1 i + a_2 i^2 + \cdots + a_{m-1} i^{m-1}\right)^{(m_{h-1})}} \\ &= (\alpha_0)^{(m_{h-1})} \left(\alpha_1^i\right)^{(m_{h-1})} \cdots \left(\alpha_{m-1}^{i^{m-1}}\right)^{(m_{h-1})} \\ &= \prod_{k=m_{h-1}}^{m-1} (\alpha_k)^{\frac{k!}{(k-m_{h-1})!} i^{k-m_{h-1}}} \end{aligned} \tag{5}$$

where $i \in \{1, \cdots, n\}$. If Eq. (5) does not hold, the other participants broadcast the transaction *PayDeposit* and redeem d ether. In the end, each participant has an incentive to implement the protocol honestly. Therefore, the proposed protocol can satisfy fairness for each party.

5.2. Performance analysis

We implement a prototype of our scheme on a laptop with an Intel Core i7-6700 CPU (4 core 3.60 GHz) and 4 GB RAM running 64-bit Windows 7. The implementation of the scheme is deployed on Ethereum's official test network. We use the Solidity programming language, which is an object-oriented, high-level language for implementing smart contracts.

In the scheme, each transaction of a participant is broadcast once. A full overview of the computational cost of each transaction is shown in Table 1, where n = 12. The computational cost of the commit stage and open stage changes with an increasing number of participants, while the computational cost of initialization remains unchanged. The data in Fig. 4 shows that the gas of commit and open grows linearly with an increasing number of participants, and Fig. 5 shows the total gas consumption of the scheme, which includes the initialization phase, commitment phase and opening phase. We also provide a time analysis of the scheme. Table 2 shows the time of transactions. Additionally, we measured the running time of the verification time and the reconstruction time. The details of the scheme performance are shown in Table 3, where n =12. Fig. 6 shows the time of verification, which directly influences the performance of the scheme. The verification time data

Table 1
Computational Cost of Transaction (gas).

Clients	$m = 5$	$m = 6$	$m = 7$	$m = 8$	$m = 9$	$m = 10$
Initialization phase	504510	504512	504513	504515	504516	504514
Commit phase	152091	179522	206953	234384	261815	289246
Open phase	111361	133646	155931	178216	200501	222786

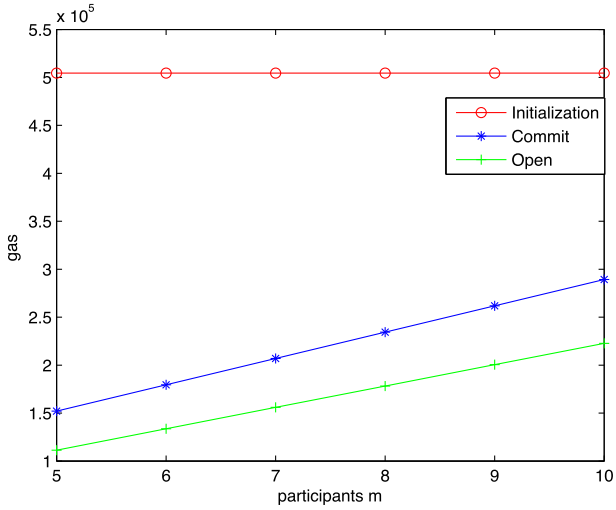

Fig. 4. Computational Cost of Transaction.

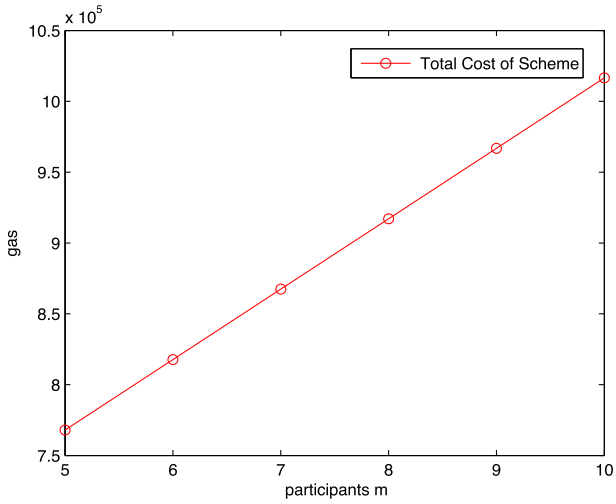

Fig. 5. Total Cost of Scheme.

range from 10.05 ms to 849.81 ms, and the number of clients varies from 6 to 10. The curve in Fig. 7 shows the reconstruction time of the scheme. The distribution time data varies from 4.17 ms to 5.36 ms as the number of participants increases.

We compare the scheme with other works in Table 4. Tassa [23] proposed a method that divides the parties into different levels and each level of the parties have different shares. As long as the subset of parties were authorized, the secret can be reconstructed, while the unauthorized subset can not learn any useful information about the secret. Traverso et al. [27] presented the verifiable and dynamic HTSS scheme that is efficient and allowed to provide shares of equal size for all shareholder in the hierarchy. In their work, each participant can verify his share of the scheme, and the scheme can protect the participant against malicious dealers and parties. However, the above schemes cannot ensure fairness. Pilaram [15] proposed a seminal lattice-based multistage secret sharing scheme that can resist against quantum algorithms. In this work, the combiner cannot misuse the pseudo-secret shares to obtain the original shares and disclose the unrecovered secrets. Parties can reconstruct a secret subset at each stage, while other secret subsets are not revealed. Moreover, the scheme inherits its efficiency from simple matrix operations. However, the scheme requires a trusted combiner to recover the secret. Tian et al. [31] suggested a rational secret sharing scheme using Bayesian mechanism. In the game, all participants cooperate with each other and obtain the maximum expected utility. The scheme also prevents clients from cheating and achieves fair computation. Unfortunately, the expected number of iterations is multiple rounds. Zhang et al. [46] introduced a verifiable rational secret sharing scheme in mobile networks that is more suitable for devices with limited size and processing power. However, the previous rational schemes require multiple rounds.

Table 2
Time of Transaction (ms).

Clients	$m = 5$	$m = 6$	$m = 7$	$m = 8$	$m = 9$	$m = 10$
Commit phase	70	73	72	73	75	70
Open phase	79	84	82	81	89	84

Table 3
Time of Secret Sharing (ms).

Clients	$m = 6$	$m = 7$	$m = 8$	$m = 9$	$m = 10$
Verification time	10.05	15.13	17.81	34.47	849.81
Reconstruction time	4.17	4.47	4.80	5.10	5.36

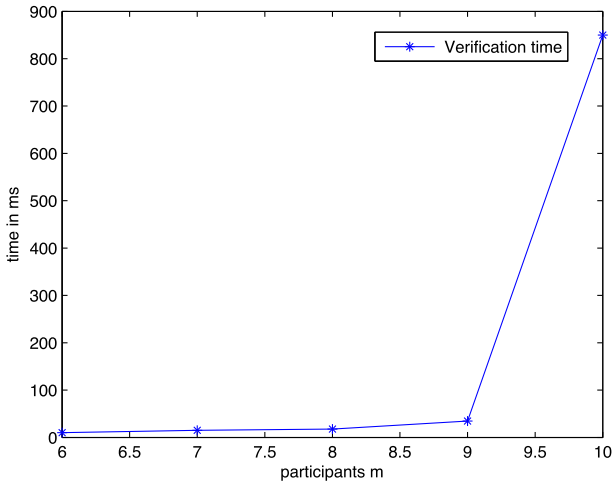

Fig. 6. Verification time.

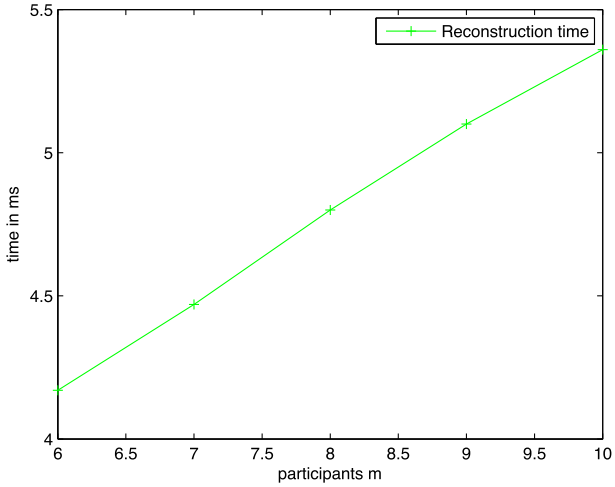

Fig. 7. Reconstruction time.

In the proposed HTSS scheme, secret shares are distributed to different levels of participants, and the secret can be obtained by implementing a smart contract. Moreover, the scheme can resist malicious participants and achieve fairness for participants without relying on a trusted third party. Specifically, every participant is required to initially pay a deposit.

Table 4
Feature comparison of schemes.

Feature	Number of iterations	Verifiability	Fairness	Trusted party
Tassa [23]	one round	No	No	No
Traverso [27]	one round	Yes	No	No
Pilaram [15]	one round	Yes	Yes	Yes
Tian [31]	multi-rounds	Yes	Yes	No
Zhang [46]	multi-rounds	Yes	Yes	No
Our scheme	one round	Yes	Yes	No

If the participants do not follow the protocol honestly, they suffer financial penalties; otherwise, the money is paid back to the honest participants. Therefore, each participant has motivation to implement the protocol honestly.

6. Conclusion

We presented a fair HTSS scheme based on smart contract. In the scheme, secret shares are distributed to different levels of participants, and any authorized subset of parties can reconstruct the secret. Moreover, all parties need to commit to secret shares, and the committer either sends his secret share honestly or pays a fine. This is written as a secret sharing smart contract that is suitable for HTSS. The smart contract can not only validate the correctness of participants' secret shares but also detect the malicious behavior of participants in time. If the participants do not follow the protocol honestly, malicious behavior can be detected, and they will suffer financial penalties. In the end, parties can reconstruct the secret fairly without relying on a trusted party.

CRediT authorship contribution statement

En Zhang: Conceptualization, Methodology, Investigation, Writing - original draft. **Ming Li:** Validation, Formal analysis, Software. **Siu Ming Yiu:** Validation, Formal analysis, Writing - review & editing. **Jiao Du:** Resources, Writing - review & editing, Data curation. **Jun-Zhe Zhu:** Investigation, Software. **Gang-Gang Jin:** Investigation, Software.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by National Natural Science Foundation of China (U1604156, 61602158, 61772176), Science and Technology Research Project of Henan Province (172102210045) and Soft Science Research Project of Henan Province (202400410088).

References

[1] U. Ogiela, M. Takizawa, L. Ogiela, Visual captcha application in linguistic cryptography, Concurrency Comput.: Practice Experience 30 (2) (2018), e4362.
[2] L. Ogiela, M.R. Ogiela, Insider threats and cryptographic techniques in secure information management, IEEE Syst. J. 11 (2) (2015) 405–414.
[3] L. Ogiela, M.R. Ogiela, Security of visual codes in service management in the cloud, in: 2017 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), IEEE, 2017, pp. 165–168.
[4] Y.-X. Liu, C.-N. Yang, C.-M. Wu, Q.-D. Sun, W. Bi, Threshold changeable secret image sharing scheme based on interpolation polynomial, Multimedia Tools Appl. (2019) 1–15.
[5] Y.-X. Liu, C.-N. Yang, S.-Y. Wu, Y.-S. Chou, Progressive (k, n) secret image sharing schemes based on boolean operations and covering codes, Signal Processing: Image Commun. 66 (2018) 77–86.
[6] Y.-X. Liu, C.-N. Yang, Y.-S. Chou, S.-Y. Wu, Q.-D. Sun, Progressive (k, n) secret image sharing scheme with meaningful shadow images by gemd and rgemd, J. Vis. Commun. Image Represent. 55 (2018) 766–777.
[7] G.R. Blakley, Safeguarding cryptographic keys, in: Proceedings of the national computer conference, Vol. 48, 1979, pp. 313–317..
[8] Adi Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.
[9] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch, Verifiable secret sharing and achieving simultaneity in the presence of faults, in: Foundations of Computer Science 1985, 26th Annual Symposium on, IEEE, 1985, pp. 383–395..
[10] P. Feldman, A practical scheme for non-interactive verifiable secret sharing, in: Foundations of Computer Science 1987, 28th Annual Symposium on, IEEE, 1987, pp. 427–438..
[11] T. P. Pedersen, Distributed provers with applications to undeniable signatures, in: Advances in Cryptology-EUROCRYPT 1991, Springer, 1991, pp. 221–242..
[12] E. Zhang, Y. Cai, Rational multi-secret sharing scheme in standard point-to-point communication networks, Int. J. Found. Computer Sci. 24 (06) (2013) 879–897.
[13] Y. Liu, F. Zhang, J. Zhang, Attacks to some verifiable multi-secret sharing schemes and two improved schemes, Inf. Sci. 329 (2016) 524–539.
[14] E. Zhang, J. Peng, M. Li, Outsourcing secret sharing scheme based on homomorphism encryption, IET Inf. Secur. 12 (1) (2018) 94–99.
[15] H. Pilaram, T. Eghlidos, An efficient lattice based multi-stage secret sharing scheme, IEEE Trans. Dependable Secure Comput. 14 (1) (2017) 2–8.

[16] Y. Liu, M. Ezerman, H. Wang, Double verification protocol via secret sharing for low-cost rfid tags, Future Generation Computer Syst. 90 (2019) 118–128.
[17] Y. Song, Z. Li, Y. Li, A dynamic multiparty quantum direct secret sharing based on generalized ghz states, Quantum Inf. Process. 17 (9) (2018) 244.
[18] V. Goyal, A. Kumar, Non-malleable secret sharing, in, in: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, ACM, 2018, pp. 685–698.
[19] S. Badrinarayanan, A. Srinivasan, Revisiting non-malleable secret sharing, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2019, pp. 593–622.
[20] F. Benhamouda, A. Degwekar, Y. Ishai, T. Rabin, On the local leakage resilience of linear secret sharing schemes, in: Annual International Cryptology Conference, Springer, 2018, pp. 531–561..
[21] A. Srinivasan, P. N. Vasudevan, Leakage resilient secret sharing and applications, in: Annual International Cryptology Conference, Springer, 2019, pp. 480–509..
[22] S. C. Kothari, Generalized linear threshold scheme, in: CRYPTO 84 on Advances in Cryptology, Vol. 84, Springer, 1984, pp. 231–241..
[23] T. Tassa, Hierarchical threshold secret sharing, J. Cryptology 20 (2) (2007) 237–264.
[24] O. Farras, C. Padro, Ideal hierarchical secret sharing schemes, IEEE Trans. Inf. Theory 58 (5) (2012) 3273–3286.
[25] N. Pakniat, M. Noroozi, Z. Eslami, Distributed key generation protocol with hierarchical threshold access structure, IET Inf. Secur. 9 (4) (2015) 248–255.
[26] P. Mohamed Fathimal et al, Hierarchical threshold secret sharing scheme for color images, Multimedia Tools Appl. 76 (4) (2017) 5489–5503.
[27] G. Traverso, D. Demirel, J. Buchmann, Dynamic and verifiable hierarchical secret sharing, in: Information Theoretic Security: 9th International Conference, ICITS 2016, Tacoma, WA, USA, August 9–12, 2016, Revised Selected Papers 9, Springer, 2016, pp. 24–43..
[28] L. Harn, C. Lin, Y. Li, Fair secret reconstruction in (t, n) secret sharing, J. Inform. Security Appl. 23 (C) (2015) 1–7.
[29] J. Halpern, V. Teague, Rational secret sharing and multiparty computation, in: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, ACM, 2004, pp. 623–632..
[30] E. Zhang, F. Li, B. Niu, Y. Wang, Server-aided private set intersection based on reputation, Inf. Sci. 387 (2017) 180–194.
[31] Y. Tian, C. Peng, D. Lin, J. Ma, Q. Jiang, W. Ji, Bayesian mechanism for rational secret sharing scheme, Sci. China Inform. Sci. 58 (5) (2015) 1–13.
[32] I. Miers, C. Garman, M. Green, A.D. Rubin, Zerocoin: Anonymous distributed e-cash from bitcoin, IEEE Symposium Security Privacy (2013) 397–411.
[33] M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, Secure multiparty computations on bitcoin, in: 2014 IEEE Symposium on Security and Privacy, IEEE, 2014, pp. 443–458..
[34] M. Andrychowicz, S. Dziembowski, D. Malinowski, Ł. Mazurek, Fair two-party computations via bitcoin deposits, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 105–121..
[35] I. Bentov, R. Kumaresan, How to use bitcoin to design fair protocols, in: International Cryptology Conference, Springer, 2014, pp. 421–439..
[36] R. Goyal, V. Goyal, Overcoming cryptographic impossibility results using blockchains, in: Theory of Cryptography Conference, Springer, 2017, pp. 529–561..
[37] R. Pass, L. Seeman, A. Shelat, Analysis of the blockchain protocol in asynchronous networks, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2017, pp. 643–673.
[38] T. K. Siba, A. Prakash, Block-chain: An evolving technology., Global Journal of Enterprise Information System 8 (4)..
[39] P. McCorry, S. F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy, in: International Conference on Financial Cryptography and Data Security, Springer, 2017, pp. 357–375..
[40] I. Eyal, A. E. Gencer, E. G. Sirer, R. Van Renesse, Bitcoin-ng: A scalable blockchain protocol., in: NSDI, 2016, pp. 45–59..
[41] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, A. van Moorsel, Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing, in, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017, pp. 211–227.
[42] S. Wu, Y. Chen, Q. Wang, M. Li, C. Wang, X. Luo, Cream: a smart contract enabled collusion-resistant e-auction, IEEE Trans. Inf. Forensics Secur. 14 (7) (2018) 1687–1701.
[43] I. Sergey, V. Nagaraj, J. Johannsen, A. Kumar, A. Trunov, K. C. G. Hao, Safer smart contract programming with scilla, Proceedings of the ACM on Programming Languages 3 (OOPSLA) (2019) 1–30..
[44] M. Coblenz, J. Sunshine, J. Aldrich, B.A. Myers, Smarter smart contract development tools, in, in: Proceedings of 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), 2019.
[45] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008)..
[46] E. Zhang, P. Yuan, J. Du, Verifiable rational secret sharing scheme in mobile networks, Mobile Inform. Syst. (2015).