IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 29, 2019, accepted August 20, 2019, date of publication August 27, 2019, date of current version September 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937917

A Key Protection Scheme Based on Secret Sharing for Blockchain-Based Construction Supply Chain System

FENG XIONG[1], RUIYANG XIAO[2], WEI REN[2], (Member, IEEE), RONGYUE ZHENG[3], AND JIANLIN JIANG[3]

[1]School of Business Administration, Zhongnan University of Economics and Law, Wuhan 430073, China
[2]School of Computer Science, China University of Geosciences (Wuhan), Wuhan 430074, China
[3]Faculty of Architecture, Civil Engineering and Environment, Ningbo University, Ningbo 315211, China

Corresponding author: Wei Ren (weirencs@cug.edu.cn)

This work was financially supported by the National Key Research and Development Program of China under Grant 2016YFC0702107.

ABSTRACT Traditional construction supply chains suffer from extra delays, costs and information wastages due to information intermediaries. Blockchain, a decentralized infrastructure, can provide irreversibility, undeniableness, uniqueness and anonymity for trades. Hence, we first propose a blockchain-based construction supply chain framework to reduce limitations in traditional ones. However, payment security by blockchain must be guaranteed and token assets in accounts must be protected. Although the loss of private keys will not result in the exposure or the modification of records in blockchain due to merkle root and blockheader hash, fake payments can be generated and all tokens in the accounts controlled by the private keys may be stolen by attackers. Existing approaches towards private-key protections include biometric-basic signature schemes, index-hidden private key designs and post-quantum blockchain schemes. Nevertheless, none of them can recover lost private keys. Therefore, we design a private-key distribution protocol in blockchains to preserve security of private keys with key recovery. Specifically, our scheme not only uses secret sharing to improve possibilities of recovering lost keys but also introduces network protocols to guarantee security of secret share transmission. The proposed scheme is then proven secure and feasible both in theoretical and experimental analysis.

INDEX TERMS Construction supply chain, private key distribution, blockchain, secret sharing.

I. INTRODUCTION

Nowadays construction supply chain (CSC) finance grows in importance along with economic globalization, which rapidly increase circulation of capital and expansion of market. It raises a claim on more interdependent relationships (e.g., managements in raw material purchases, logistic transportations, tender managements and so on) among enterprises, suppliers and dealers [1]. Nevertheless, CSC has been suffered from ineffective information transmissions. In traditional CSC modes, information intermediaries takes the role of transferring information, and every two enterprises solely communicate with each other through an information intermediary [2]. Existence of information intermediaries not only ensures fairness, reliability and transparency in trades, but also leads to long time span, expensive management fees and possible information wastages because of scattered geographical distribution and large amounts of participants [3], [14]. Lack of a coordinative information platform makes CSC eager for a peer-to-peer structure (e.g., a blockchain-based system) to replace information intermediaries.

Blockchain was first introduced by Satoshi Nakamoto to propose a virtual crypto-currency Bitcoin (BTC) in 2008 [4], which introduces a decentralized infrastructure to provide irreversibility, undeniableness, uniqueness and anonymity for transactions. Each user in blockchains interacts with each other directly and generates information to be recorded in blockchains. Moreover, the ownerships of records are solely proven by correct private keys, which provides authentications through signatures. Records in blockchains are monitored by all nodes, ensuring non-repudiation and integrity of information. Blockchain's coming out has brought in

The associate editor coordinating the review of this article and approving it for publication was Sabu M. Thampi.

plentiful other extended applications (e.g., Ethereum (ETH), Ripple (XRP), Bitcoin Cash (BCH) and so on), setting off billions of investments in blockchain applications. Nevertheless, growing value in blockchain industry has brought in increasing security risks. Most owners' addresses, public keys and private keys (PriKeys) are stored in terminals locally, while few are stored in online wallet servers. Therefore, security of ownership is tightly related with PriKeys [8]. For instance, once a PriKey was breached by single point of failure, its ownership can be tampered illegally [5].

Major security threats about PriKeys include being lost by terminals, being breached by quantum computing attacks and being stolen of by online PriKey generators [6]. Some approaches are to prevent losing PriKeys by hiding them into specific indexes (such as hiding them into plain English text [7], fractal trees [8] and so on). Some approaches are to defend against quantum computing attacks by redesigning post-quantum blockchain schemes, which involve lattice-based signature schemes [10], double-signature schemes [9] and anti-quantum transaction authentication schemes [11]. However, methods above merely cope with security threats from external attacks. There exist no approaches towards how to prevent losing PriKeys caused by terminals' physical damages (internal attacks) or how to recover lost PriKeys.

Therefore, to mitigate limitations on low information transmission efficiency in CSC modes and unrecovery of PriKeys in blockchains, in this paper we propose a secret-sharing-based private-key protection protocol, which is designed under CSC modes in blockchains. Specifically, the contributions of this paper are as follows:

1) To avoid centralized-service delays, transmission costs and information wastages in transmission processes, we transplant the basic structure of consortium blockchains and then propose a framework for CSC based on blockchains, where transaction details are recorded in blockchains instead of information intermediaries, ensuring more fairness, reliability and correctness of trades.
2) To reduce common attacking models and help recover lost PriKeys, we introduce a general private-key distribution protocol composed of ten sub-protocols, where each participant memorizes as little information as possible. Moreover, this protocol can be further extended in blockchain-based applications (e.g., our CSC framework).

Having introduced the paper, we will briefly review related work in the next section (see Section II). In Section III, we describe the problem formulation and other relevant materials. In Section IV, we present our proposed approach, prior to presenting the evaluation findings in Section V. We conclude the paper in Section VI.

II. RELATED WORK

A. BLOCKCHAIN

Satoshi Nakamoto primarily introduced blockchain to propose the world's first crypto-currency ''bitcoin'' in 2008 [4]. Compared with centralized infrastructures, blockchain establishs a consensus mechanism among decentralized s over the Internet. Validators pack a list of transactions into a block. Moreover, blockheader includes a merkle root hash to maintain integrity of transactions, a timestamp to claim each block's generating order, the hash of previous block to confirm chain's architecture and so on. Consensus algorithms (e.g., Proof-of-Work, Proof-of-Stake and Delegated Proof of Stake) select a validator at set intervals to link his block to the previous one. These algorithms not only eliminate double-spending transactions but also ensure transactions visible and reliable. All participants share the same copy of data sets (e.g., account book of crypto-currency) in blockchains, assuring undeniableness and irreversibility of transactions.

B. BLOCKCHAIN ENABLED SUPPLY CHAIN

Yafei Ji *et al.* concluded two attack scenarios towards cyber supply chains: attack via manufacturer source code or product and attack via vendor remote access [18]. Han Jeong Hugn *et al.* investigated possible supply chain devevlopments influenced by blockchains, where they concluded trust as the most influential driving factor. Besides, blockchain technologies are highly promising tools on supply chains from following aspects: extended visibility and traceability, supply chain digitalisation and disintermediation, improved data security and smart contracts [13]. Therefore, supply chain managements gain strong potential in blockchains. For example, Yu Cui and Hiroki Idota proposed a decentralized information sharing blockchain system to improve supply chain resilience [14]. Si Chen proposed a blockchain-based supply chain management framework to solve limitations in quality inspections [15]. However, Youness Tribis statistically surveyed current supply chain managements based on blockchains, which illustrates that major researches focus on pharma and food. Except for general discussions without a particular domain, there exist no researches on construction supply chains [12]. Hence, we will first introduce a blockchain-based design particularly in construction supply chain.

C. TECHNOLOGICAL CHALLENGES

Even if blockchain technologies possess preponderances in overcoming shortcomings in supply chains (e.g., extra time cost, expensive management fees and possible information wastages [3]), they bring in extra weaknesses in security. Pascal Urien analyzed that main challenges among trusted transactions in blockchains include PriKey security and malwares. And he designed a crypto terminal which employs firmware programing and security policies to represent current transaction generators [17]. Beini Zhou discussed security risks brought by online PriKey generators and online crypto-currency wallets [6]. Furthermore, there still exist threats from quantum computers, whose computation abilities can destroy trust policies of consensus algorithms.

Existing approaches towards PriKey protections include: biometric-based signature schemes [19], index-hidden

PriKey designs [7], [8] and post-quantum blockchain schemes [9]–[11].

1) **Biometric-based Signature Schemes**
Yosuke Kaga *et al.* firstly proposed a secure and practical signature scheme which uses participant's biometric information (e.g., fingerprint, face, finger vein) as a PriKey. By combining blockchains and biometrics together, privey keys are secured without ownerships' being tampered illegally. And even lost PriKeys can be recovered through biometrics at any time [19]. But such an index-hidden scheme breaks anonymity in blockchain because relationships between biometrics and accouts are one-to-one, which does not suits for unreal-name scenarios. Moreover, despite high efficiency of recovering PriKeys, costs on collecting and analyzing biometric information are too heavy to be afforded for most users.
2) **Index-hidden Private Key Designs**
James Stanley proposed a stegoseed method by hiding PriKeys into plain English text [7]. Osama Hosam proposed a robust steganography technique, which hides PriKeys into fractal trees and discretizes angles and lengths of tree branches [8]. Nevertheless, nor plain text or fractal trees can prevent risks of losing by itself. For example, if a user is likely to lost his steganographic image remissly beacuse of laptop crashes, it will be scarcely possible for him to recover PriKeys again.
3) **Post-quantum Blockchain Schemes**
Yulong Gao *et al.* designed a post-quantum blockchain (PQB) composed of a lattice-based signature scheme and a double-signature scheme [9]. Wei Yin *et al.* proposed a anti-quantum transaction authentication scheme to extend multiple signatures from single lattice space to multiple ones [10]. Chaoyang Li and Xiubo Chen *et al.* proposed a new lattice-based signature scheme where PriKeys are generated by Bonsai Trees technology [11]. However, these post-quantum blockchain schemes cope with quantum computing threats which require new blockchain architectures in telecommunications and digital signatures, decreasing compatibility and extensibility of current blockchains.

As concluded above, current approaches can not support recoverability of PriKeys or cope with security threats caused by terminals' physical damages. Hence, existing approaches focus on secret sharing. For instance, Jan Camenisch *et al.* proposed the first t-out-of-n threshold password-authenticated secret sharing protocol, which is proven secure in the universal composability framework and offers important advantages over property-based definitions [20]. Moreover, Stanislaw Jarecki *et al.* proposed a Password-Protected Secret Sharing (PPSS) scheme which does not require secure channels or PKI other than in the initialization stage [21]. Considering that schemes above are mainly cope with online bitcoin wallet trust problems, we will extend existing scenario into any bitcoin wallet and then design a secret-sharing-based key protection scheme in blockchain to improve security of PriKeys.

III. PROBLEM FORMULATION

A. DESIGN PRINCIPLES

We outlined main disadvantages of existing private-key security schemes in Section I and Subsection II-C, which concluded that current approaches can not support recovering lost PriKeys. To enhance both the recoverability of PriKeys and the extensibility of blockchians, we provide three specific design goals listed below:

1) A basic CSC-blockchain mode should be compatible for extensional CSC applications.
2) Local devices should save as less information as possible to prevent single point of failure.
3) Schemes designed for PriKey protection should be compatible and extensible.

Goal 1 and Goal 3 put forward requests for basic schemes (designs) which are applicable for extended designs. That is, such a scheme can be further discussed without extensive infrastructures. Goal 2 is proposed to facilitate users' private-key saving challenges. Users can memorize a shorter or meaningful password instead of a meaningless 256-bit PriKey, which leaves extra back-ups in their brain. Moreover, Goal 2 allows users to distribute secret shares in several devices, where even if some shares are stolen or missing, users can recover initial secret from the remaining informations.

B. ADVERSARY MODELS

Noted that considerable devices (e.g., note papers, smart phones, laptops, desktops, brains) suit for storing passwords, we give reasonable assumptions as following:

1) Attackers cannot manage more than 50 percent computing velocities at the same time, making it impossible for more than 50 percent of devices losing their shares meanwhile.
2) Users prefers to generate an as private and safe as possible password in brain and will not leak them out on their initiatives.

The design of adversary models depends on attacks. Existing attacking approaches include: vulnerable links, passive attacks, active attacks, cryptography attacks, special network transport protocol attacks, special secret sharing protocol attacks and so on.

In this way, twelve adversary models can be listed, where the former five models attack on the (t, n) secret sharing algorithm and the latter seven models attack on network protocols: (1) Compromise Attacks, (2) Collusion Attacks, (3) History Analysis Attacks, (4) Malicious Behavior Attacks, (5) Compromise of Entities with *NodeA*, (6) Eavesdropping Attacks, (7) Reflect Attacks, (8) Interleave Attacks, (9) Replay Attacks, (10) Combination Attacks, (11) DDoS Attacks and (12) Sybil Attacks.

Furthermore, the specific descriptions over above adversary models are given in Subsection V-A for better understanding in security analysis.

Taking both principles and adversary models into consideration, we explore a blockchain-based CSC framework and a private-key distribution protocol to defend against above attacking models.

IV. PROPOSED SCHEMES

Before introducing our private-key distribution protocols, we first propose a basic blockchain-based CSC framework to avoid unfairness, unreliability and non-transparency in trades. On the basis of a basic CSC framework, we then introduce a private-key distribution protocol composed of ten sub-protocols to hence recoverability of PriKeys. To simplify descriptions of algorithm in sub-protocols, we introduce ten notations listed in Fig. 3.

A. A BLOCKCHAIN-BASED CONSTRUCTION SUPPLY CHAIN (CSC) FRAMEWORK

As is shown in Fig 1, suppliers, enterprises and dealers compose three basic entities in traditional construction modes. Enterprises purchase materials from suppliers and market goods through dealers. The whole self-liquidating trade finance process includes pledges of receivable accounts, third party supervisions, project bids and so on, requiring diverse records to be monitored. Thus we introduce blockchains to improve information transmission performances.

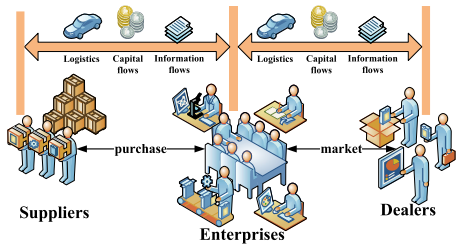


FIGURE 1. The basic traditional CSC mode.

We outline designs of blocks as below.

1) **Block Designs** Any block is composed of a blockheader and a block dataset in Table 1. The block dataset includes detail records and its signatures, while the blockheader includes the block size, the block type, the block version, the merkle root hash of block dataset, the hash of last block and the time-stamp.
2) **Entities** Considering that enterprises are involved in all information flows, capital flows and logistics processes, we increase functions of enterprises as validators in blockchains. Moreover, other entities (e.g., suppliers, dealers and so on) shall act as record-senders and regulators.
   The identity of a entity is a PriKey generated by itself, which is kept as a secret. And the hash of a PriKey is public to all as an identity address. Hence, each entity can sign to any messages through its PriKey while any other identities can verify the validity of a signature through its related identity address.
3) **Consensus Mechanisms** Any entity sends their records to the enterprise with their signatures. After receiving records, the enterprise verifies the validity of them and digs them into blocks. Blocks will be put into blockchains by enterprises. Finally, each entity store a same copy of all records in blockchains locally, which prevents records from tamper-proofing.

TABLE 1. The design of a block.

Block		
Blockheader	the block size	the size of block dataset
	the block type	the type of block which includes information flow, capitalflow and logistics
	the block version	the updata version of block
	the merkle root hash of block dataset	the result of merkle root block dataset through any hash function
	the hash of last block	the result of last block in blockchains
	the timestamp	the time record of putting the block into blockchains
Block Dataset	the detail records	the records which enterprises receive from any entities
	the signatures	the signatures of all records

Some disadvantages of traditional CSC mode can be solved. For instance, both information wastages caused by scattered geographical distributions and centralized-service delays caused by repeat responses can be eliminated because every entity keeps a same copy of records locally and can refer to any information at any moment. And transmission costs can be avoided because there exist no specific information intermediaries. Furthermore, our framework prevents records from tampering by monitoring records within full participations, thus maintaining the fairness, reliability and safety of trades.

B. PRIVATE-KEY DISTRIBUTION PROTOCOLS

As is mentioned in Fig. 2, the former five sub-protocols are proposed to generate and send secret shares when the latter five sub-protocols are proposed to recover secret shares and regenerate secrets.

Before describing private-key distribution protocols, we first introduce several notations to simplify symbols. Table 2 lists notations used in these protocols.

1) SUB-PROTOCOL 1: THE SECRET ENCRYPTION (SECRETENC) ALGORITHM

In this sub-protocol, *NodeA* encrypts its password PWD_A through its private key *PriKeyA* and outputs encrypted ciphertext C.

$$C \Leftarrow ENC(PWD_A, PriKey_A), \quad where\ PWD_A \in \{0,1\}^m, \\ PriKey_A, C \in \{0,1\}^L. \quad (1)$$

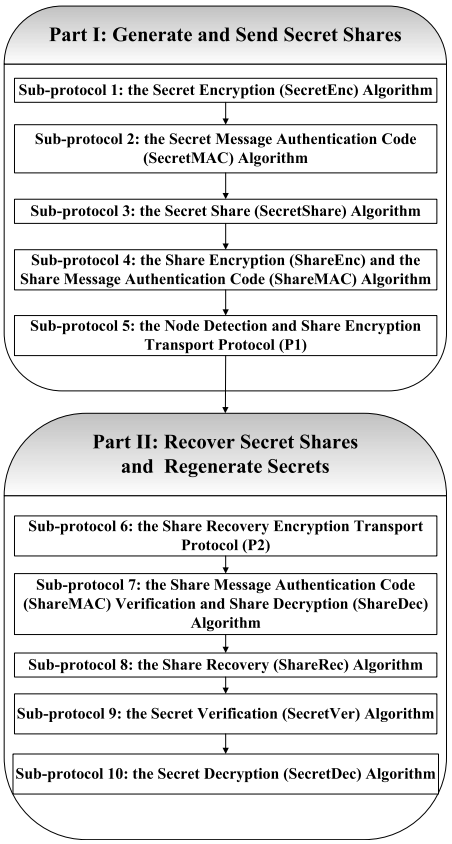


FIGURE 2. Divisions and notations of ten sub-protocols.

As is shown in (1), $ENC : X_1 \times X_2 \rightarrow Y$ represents any general encryption algorithm (e.g., AES, which is off-the-shelf), where the first input X_1 (here is PWD_A) represents symmetric encryption keys, the second input X_2 (here is $PriKey_A$) represents plaintexts to be encrypted and the output Y (here is C) represents encrypted ciphertexts. Moreover, m represents the length of PWD_A, L represents the length of $PriKey_A$.

Furthermore, in order to prevent attackers' collision conflicts, both the password PWD_A and the private key $PriKey_A$ can be extended by adding superfluous messages (e.g., *NodeA*'s username $USERNAME_A$, *Device ID*, memory questions $Q1, Q2, Q3$, etc.) to initial messages. For example, PWD_A can be extended as $(PWD_A||USERNAME_A)$ when $PriKey_A$ can be extended as $(PriKey_A||Device\ ID)$.

2) SUB-PROTOCOL 2: THE SECRET MESSAGE AUTHENTICATION CODE (SECRETMAC) ALGORITHM

During reconstructions in Sub-protocol 9, ciphertext C must be checked for its correctness. To design this, *SecretMAC Algorithm* in Sub-protocol 2 aims to add some integrity checking information or an integrity check of the share at the end of C. The (2) is listed below.

$$\begin{aligned} MAC &\Leftarrow Hash(C||Q1) \\ D &\Leftarrow C||MAC \end{aligned} \tag{2}$$

TABLE 2. Notations.

Notation	Description
$NodeA$	the private-key distribution node
$NodeB$	the private-key share-holder node
PWD_A	the password of $NodeA$
$USERNAME_A$	the username of $NodeA$
$PriKey_A$	the private key of $NodeA$
$PriKey_{B[i]}$	the private keys of $NodeB$
$\|message\|$	the binary length of message
$Q1, Q2, Q3$	the memory question 1, 2, 3
C	the private-key encrypted ciphertexts
L	$L = \|PriKeyA\| = \|C\|$ =the length of C
MAC	the message authentication code of C
L_{MAC}	the length of MAC
D	$D = C\|\|MAC$ with $\|D\| = \|C\| + \|MAC\|$
$Algorithm_{ENC}$	general encryption algorithms
$Algorithm_{DEC}$	general decryption algorithms
P	a random big prime number, $\|P\| > L$
$X[i]$	the horizontal ordinate
$Y[i]$	the longitudinal coordinates
$D[i]$	$D[i] = X[i]\|\|Y[i]$
$D'[i]$	the private-key encrypted message of $D[i]$
$MAC[i]$	the MAC of $D[i]$
REQ	the package of requests with "00001111"
ACK	the package of acknowledgements "11110000"
SND	the package of share-sending messages "10101010"
RCR	the package of share-recovery requests "11001100"
RTN	the package of share-return replys "00110011"
$ShareKey$	the share encryption private key
$SessionKey$	the share-transportation encryption private key
$PairwiseKey$	the share-collection encryption private key
(t, n)	t is the threshold value, n is the number of shares

As is shown in (2), $Q1$ is a memory question chosen by *NodeA*. *MAC* represents the integrity checking information to be added at the end of C. Moreover, the output result D is the "secret" to be divided into several shares in following sub-protocols.

3) SUB-PROTOCOL 3: THE SECRET SHARE (SECRETSHARE) ALGORITHM

Sub-protocol 2 returns ciphertexts with its integrity checking information D to maintain correctness of secrets, while Sub-protocol 3 aims to divide D into several shares to provide recoverability for secrets.

First, *NodeA* chooses a random natural number n to be the number of shares, a random natural number t to be the threshold value (where $0 < t < n$) and a random big prime number P (where the length of $P = L + L_{MAC} + 1$). Then *NodeA* chooses any (t, n) secret sharing protocol (e.g., Shamir (t, n) protocol, RSA (t, n) protocol, CRT (t, n) protocol, etc) to divide n shares of D. Resulting shares are named after $D[1], D[2], \ldots, D[n]$. Details of *SecretShare* Algorithm are shown in Algorithm 1:

After generating Algorithm 1, *NodeA* sends all secret shares $D[i] = X[i] + Y[i]$ to *NodeB*[i]s, where $|D[i]| = |X[i] + Y[i]| = 2|P|$.

In Algorithm 1, n and t are determined by current measured network state and system design parameters according to the Node Detection Protocol (P1) in Sub-protocol 5. Moreover, as Goal 2 in Section III requested, P shall be stored locally as a sysytem parameter.

Algorithm 1 The *SecretShare* Algorithm

Input: D, t, n, C, P, where $t, n \in N$, $1 \leq t < n$, P is a random big prime number with $|P| = L + L_{MAC} + 1$

Output: $D[i] = (X[i], Y[i])$, where $i = 1, 2, \ldots, n$, $X[i], Y[i] \in [0, P-1]$, $|X[i]| = |Y[i]| = |P|$

1 Choose $t-1$ random numbers $a[1], a[2], \ldots, a[t-1] \in \{0,1\}^L$, where $a[i] \in [0, 2^L - 1]$, $i = 1, 2, \ldots, t$.

2 Choose n random numbers $X[1], X[2], \ldots, X[n] \in [1, 2^L - 1]$.

3 For every input value of step1 and step2, go through function $F(X) = a[t-1]X^{t-1} + a[t-2]X^{t-2} + \ldots + a[1]X + C \bmod P$, and return $Y[i] \Leftarrow F(X[i])$.

4) SUB-PROTOCOL 4: THE SHARE ENCRYPTION (SHAREENC) AND THE SHARE MESSAGE AUTHENTICATION CODE (SHAREMAC) ALGORITHM

Considering that "secret" D has been divided into n shares in Sub-protocol 3, the next step is to distribute these shares to $NodeB[i]$s. However, to avoid attackers spying on transportation channels to filch shares, we design Sub-protocol 4 to defend against it.

Algorithm 2 The *ShareEnc* & *ShareMAC* Algorithm

Input: PWD_A, $Q2$, $Q3$, $USERNAME_A$, ID, $D[i]$, where $i = 1, 2, \ldots, n$.

Output: $E[i]$

1 Input PWD_A, $Q2$, $Q3$, ID and $USERNAME_A$ to go through hash functions, and return $ShareKey \Leftarrow Hash(PWD_A||Q2||USERNAME_A)$, $TAG_A \Leftarrow Hash(Q3||ID)$.

2 Input $ShareKey$, ID and $D[i]$ to go through encryption function ENC, *and return* $D'[i] \Leftarrow ENC(ShareKey, D[i]||ID)$.

3 Input $D'[i]$ and $ShareKey$ to go through hash functions, *and return* $MAC[i] \Leftarrow Hash(D'[i]||ShareKey)$, $i = 1, 2, \ldots, n$.

4 Input $D'[i]$, $MAC[i]$, TAG_A and return $E[i] \Leftarrow D'[i]||MAC[i]||TAG_A$.

In Algorithm 2, *NodeA* uses the same encryption function *ENC* as that in Sub-protocol 1. Moreover, considering that the output length of hash function *Hash* may be longer than the length of first input in function *ENC*, *NodeA* should cut partial bit-streams from the output of *Hash* to match *ENC*'s input. Besides, as Goal 2 in Section III requested, *NodeA* shall store *ShareKey* dynamically. In other words, *ShareKey* should be generated in real time during Algorithm 2 and be destroyed as soon as it finishes.

5) SUB-PROTOCOL 5: THE NODE DETECTION AND SHARE ENCRYPTION TRANSPORT PROTOCOL (P1)

$P1$ is a three-round protocol which includes a Diffie-Hellman Key Agreement Protocol, a node-detecting protocol and a share encryption transport protocol.

First, *NodeA* broadcasts its request by sending package *REQ*. Then every *NodeB* resends the package *ACK* to *NodeA*. In this process, package *REQ* and package *ACK* constitute the Diffie-Hellman Key Agreement Protocol. Besides, *NodeA* sends the share which is encrypted by the one-off negotiation session key to $NodeB[i]$s. In this way, *NodeA* and $NodeB[i]$s negotiate with each other to distribute $NodeA's$ secret shares.

The procedures are shown in Algorithm 3 and the details are as below:

Algorithm 3 P1 Algorithm

Input: $REQ, REQ_{ID}, REQ_{ADDR}, ACK, ACK_{ID}, ACK_{ADDR}$, $PriKey_{B[i]}, PWD_A, E[i]$, where $i = 1, 2, \ldots, n$

Output: *NegotiationProcess*

1 $NodeA \rightarrow * :< REQ, REQ_{ID}, REQ_{ADDR} >$.

2 $NodeB[i] \rightarrow NodeA$:

$$\left\langle \begin{matrix} ACK, ACK_{ID}, REQ_{ADDR}, ACK_{ADDR}, \\ g^{Hash(PriKey_{B[i]})} modP, \\ g^{Hash(REQ_{ID}||PriKey_{B[i]})} modP \end{matrix} \right\rangle$$

3 $NodeA \rightarrow NodeB[i]$:

$$\left\langle \begin{matrix} SND, ACK_{ID}, REQ_{ADDR}, ACK_{ADDR}, \\ g^{Hash(PWD_A)+\triangle} modP, \\ g^{Hash(PWD_A)+\alpha\cdot\triangle} modP, \\ ENC(SessionKey[A, B[i]], \\ ENC(PairwiseKey[A, B[j]], E[i])) \end{matrix} \right\rangle$$

where
$PairwiseKey[A, B[j]] = g^{Hash(PriKey_{B[j]})\cdot Hash(PWD_A)} modP$,
$SessionKey[A, B[i]] = g^{Hash(REQ_{ID}||PriKey_{B[i]})\cdot(Hash(PWD_A)+\triangle)} modP$

1) *NodeA* broadcasts its request package in (3).

$$A \rightarrow * :< REQ, REQ_{ID}, REQ_{ADDR} > \quad (3)$$

where *REQ* represents the package of requests with code "00001111", REQ_{ID} represents a random integer number at the length of 24 bits, REQ_{ADDR} represents $NodeA'$s IP address.

2) After receiving $NodeA's$ request package, $NodeB[i]$ figures out

$$g^{Hash(PriKey_{B[i]})} modP \quad (4)$$

which represents $NodeB[i]'s$ Diffie-Hellman storage key to be transported and

$$g^{Hash(REQ_{ID}||PriKey_{B[i]})} modP \quad (5)$$

which represents $NodeB[i]'s$ Diffie-Hellman session key to be transported. Then $NodeB[i]$ resends its response package to $NodeA$:

$$B[i] \rightarrow A : \left\langle \begin{array}{c} ACK, ACK_{ID}, \\ REQ_{ADDR}, ACK_{ADDR}, \\ g^{Hash(PriKey_{B[i]})} modP, \\ g^{Hash(REQ_{ID}||PriKey_{B[i]})} modP \end{array} \right\rangle \quad (6)$$

where ACK represents the package of acknowledgements with code "11110000", $REQ_{ID} = REQ_{ID} + 1$ with $|REQ_{ID}| = |REQ_{ID}|$, REQ_{ADDR} represents $NodeA'$s IP address, ACK_{ADDR} represents $NodeB[i]'s$ IP address. Moreover, P is a prime number and g is an element of Z_P with the order of q.

3) After receiving $NodeB[i]'s$ response package, $NodeA$ chooses a nonce $\triangle$, a big prime number q which divides $P-1$, a nonce α between 1 and q, and g which is the element of order q in Z_p.
Then $NodeA$ figures out (7),(8),(9) and(10):

$$g^{Hash(PWD_A)+\triangle} modP \quad (7)$$

where $\triangle$ is a random number which represents a random key to protect PWD_A from brute-force attacks. Besides, $\triangle$ should not stored in terminals of $NodeA$.

$$g^{Hash(PWD_A)+\alpha\cdot\triangle} modP \quad (8)$$

where α is a random number between 1 and q ($\alpha \in (1, q]$) and must be stored in terminals of $NodeA$.

$$PairwiseKey[A, B[j]] = g^{Hash(PriKey_{B[j]})\cdot Hash(PWD_A)} modP \quad (9)$$

where j is a random number between 1 and n ($j \in [1, n]$). $PairwiseKey$ helps key-entropy expansions, making it more difficult for key-attempting attacks. Furthermore, any $NodeB[i]$ can not speculate the α of $NodeB[j]$ through (7),(8) because of difficulties in solving discrete logarithm problems in (9).
After introducing (9), $NodeA$ uses $Pairwisekey[A, B[j]]$ to encrypt $E[i]$ through function ENC and returns the result of
$ENC(Pairwisekey[A, B[j]], E[i])$.

$$SessionKey[A, B[i]] = g^{Hash(REQ_{ID}||PriKey_{B[i]}) \cdot (Hash(PWD_A)+\triangle)} modP \quad (10)$$

By introducing one-off session elements REQ_{ID}, $SessionKey$ can prevent network-spying behaviors.
Then $NodeA$ encrypts $ENC(Pairwisekey[A, B[j]], E[i])$ with $SessionKey$ and returns the result of $ENC(SessionKey, ENC(PairwiseKey[A,B [j]], E[i]))$.
Finally, $NodeA$ sends its transfer package to $NodeB[i]$:

$$A \rightarrow B[i] : \left\langle \begin{array}{c} SND, ACK_{ID}, \\ REQ_{ADDR}, ACK_{ADDR}, \\ g^{Hash(PWD_A)+\triangle} modP, \\ g^{Hash(PWD_A)+\alpha\cdot\triangle} modP, \\ ENC(SessionKey[A, B[i]], \\ ENC(PairwiseKey[A, B[j]], E[i])) \end{array} \right\rangle \quad (11)$$

After receiving $ENC(SessionKey, ENC(PairwiseKey[A, B[j]], E[I]))$, $NodeB[i]$ decrypts $ENC(PairwiseKey[A, B[j]], E[I])$ with $SessionKey$.

Because of requests in Goal 2 in Section III, $NodeB[i]$ should store $ENC(PairwiseKey[A, B[j]], E[I])$, $g^{Hash(PWD_A)+\triangle} modP$ and $g^{Hash(PWD_A)+\alpha\cdot\triangle} modP$ locally to make verifications for share holders in Sub-protocol 6.

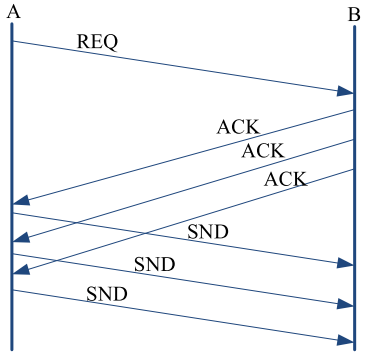


FIGURE 3. Transferring in P1.

As is shown in Fig. 3, in order to improve transferring efficiency, $NodeB[i]$s shall repeat sending response packages to $NodeA$. Then $NodeA$ chooses to resend SND or not.

6) SUB-PROTOCOL 6: THE SHARE RECOVERERY ENCRYTION TRANSPORT PROTOCOL (P2)

In order to retrieve t valid shares, we need to design an interactive protocol $P2$ to for $NodeB[i]$ to verify $NodeA'$s Identification. $P2$ is a four-round protocol, whose functions include share-recovery request RCR and share-return reply RTN. The design of $P2$ is based on Zero-knowledge Proof (ZkP) to prove the ownership of PWD without leaking any details about it. Moreover, $P2$ adopts the design idea of Schnorr Identification Protocol.

The procedures are shown in Algorithm 4 and the details are as below:

1) $NodeA$ first figures out a commit which represents a one-off session key for negotiation.

$$Commit = g^r modP \quad (12)$$

where r is a dynamic nonce which will be destroyed after $P2$ finishes and g is Sub-protocol 5′s element of order q in Z_p.
Then $NodeA$ figures out a challenge for $Nodeb[i]$ as an intermediate variable.

$$Challenge = Hash(RCR_{ID}||g||Commit) \quad (13)$$

where RCR_{ID} represents a integer nonce at the length of 24 bits.
On the basis of (13), $NodeA$ figures out a response message.

$$Response = r + Hash(PWD_A) * ChallengemodP \quad (14)$$

Algorithm 4 P2 Algorithm

Input: $RCR, RCR_{ID}, REQ_{ADDR}, RTN, RTN_{ID}, ACK_{ADDR}, g, P,$ $PriKey_{B[i]}$, where $i = 1, 2, \ldots, n$

Output: *NegotiationProcess*

1 $NodeA \rightarrow *:$

$$\left\langle \begin{matrix} RCR, RCR_{ID}, REQ_{ADDR}, \\ Commit, Response, \alpha \end{matrix} \right\rangle$$

2 $NodeB[i] \rightarrow NodeA:$

$$\left\langle \begin{matrix} RTN, RTN_{ID}, REQ_{ADDR}, ACK_{ADDR} \\ g^{Hash(RTN_{ID}||PriKey_{B[i]})} modP, \\ g^{Hash(PriKey_{B[i]})} modP \\ ENC(SessionKey[A, B[i]], \\ ENC(PairwiseKey[A, B[j]], E[i])) \end{matrix} \right\rangle$$

Finally, *NodeA* broadcasts its share-recovery request package *RCR*:

$$A \rightarrow *: \left\langle \begin{matrix} RCR, RCR_{ID}, REQ_{ADDR} \\ Commit, Response, \alpha \end{matrix} \right\rangle \quad (15)$$

where *RCR* represents the package of share-recovery requests with code "11001100" and REQ_{ADDR} represents *NodeA*'s IP address, α is the one stored in terminal of *NodeA* in Sub-protocol 5.

2) After receiving the broadcast from *NodeA*, *NodeB*[*i*] figures out (13) again.
Let $v_1 = g^{Hash(PWD_A)+\triangle} modP$ and $v_2 = g^{Hash(PWD_A)+\alpha\cdot\triangle} modP$. Since v_1, v_2 and *ENC* (*PairwiseKey*[*A*, *B*[*j*]], *E*[*I*]) were stored locally in Sub-protocol 5, *NodeB*[*i*] merely needs to verify whether the left and the right sides of (16) are equal.

$$(Commit * v_1^{Challenge} / g^{Response})^{\alpha} = Commit * v_2^{Challenge} / g^{Response} modP \quad (16)$$

If the answer is true, *NodeB*[*i*] makes the judgment that *NodeA* is identification-verified and then resends his shares along with the share-return reply package *RTN*. That is $B[i] \rightarrow A:$

$$\left\langle \begin{matrix} RTN, RTN_{ID}, \\ REQ_{ADDR}, ACK_{ADDR} \\ g^{Hash(RTN_{ID}||PriKey_{B[i]})} modP, \\ g^{Hash(PriKey_{B[i]})} modP \\ ENC(SessionKey[A, B[i]], \\ ENC(PairwiseKey[A, B[j]], E[i])) \end{matrix} \right\rangle \quad (17)$$

where *RTN* represents the package of share-return replys with code "00110011", $RTN_{ID} = RCR_{ID} + 1$, REQ_{ADDR} represents *NodeA*'s IP address, ACR_{ADDR} represents *NodeB*'s IP address, $g^{Hash(PriKey_{B[i]})}$ represents the storage key in Diffie-Hellman Key Agreement Protocol and $g^{Hash(RTN_{ID}||PriKey_{B[i]})} modP$ represents the session key in Diffie-Hellman Key Agreement Protocol.

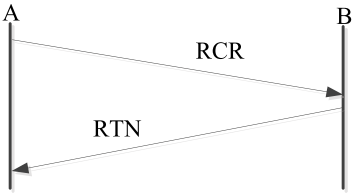


FIGURE 4. Transferring in P2.

Fig. 4 shows the two-round interaction between *NodeA* and *NodeB*[*i*].

7) SUB-PROTOCOL7: THE SHARE MESSAGE AUTHENTICATION CODE (SHAREMAC) VERIFICATION AND SHARE DECRYPTION (SHAREDEC) ALGORITHM

In Sub-protocol 6, *NodeA* receives packages of *RTN* from all *NodeB*[*i*]s. Then *SessionKey*[*A*, *B*[*i*]] can be reconstructed as below.

$$\begin{aligned} SessionKey[A, B[i]] &= (g^{Hash(RTN_I D||Prikey_{B[i]})}) \\ &= g^{Hash(RTN_{ID}||PriKey_{B[i]})*r} modP \end{aligned} \quad (18)$$

NodeA can decrypt *ENC*(*SessionKey*[*A*, *B*[*i*]], *ENC* (*PairwiseKey*[*A*, *B*[*i*]], *E*[*i*])) to get

$$ENC(PairwiseKey[A, B[i]], E[i])$$

Furthermore, *NodeA* shall list all *PairwiseKey*s to decrypt function *ENC* because Goal 2 in Subsection II-C requests that *NodeA* should not store any relationships between *PairwiseKey*s and *NodeB*[*i*]s. The only method for *NodeA* to match them is traverse.

If $|TAG_A| = TAG_A = Hash(Q3||ID)$, the decryption result $PairwiseKey[A, B[i]] = g^{Hash(PriKey_{B[i]})* \ Hash(PWD_A)} modP$ is correct. Furthermore, the average share-decryption times shall be *n*.

1) **ShareMac Algorithm**
To verify the message authentication code $MAC'[i]$ and the ownership of *ShareKey* for each $D'[i]$, we can redefine above decryption result as $FRONT||BACK$. Take the former length of $|ENC(.)| = L$ bit strings as *FRONT* and the latter length of $|MAC[i] = L_{MAC}|$ bit strings as *BACK*, then judge whether or not

$$Hash(FRONT||ShareKey) = BACK \quad (19)$$

If the answer is no, $D'[i]$ turns out to be invalid and should be destroyed. Otherwise, the ownership of *ShareKey* is proven and *NodeA* can recover $D[i]||ID$ from $D'[i]$. Besides, *NodeA* should verify whether *ID*s are consistent for the desired $PriKey_A$s. If the verification result is true, *NodeA* can proceed to the Share Decryption (ShareDec) Algorithm as below.
Since *NodeA* can not store *ShareKey* locally because of Goal 2 in Subsection II-C, *NodeA* can generate

ShareKey from PWD_A, $Q2$ and $USERNAME_A$ again.

$$ShareKey \Leftarrow Hash(PWD_A||Q2||USERNAME_A) \quad (20)$$

2) **ShareDec Algorithm**
The Share Decryption (*ShareDec*) Algorithm here is closely corresponding to the Share Encryption (*ShareEnc*) Algorithm introduced in Sub-protocol 4, where the decryption function *DEC* can recover the second input of the encryption function *ENC*. That is,

$$DEC(ENC(ShareKey, X)) = X \quad (21)$$

Therefore, $D[i]$ can be recovered as following.

$$D[i] \Leftarrow DEC(ShareKey, D'[i]) \quad (22)$$

8) SUB-PROTOCOL 8: THE SHARE RECOVERY (SHAREREC) ALGORITHM

To verify the integrity of D to be recovered, *NodeA* must confirm that there are at least t valid $D[i]$s. According to the *Shamir* Interpolation Secret Sharing Recovery Algorithm, *NodeA* can recover D from $D[i] = X[i]||Y[i]$, where $i = 1, 2, \ldots, n$.

Algorithm 5 ShareRec Algorithm

Input: t; n; $(X[i], Y[i])$, where $i = 1, 2, \cdots, n$; $D[i] \in [0, P-1]$ from Sub-protocol 3; P is a big prime number where $|P| > L + L_{MAC}$

Output: D

```
for i = 1; i ⩽ n; i++ do
    t = rand() mod n +1;
    j ⇐ 1;
    XX[j] ⇐ X[t];
    YY[j] ⇐ Y[t];
    j=j+1;
end
for K = 1; K ⩽ t; K++ do
    PI ⇐ 1;
    for j = 1; j < t; j++ do
        A ⇐ XX[j] * (XX[j] − XX[K])^{−1} mod P;
        PI ⇐ PI * A;
    end
    B[K] ⇐ PI;
end
D ⇐ 0;
for K = 1; K ⩽ t; K++ do
    D ⇐ D + B[K] * YY[K] mod P;
end
return D;
```

Define a secret sharing recovery function *ShareRec* in Algorithm 5. This algorithm is based on *Lagrange* interpolation method, which constructs the polynomial function D without figuring out coefficients in systems of liner equations.

9) SUB-PROTOCOL 9: THE SECRET VERIFICATION (SECRETVER) ALGORITHM

Corresponding to the SecretMAC Algorithm in Sub-protocol 2, this algorithm is designed to judge the correctness of encrypted ciphertext D.

Sub-protocol 2 proposed that

$$MAC \Leftarrow Hash(C||Q1)$$
$$|MAC| = |Hash(C||Q1)| = L_{MAC}$$
$$D \Leftarrow C||MAC$$

Like Sub-protocol 7, we can redefine D as $FRONT'||BACK'$. Take the former length of L bit strings as $FRONT'$ and the latter length of L_{MAC} bit strings as $BACK'$, then judge whether or not

$$BACK' = Hash(FRONT'||Q1) \quad (23)$$

If the answer is yes, D turns out to be correct. Therefore, C is the former length of L bit strings in D.

10) SUB-PROTOCOL 10: THE SECRET DECRYPTION (SECRETDEC) ALGORITHM

Since Sub-protocol 9 reconstructs C, *NodeA* can uses his password PWD_A to decrypt C and returns $PriKey_A$, where the decryption function *DEC* is corresponding to the encryption function *ENC* in Sub-protocol 1.

$$PriKey_A \Leftarrow DEC(PWD_A, C), \quad where\ PWD_A \in \{0,1\}^m, \quad PriKey_A, C \in \{0,1\}^L. \quad (24)$$

V. ANALYSIS

A. SECURITY ANALYSIS

We introduced twelve adversary models in Subsection III-B from the aspects of possible attacks in brief. And the former five models attack on (t, n) secret sharing algorithms (A) when the latter seven models attack on network protocols (B). Here we give specific introductions to these adversary models and analyze the security of our schemes.

1) ATTACKS ON (t, n) SECRET SHARING ALGORITHMS

Attacks on (t, n) secret sharing algorithms are composed of five models below. Since there exist no professional definitions towards following attack models, we first give specific definitions to introduce them and then prove that our schemes can defend against these attacks.

1) **Compromise Attacks**
Definition: A private-key share-holder node ($NodeB[i]$) conducts a single-node defect. In other words, captured *NodeB* attempts to infer $PriKey_A$ through his own share.

Proof: Such an attacker merely possesses an encrypted single share without its corresponding decrypted share. The probability of speculation is $\frac{1}{2}^m$, where $m = |PWD_A|$. Moreover, even if the decrypted share is speculated, this attacker can not recover $PriKey_A$ because of (t, n) secret sharing recovery algorithm. This algorithm requests that $PriKey_A$ can not

be recovered until at least t correct shares are collected. Therefore, the possibility of attack-successfully rate is $\frac{1}{2}^{L}$, whose limit approaches to 0. Hence, our proposed schemes can defend against Compromise Attack. ■

2) **Collusion Attacks**
Definition: All private-key share-holder nodes (*NodeB*[*i*]s) collude to defect completely. That is to say, the Byzantine failure occurs to speculate $PriKey_A$.

Proof: All shares are encrypted by *ShareKey* which is merely known by *NodeA*. So even if all *NodeB*[*i*]s are compromised, they only possess the encrypted shares instead of the decrypted ones. In this way, the security of $PriKey_A$ depends on that of *ShareKey*. Moreover, *ShareKey* depends on PWD_A, regulating the security of *ShareKey* to that of PWD_A tightly. Furthermore, the encryption algorithm *ENC* proposes that the security of $PriKey_A$ is also reduced to the security of PWD_A. Therefore, if PWD_A is secure (which has already been assumed in Subsection III-B), *ShareKey* is secure and the encrypted share cannot be decrypted. Consequently, Collusion Attack is invalid towards our proposed schemes. ■

3) **History Analysis Attack**
Definition: All private-key share-holder nodes (*NodeB*[*i*]s) not only collude to defect completely, but also attempt to speculate $PriKey_A$ through multiple history shares.

Proof: It is impossible for any single *NodeB*[*i*] to speculate $PriKey_A$ independently, because nonces (α and $\triangle$) are different between every two previous shares. Therefore, there exist no linkages among any *NodeB*[*i*]′s previous shares.
On the basis of this, History Analysis Attack shall be degenerated into Collusion Attack. Furthermore, it is impossible for all *NodeB*[*i*]s together to speculate $PriKey_A$ through multiple history shares because Collusion Attack has been proven unsuccessful. ■

4) **Malicious Behaviours**
Definition: A private-key share-holder node (*NodeB*[*i*]) behaves maliciously (such as tampering initial shares, forging his *ID*, generating wrong keys and so on).

Proof: Once a private-key share-holder node (*NodeB*[*i*]) tries to tamper his shares, he shall be found because the *MAC* of share does not match the tampered one. Besides, neither a fate *ID* or a wrong key cannot go through the integrity detection. That is because (t, n) secret sharing recovery algorithm allows at most $n - t$ *NodeB*[*i*]s to behave maliciously. ■

5) **Compromise of Entities with NodeA**
Definition: NodeA′s entities (Storage terminals such as smart phones, laptops, desktops and so on) are breached by attackers, leading to the exposure of all locally-stored information.

Proof: Such an attack is the strongest security assumption for the opponent, which has been reduced to the basic security assumption, namely the security of PWD_A. However, even if entities of *NodeA* are compromised, $PriKey_A$ is still secure as long as PWD_A is proven secure. That is because PWD_A can be merely recorded in *NodeA*′s brain, which will never be stolen of on *NodeA*′s initiatives. ■

2) ATTACKS ON NETWORK PROTOCOLS

Attacks on network protocols are composed of seven models below. Since there exist professional definitions towards some attack models, we shall not redefine them again. Then we give proofs that our schemes can defend against these attacks.

6) **Eavesdropping Attack**
Proof: Since each conversation in Sub-protocol 5 and 6 independently generates one-off Deffie-Hellman *SessionKey*s to encrypt encryption shares and these *SessionKey*s are closely related to ACK_{ID}, RCR_{ID} or RTN_{ID}, attackers can not eavesdrop conversation details. Therefore, our proposed scheme can defend against Eavesdropping Attack. ■

7) **Reflect Attack**
Definition: By flipping the sender and the receiver, attackers resend the challenge posed by the sender to the sender, attempting to acquire correct responses from the sender. Then attackers can employ these responses to answer the challenge posed by the previous sender.

Proof: This attack is ineffective because all response packages in our design do not contain any corresponding knowledge (response) required for future interactions, thus the receiver cannot obtain any useful answers by reversing senders' questions. Moreover, Reflection Attacks are easy to detect because some packages retain REQ_{ID}, ACK_{ID}, $REQ_{ADDRESS}$, $ACK_{ADDRESS}$, RCR_{ID}, RTN_{ID} and so on. In all, our proposed scheme can defend against Reflect Attacks. ■

8) **Interleave Attack**
Definition: Interleave Attack is to seek the completion of a single protocol through information among packages in multiple concurrent protocols.

Proof: Since ID messages (e.g., REQ_{ID}, ACK_{ID}, RCR_{ID} and RTN_{ID}) are preserved in conversation packages, parallel protocols in our schemes can be distinguished through IDs. That is, our proposed schemes can defend against Interleave Attack. ■

9) **Replay Attack**
Definition: Attackers attempt to lessen network protocol efficiency by replaying packages.

Proof: Since ID messages like REQ_{ID} and RCR_{ID} are included in conversation packages, it is easy to detect possible Replay Attacks and ignore the repeated ones. Moreover, we can list IP addresses of Replay Attack nodes in the blacklist. ■

10) **Combination Attack**
Definition: Such an attack combine above attack models together.
Proof: Since we have proved that above attack models are invalid for our schemes, no matter how this attack combines others can we confirm its invalidity. ■

11) **DDoS Attack**
Proof: If a malicious node attempts to modify loads in *RTN*, our proposed schemes can verify the correctness of *RTN* through $MAC[i]$, which can avoid *NodeA* from using wrong shares to reconstruct "secret", so as to minimize times of reconstruction calculation. Therefore, our proposed scheme can reduce harm caused by DDoS Attacks. ■

12) **Sybil Attack**
Proof: Even if the node changes its IP address frequently, no additional loss will be caused because *NodeA* merely require a sufficient number of IP addresses instead of all of that to return their correct shares. ■

B. PERFORMANCE ANALYSIS

1) PERFORMANCE INDICATORS

Main performance indicators include communication performance and computing performance. However, both communication delays and computing powers can be ignored here. The former reason is that in most cases communication process does not directly affect the user experience while the latter reason is that there hardly exist low-power consumption devices on secret sharing technologies. Moreover, the storage costs are also out of consideration because current devices are cheap.

2) ANALYSIS

For better performances in interactions, we shall minimize four main indicators in interaction rounds, encryption and decryption calculations, size of protocol packages and semantics.

1) Minimum number of interaction rounds
In our schemes, P1 is a 3-round protocol when P2 is a 2-round protocol. And both network transmission costs and latency of them are low. To begin with, traditional Diffie-Hellman Key Agreement Protocol requires at least 2 rounds. However, P1 cannot be changed to a 2-round protocol unless shares cannot be transferred. Therefore, P1 already reaches its minimum round number protocol. Moreover, P2 cannot be changed to a one-round protocol because its functions must include zero-knowledge authentication, key negotiation and data transfer, which allows 2 to be the minimum round of P2. Consequently, both P1 and P2 have reached minimum number of interaction rounds.
2) Minimum encryption and decryption calculations
On the basis of security, times of encryption calculation have been minimized.
3) Minimum size of protocol packages
The package designs in P1 and P2 reach the minimum size of package without the premise of security and semantic clarity, so as to minimize both network transmission costs and transmission delays.
4) Minimum semantics
As single semantic protocols, both P1 and P2 have reached minimum semantics.

3) RELIABILITY ANALYSIS

Assume that *NodeA* receives m $D[i]s$ from *NodeB*$[i]s$, where there are p $D[i]s$ out of m are valid, i.e., the proportion of valid shares is $\frac{p}{m}$ ($p \geq t$, t represents the threshold value).

Theorem 1: If the proportion of valid shares is $\frac{p}{m}$, the probability of recovering the secret D is

$$Pr = \frac{(C_{m-p}^{0} + C_{m-p}^{1} + \cdots + C_{m-p}^{m-p}) \cdot (C_{p}^{t} + C_{p}^{t+1} + \cdots + C_{p}^{p})}{(C_{m}^{t} + C_{m}^{t+1} + \cdots + C_{m}^{m})} \tag{25}$$

Proof: To begin with, there are C_m^t permutations on condition of selecting t shares from m shares. Similarly, there are $C_m^t, C_m^{t+1}, \cdots, C_m^m$ permutations on condition of selecting $t, t+1, \cdots, m$ shares from m shares.

m shares can be further divided into two categories, where there are $m-p$ invalid shares (IVS) and p valid shares (VS). Conditions of reconstructing D are listed in Table 3. ■

TABLE 3. Permutations of reconstructing *D*.

Number of IVS Selected	IVS Permutations	VS Permutations
0	C_{m-p}^{0}	$C_p^t + C_p^{t+1} + \cdots + C_p^p$
1	C_{m-p}^{1}	$C_p^t + C_p^{t+1} + \cdots + C_p^p$
2	C_{m-p}^{2}	$C_p^t + C_p^{t+1} + \cdots + C_p^p$
$\vdots$	$\vdots$	$\vdots$
$m-p$	C_{m-p}^{m-p}	$C_p^t + C_p^{t+1} + \cdots + C_p^p$

Hence, there are $(C_{m-p}^{0} + C_{m-p}^{1} + \cdots + C_{m-p}^{m-p}) * (C_p^t + C_p^{t+1} + \cdots + C_p^p)$ permutations on condition of reconstructing D. The proportion of reconstruction is

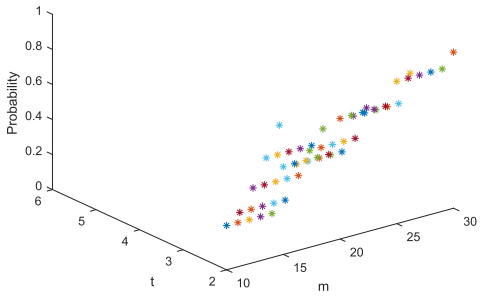
$$Pr = \frac{(C_{m-p}^{0} + C_{m-p}^{1} + \cdots + C_{m-p}^{m-p}) \cdot (C_{p}^{t} + C_{p}^{t+1} + \cdots + C_{p}^{p})}{(C_{m}^{t} + C_{m}^{t+1} + \cdots + C_{m}^{m})}$$

4) SIMULATION RESULTS AND ANALYSIS

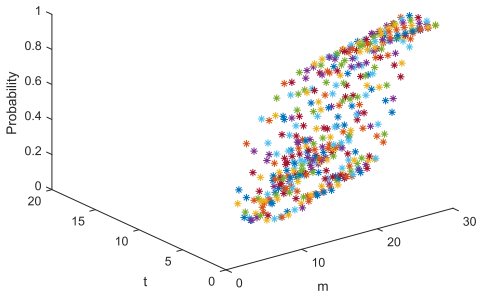
The probability of reconstruction in (25) illustrates relationships among p, m and t, thus allowing us to analyze it numerically.

Firstly, we separately set $\frac{p}{m} \in \{0.2, 0.4, 0.6, 0.8, 1\}$ as a fixed proportion and then simulate probability-variation trends resulting from m and t in Fig. 5.

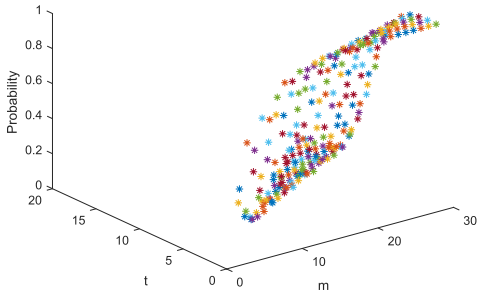
Secondly, we set $\frac{t}{p} = 1$ and then simulate probability-variation trends resulting from p and m in Fig. 6.



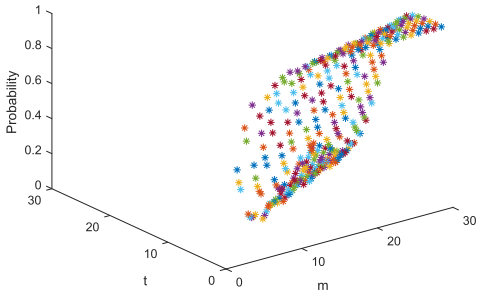
(a) Successful rates when p=0.2m



(b) Successful rates when p=0.4m

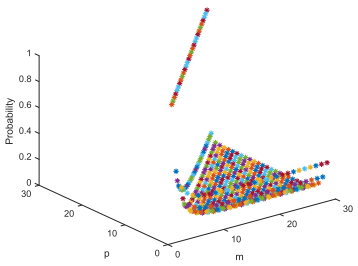


(c) Successful rates when p=0.6m

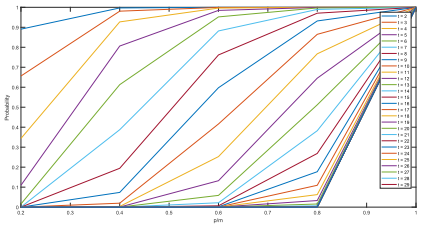


(d) Successful rates when p=0.8m

FIGURE 5. Successful rates of reconstructing D.



(a) 3-dimension



(b) 2-dimension

FIGURE 6. The probability-variation trends resulting from *p* and *m*.

Furthermore, we wonder that if it is feasible to rely on multiple reconstructions and majority voting schemes to replace integrity verifications. Therefore, security analysis is conducted as following.

Assume that $2 < t \leq m$,

- if there exist C_m^t reconstructions, we can generate C_m^t results where only one result is correct.
- if there exist C_m^{t+1} reconstructions, we can generate C_m^{t+1} results where C_{m-t}^1 results are correct.
- if there exist C_m^{t+x} reconstructions, we can generate C_m^{t+x} results where C_{m-t}^x results are correct. $(1 \geq x \geq m-t)$

Only when $C_{m-t}^x > \frac{1}{2}C_m^{t+x}$ can *NodeA* recover correct result D.

Therefore, in order to test the minimum value of x, we conduct numerical analysis in Fig. 7.

5) EXPERIMENT RESULTS AND ANALYSIS

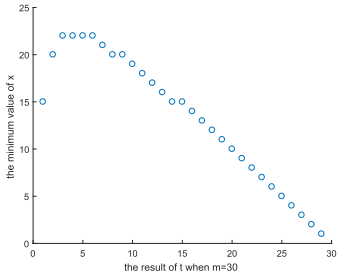
We separately tested time costs of generating shares from the secret (GenSha)and reconstructing the secret from shares (RecSec) under different (t, n) parameters.

System Parameters:

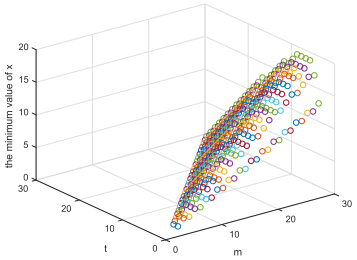
- The language environment of codes: Java.
- The computing platform: Android 6.0.1, Nubia z11minis, Qualcomm Snapdragon 625MSM8953, CPU frequency 2.0ghz, core number of eight cores, Qualcomm Adreno506 GPU model, 4GB RAM.

Experiment results are listed in Table 4.

Results in Table 4 show that orders of magnitude in delays are millisecond. Therefore, our proposed scheme is feasible.



(a) The trends of the minimum value of x resulted by t when m=30



(b) The trends of the minimum value of x resulted by t and m

FIGURE 7. The trends of the minimum value of x.

TABLE 4. Experiment Results of time costs.

	t=2,n=3		t=3,n=6	
Order	**GenSha**	**RecSec**	**GenSha**	**RecSec**
1	601511	235885	1140104	676719
2	528334	247188	1252813	651458
3	574114	241302	1235052	652187
4	508281	236719	1252605	642447
5	515104	238750	1255521	640990
6	524218	228438	1245781	689219
7	506042	253698	1303385	638802
8	513177	234323	1280156	634532
9	519427	239427	1239010	642605
10	529375	242343	1253750	650417
Average(ns)	524230.2	239807.3	1245817.7	651937.6
Average(ms)	0.52	0.24	1.25	0.65
	t=4,n=10		**t=6,n=20**	
Order	**GenSha**	**RecSec**	**GenSha**	**RecSec**
1	2396040	1277240	6033490	3103490
2	2414479	1254791	6020574	3101823
3	2368437	1260938	6121823	3133646
4	2360366	1264947	6166405	3134167
5	2347292	1263125	6012395	3155209
6	2369688	1247552	5839583	3160884
7	2420574	1256145	5672084	3178177
8	2396510	1257135	5642865	3152241
9	2349010	1269271	5614792	3149427
10	2341563	1291823	5993334	3115052
Average(ns)	2376395.9	1264296.7	5911734.5	3138411.6
Average(ms)	2.38	1.26	5.91	3.14

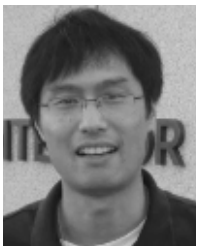
VI. CONCLUSION

In this paper, we proposed a secret-sharing-based key protection scheme for blockchain. We not only designed a basic CSC framework to solve disadvantages of traditional CSC modes (such as extra delays, transmission costs and information wastages) but also introduced a private-key distribution method to help recover lost private keys. Our scheme can defend against both attacks on secret sharing algorithms and attacks on network protocols theoretically. Furthermore, the interaction performances of our scheme are proven to be optimized. And real experiments illustrate that time costs in our scheme can be ignored since the orders of magnitude in delays are millisecond. The scheme can be used in any general scenarios where private keys need to be protected.

REFERENCES

[1] M. Lamoureux. *A Supply Chan Finance Primer*. Accessed: Jul. 29, 2019. [Online]. Available: http://www.esourcingwiki.com/index.php/A_Supply_Chain_Finance_Primer
[2] C. Yan and X. Zhangong, "Study on the information technology-based lean construction supply chain management model," in *Recent Progress in Data Engineering and Internet Technology* (Lecture Notes in Electrical Engineering), vol. 157. Berlin, Germany: Springer, 2012.
[3] J. Wang, P. Wu, X. Wang, and W. Shou, "The outlook of blockchain technology for construction engineering management," *Frontiers Eng. Manage.*, vol. 4, no. 1, pp. 67–75, 2017.
[4] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[5] T. Volety, S. Saini, T. McGhin, C. Z. Liu, and K.-K. R. Choo, "Cracking bitcoin wallets: I want what you have in the wallets," *Future Gener. Comput. Syst.*, vol. 91, pp. 136–143, Feb. 2019.
[6] B. Zhou, H. Li, and L. Xu, "An authentication scheme using identity-based encryption & blockchain," in *Proc. 2018 IEEE Symp. Comput. Commun. (ISCC)*, Natal, Brazil, Jun. 2018, pp. 556–561.
[7] J. Stanley. *Steganographic Bitcoin Seeds: Hiding Cash in Plain Sight*. Accessed: Jul. 29, 2019. [Online]. Available: https://incoherency.co.uk/blog/stories/steganographic-bitcoin-seeds.html
[8] O. Hosam, "Hiding bitcoins in steganographic fractals," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Louisville, KY, USA, Dec. 2018, pp. 512–519.
[9] Y.-L. Gao, X.-B. Chen, Y.-L. Chen, Y. Sun, X.-N. Niu, and Y.-X. Yang, "Secure cryptocurrency scheme based on post-quantum blockchain," *IEEE Access*, vol. 6, pp. 27205–27213, 2018.
[10] W. Yin, Q. Wen, W. Li, H. Zhang, and Z. Jin, "An anti-quantum transaction authentication approach in blockchain," *IEEE Access*, vol. 6, pp. 5393–5401, 2018.
[11] C.-Y. Li, X.-B. Chen, Y.-L. Chen, Y.-Y. Hou, and J. Li, "A new lattice-based signature scheme in post-quantum blockchain network," *IEEE Access*, vol. 7, pp. 2026–2033, 2019.
[12] Y. Tribis, A. E. Bouchti, and H. Bouayad, "Supply chain management based on blockchain: A systematic mapping study," in *Proc. MATEC Web Conf. (IWTSCE)*, Sep. 2018, Art. no. 20.
[13] Y. Wang, J. H. Han, and P. Beynon-Davies, "Understanding blockchain technology for future supply chains: A systematic literature review and research agenda," *Supply Chain Manage.*, vol. 24, no. 1, pp. 62–84, Jan. 2019.
[14] Y. Cui and H. Idota, "Improving supply chain resilience with establishing a decentralized information sharing mechanism," in *Proc. 5th Multidisciplinary Int. Social Netw. Conf. (MISNC)*, Saint-Etienne, France, Jul. 2018, Art. no. 23.
[15] S. Chen, R. Shi, Z. Ren, J. Yan, A. Shi, and J. Zhang, "A blockchain-based supply chain quality management framework," in *Proc. IEEE 14th Int. Conf. E-Bus. Eng. (ICEBE)*, Shanghai, China, Nov. 2017, pp. 172–176.
[16] Z. Gao, X. Lei, L. Chen, X. Zhao, Y. Lu, and W. Shi, "CoC: A unified distributed ledger based supply chain management system," *J. Comput. Sci. Technol.*, vol. 33, no. 2, pp. 237–248, Mar. 2018.
[17] P. Urien, "Crypto terminal based on secure element for consumer trusted blockchain transactions," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2019, pp. 1–2.
[18] X. Liang, S. Shetty, D. Tosh, Y. Ji, and D. Li, "Towards a reliable and accountable cyber supply chain in energy delivery system using blockchain," in *Security and Privacy in Communication Networks* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 255. Cham, Switzerland: Springer, 2018, pp. 43–62.

[19] Y. Kaga, M. Fujio, K. Naganuma, K. Takahashi, T. Murakami, T. Ohki, and M. Nishigaki, "A secure and practical signature scheme for blockchain based on biometrics," in *Information Security Practice and Experience* (Lecture Notes in Computer Science), vol. 10701. Cham, Switzerland: Springer, 2017, pp. 877–891.
[20] J. Camenisch, A. Lehmann, A. Lysyanskaya, and G. Neven, "Memento: How to reconstruct your secrets from a single password in a hostile environment," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 8617. Berlin, Germany: Springer, 2014, pp. 256–275.
[21] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online)," in *Proc. IEEE Eur. Symp. Secur. Prsivacy (EuroSP)*, Saarbrucken, Germany, Mar. 2016, pp. 276–291.



FENG XIONG received the Ph.D. degree in management science and engineering from the Wuhan University of Technology, China, in 2009. He was a Postdoctoral Fellow with the University of California at Santa Cruz, USA. He is currently an Associate Professor with the School of Business Administration, Zhongnan University of Economics and Law, China. His research interests include industrial engineering and supply chain management.

RUIYANG XIAO is currently pursuing the degree with the School of Computer Science and the School of Mathematics and Physics, China University of Geosciences, Wuhan, China. She has been pre-admitted by the University of Science and Technology of China (USTC). Her research interests include blockchain and privacy protection.

WEI REN (M'09) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, China. He was with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, USA, from 2007 to 2008, the School of Computer Science, University of Nevada at Las Vegas, USA, from 2006 to 2007, and the Department of Computer Science, The Hong Kong University of Science and Technology, from 2004 to 2005. He is currently a Professor with the School of Computer Science, China University of Geosciences (Wuhan), China. He has published more than 70 refereed articles, one monograph, and four textbooks. He has obtained ten patents. He is also a Senior Member of the China Computer Federation. He received five innovation awards.

RONGYUE ZHENG received the B.Eng., M.Eng., and Ph.D. degrees from the National University of Defense Technology, China. He is currently a Professor with the Faculty of Architecture, Civil Engineering and Environment, Ningbo University, China. His research interest includes smart construction supply chain.

JIANLIN JIANG is currently a Lecturer with the Faculty of Architecture, Civil Engineering and Environment, Ningbo University, China. His research interest includes construction supply chain.