IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 17, 2019, accepted December 16, 2019, date of publication December 20, 2019,
date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2961183

A Novel Cryptocurrency Wallet Management Scheme Based on Decentralized Multi-Constrained Derangement

XIAOJIAN HE, JINFU LIN, KANGZI LI, AND XIMENG CHEN
School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
Corresponding author: Kangzi Li (likangzi@foxmail.com)

ABSTRACT Secure and stable cryptocurrency key management is important in modern cryptocurrency because the keys are the only way to access digital assets. Although many cryptocurrency wallet schemes have been proposed, some application limitations and inherent security risks still exist. In this paper, we propose a novel cryptocurrency wallet management scheme based on Decentralized Multi-Constrained Derangement (DMCD) to store the keys securely and stably in a decentralized network. Serving as the data distribution strategy, DMCD has high data dispersion and a better balance between the rate of storage space utilization and contribution, which can guarantee the security and stability of the key storage and recovery. In our scheme, to cope with the problem that nodes are frequently online and offline in the decentralized network, we employ a Shamir-Kademlia-Neighbor (SKN) redundancy strategy to ensure the high availability of stored key. Meanwhile, for achieving anonymous communication during DMCD data distribution, based on the Kademlia protocol, we change the Client/Server (C/S) mode of Hordes protocol to a decentralized version. All the proposed technologies can ensure that our scheme works well in a decentralized mode. The experiments and evaluations demonstrate that our scheme is efficient, stable, and secure in the decentralized network.

INDEX TERMS Wallet management protocols, cryptographic protocols, decentralized multi-constrained derangement, flow network, decentralized anonymous communication.

I. INTRODUCTION

Cryptocurrency is a transactional medium, and it is based on cryptography to ensure the security of transactions. Based on the decentralized consensus mechanism, bitcoin was proposed by Nakamoto [1] and became the first cryptocurrency in 2009. Currently, represented by bitcoin, ethereum, and EOS.IO, the market value of the decentralized cryptocurrency is expected to be as high as 1 to 2 trillion dollars [2]. In a decentralized cryptocurrency system, asset ownership and security are verified by the public key cryptosystem. The public key is used to represent the asset address while the private key is used to verify the ownership of assets uniquely. A cryptocurrency wallet is a software or tool mainly used to help users manage the public key and private key. Many studies [3]–[5] proved that cryptocurrency key management schemes based on modern public-key cryptography are too complicated, which brings great learning costs, security risks, and even assets loss [6], [7] to the public. Thus, managing the cryptocurrency keys more securely and efficiently has become the research focus on blockchain technology.

Digital currency developers have proposed different cryptocurrency wallet management schemes to manage the cryptocurrency keys, such as keys in local storage, password-protected (encrypted) wallets, offline storage of keys, air-gapped key storage, password-derived keys, hosted wallets, etc. However, these schemes also have some disadvantages. The local storage wallet scheme requires users periodically back up the keys, which will cause high maintenance costs and lower ease of use. What's worse, the keys stored locally are easily stolen by malware [8]. The offline storage scheme isolates the storage device from the network to ensure the wallet will not be attacked through the network. A typical application is the paper wallet which turns a private key into QR code and prints it in the paper, making it more convenient to back up but easier to be revealed. The hosted wallets scheme refers to hosting the key to the server and it is managed with the help of the server, which makes it more

The associate editor coordinating the review of this manuscript and approving it for publication was Alessandra De Benedictis.

convenient and efficient for users [9]. However, this solution is troubled by poor data distribution and risk aggregation because the central server stores a large amount of cryptocurrency key data. Up to now, more than 980,000 bitcoins have been stolen due to attacks on trading platforms and third-party hosting services [10]–[12].

Most of the local storage schemes focus on how to make it difficult for attackers to steal and crack the cryptocurrency key. But they are troubled by self-management problems and poor ease of use. Conversely, the central hosting scheme is convenient for users to manage the keys, but it is easy to suffer from the risk aggregation. For weakening the role of the central node and reducing the risk aggregation, some semi-decentralized and decentralized schemes [13], [14] are proposed, but there are problems such as unbalanced data storage and overall instability. Compared with local management and central management, decentralized management has more advantages. Therefore, it is a critical problem to design an efficient, secure, and stable management to store the cryptocurrency key in the decentralized network. In this paper, we propose a Decentralized Multi-Constrained Derangement (DMCD) based cryptocurrency wallet management scheme, and our main contributions are as follows:

1. We propose a DMCD based cryptocurrency wallet management scheme to store keys in a decentralized network. Serving as a data distribution strategy, DMCD can balance the rate of space utilization and contribution, which guarantees the stability of the wallet. Meanwhile, with high data dispersion, DMCD ensures the security and recoverability of the storage key when attacked.

2. This paper uses the flow network for the first time to flexibly solve the Multi-Constrained Derangement (MCD).

3. Combining with Kademlia [15] routing protocol, we change the C/S mode of the Hordes [16] to a decentralized mode, which is denoted as D-Hordes. The D-Hordes serves as the decentralized anonymous communication protocol in DMCD to guarantee the security of data distribution.

4. We present Shamir-Kademlia-Neighbor (SKN) redundancy strategy to increase the availability and stability of the proposed scheme.

5. We conduct performance experiments and prove that our scheme performs better in data dispersion, availability, and stability compared with the traditional methods. Furthermore, we also evaluate various aspects of the proposed scheme in a widely accepted key management evaluation framework.

II. RELATED WORK

A lot of work has been done to design a more efficient and secure cryptocurrency key management scheme. To more comprehensively and reasonably evaluate the feasibility of the current schemes, Eskandari *et al*. [9] proposed a feasibility assessment framework for the wallet management scheme and compared six mainstream digital wallet schemes. Tschorsch *et al*. [17] evaluated the structural principles and security risks of digital wallets in the cryptocurrency framework and pointed out that the management of digital wallets was the relatively fragile link in the cryptocurrency system. Based on Ethereum's smart contract, Homoliak *et al*. [18] designed an offline key wallet framework and used a two-stage verification strategy to identity authentication. Goldfeder *et al*. [19] presented a digital wallet signature scheme based on the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) to achieve multi-user co-management. Bosamia *et al*. [20] analyzed the advantages of Merkle tree technology in securing digital wallets and proposed a digital wallet management scheme based on it. Nolan *et al*. [21] proposed a distributed shared digital wallet based on blockchain, which used the blockchain publishing mechanism to guaranteed high security.

To make backup key more difficult to crack, Zheng *et al*. [13] propounded a digital key backup mechanism based on ECDSA. Their scheme utilized the inferable defect of the ECDSA algorithm and combined the key variant with the blockchain transaction record to infer the original private key. Hosam [22] used steganalysis fractal techniques to hide key data in fractal tree graphics and made it more difficult for attackers to distinguish the implicit information. He *et al*. [14] put forward an encryption wallet scheme based on a semi-trusted social network. Their scheme built a flexible authorization wallet backup strategy based on secret sharing and symmetrical encryption to ensure security. Abe [23] proposed a digital wallet security management system that ensured the legitimacy and security of the backup process through authentication and access control. Dai *et al*. [30] designed a lightweight bitcoin wallet based on Trustzone to guarantee the security of simplified payment verification (SPV), private key, and the address of the wallet.

Compared with the previous schemes, the idea behind this paper is to make the backup data of the cryptocurrency key more dispersed to achieve security and stability. Our scheme is based on two aspects, one is the decentralized network, which can avoid the restriction and risk aggregation caused by center servers. And the other is the derangement-based data distribution strategy, which can achieve a high rate of dispersion and the balance of the entire network. Through these two methods, the DMCD can reduce the risk of digital assets being attacked. Meanwhile, with high dispersion, DMCD can guarantee the security and stability of our scheme during key recovery when some nodes are attacked.

III. DMCD BASED CRYPTOCURRENCY KEY MANAGEMENT SCHEME

In this section, we introduce our cryptocurrency wallet management scheme. The process is shown in Fig. 1. In a peer-to-peer network, within a certain time slice, nodes use the subset as a unit to back up in parallel. Using the MCD rules and solving method described in section III A, each node in the subset gets a backup mapping path table. Then the backup node subset performs decentralized negotiation according to steps described in section III B to achieve DMCD. Meanwhile, each node will perform a redundant backup of the data

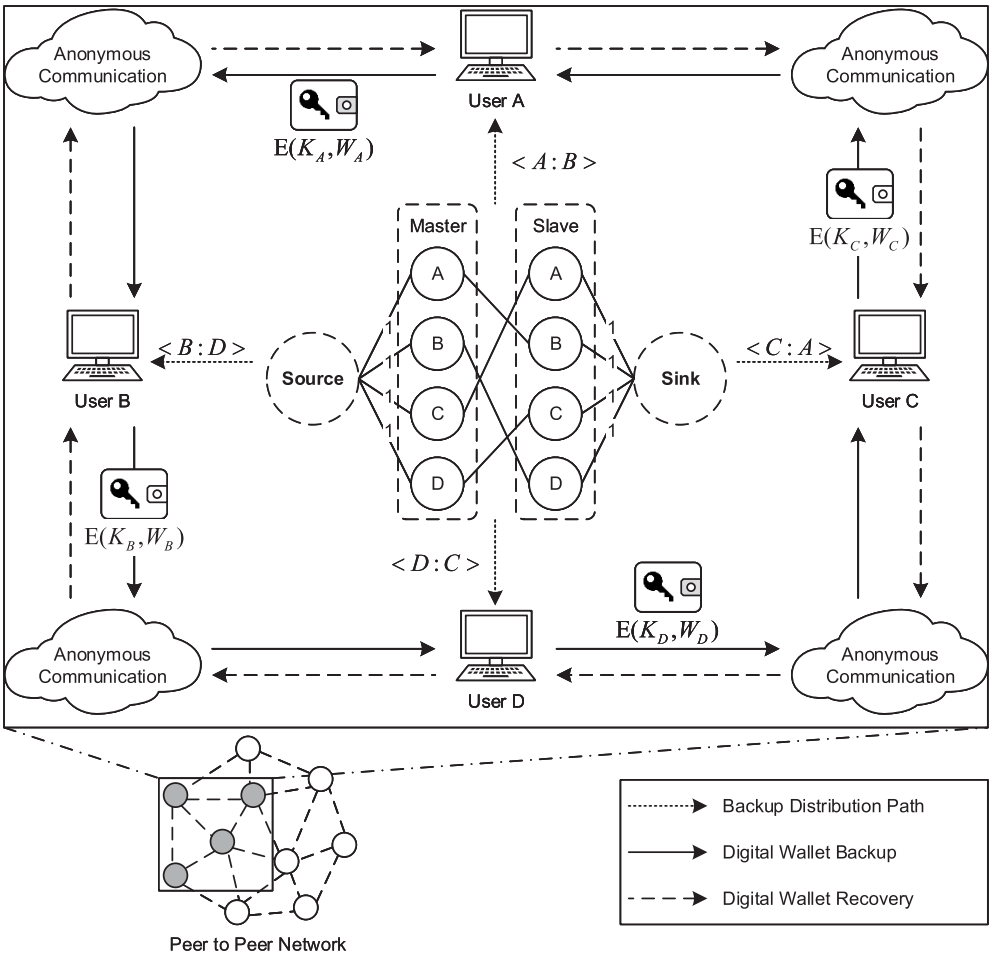


FIGURE 1. An overview of our proposed cryptocurrency wallet management scheme.

of keys and mapping paths with the strategy introduced in section III C. The anonymous communication strategy used in the data distribution and negotiation process is described in section III D.

A. FLOW NETWORK TO SOLVE THE DMCD

Definition 1 (Derangement): $\varphi : \{n_1, n_2, n_3, \ldots, n_n\} \rightarrow \{n_1, n_2, n_3, \ldots, n_n\}$, where φ is an arrangement operation and $\varphi(i)$ represents the value at *ith* position after the arrangement φ. If $\forall \quad 0 \leq i \leq n$, there is $\varphi(i) \neq n_i$, then φ is derangement. The set of the derangement with n elements is DR_n, $DR_n = \{\varphi : \{n_1, n_2, n_3, \ldots, n_n\} \rightarrow \{n_1, n_2, n_3, \ldots, n_n\} | \varphi(i) \neq n_i, \forall 0 \leq i \leq n\}$. The total number of elements in DR_n is $D_n = \left\lfloor \frac{n!}{e} + 0.5 \right\rfloor$.

The mapping rules of derangement ensure each node is mapped and mapped only once, which guarantees every node is utilized during the mapping. When serving as a data distribution strategy, derangement rules can ensure the fairness of the consumption and contribution of storage space. Meanwhile, according to the number of results of derangement, when $n = 20$, $D_n = 8.950146 \times 10^{17}$. This means derangement still possesses high randomness and can prevent the mapping results from being brute force cracked to some extent if n is relatively large. Furthermore, in many practical application scenarios, each backup node may have many additional constraints. For example, a node does not store data to nodes previously had a trading relationship with it. We can easily add new mapping rules or new constraints to derangement to form multi-constrained derangement (MCD), which can handle many real-world scenarios while retaining all the advantages of derangement. These advantages make MCD a suitable data distribution strategy in a decentralized network.

The derangement process can correspond to the data distribution process between the source nodes and the destination nodes: n_i represents the source node while $\varphi(i)$ represents the destination node. And there exists only one constraint of each node in the original derangement according to its definition, which corresponds to Single Constraint Derangement (SCD). In the actual situation, each node will have more than one requirement and constraint, which is denoted as MCD. In a decentralized network, backup paths are formed between the source nodes requesting the backup and the destination nodes storing the backup data. The capacity of the backup path can correspond to the constraint conditions between the nodes.

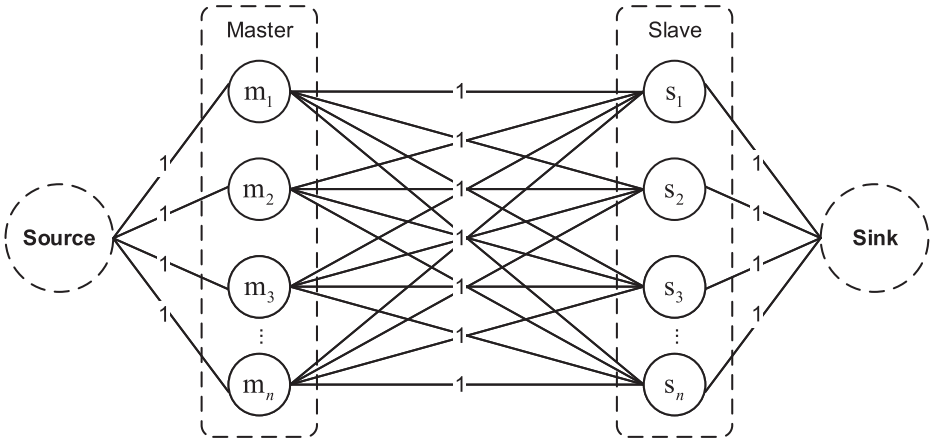


FIGURE 2. the flow network to solve the derangement.

Thus, we propose to model the backup process through flow network. We denote the source node as the master node and the storage node as the slave node. We use a key-value pair $< m_i, s_j >$ to represent the mapping path between a master node and its slave node. As shown in Fig. 2: *Source* is the source of the flow network and it is a virtual node. *Master* set includes all the master nodes; *Slave* set includes all the slave nodes. *Sink* is the sink of the flow network and it is a virtual node. Except for *Source* and *Sink*, which are serving as auxiliary nodes, each node in the flow network model acts as both the master node and the slave node. We describe this setting as $m_1 = s_1, m_2 = s_2, \ldots, m_n = s_n$. The connecting line represents the mapping path and the number above the line represents the traffic capacity. We use the binarized capacity value to represent whether the path is passable, and the capacity value of 1 represents the flow path is reachable while 0 represents the path is unreachable.

After completing the modeling, we add the following constraints to make the model achieve MCD rules:

(1) The path capacity value of the *Source* node to all master nodes is set to 1, so does the *Sink* node to all the slave nodes.
(2) There is one and only one path with a capacity value 1 between each master node and the slave node. The mapping path table of node i is denoted as $T_i = \{< m_1, s_{j_1} >, < m_2, s_{j_2} >, \ldots, < m_n, s_{j_n} >\}$, where $j_i \in \{1, 2, 3, \ldots, n\}$.
(3) Assuming that C_i is the constraint node set of m_i, there is $\forall c \in C_i, \nexists < m_i, c > \in T_i$.
(4) Backup data cannot be stored in its owner. It means that $\forall < m_i, s_j > \in T_i, m_i \neq s_j$.
(5) Backup data cannot be stored in the key wallet of users who involved in transactions with the source node previous. For example, when s_i and m_i have a trading relationship before, then we have $s_i \in C_i$.

The condition (1), (2), (3), and (4) are the sufficient and necessary condition of derangement. And the constraint condition (5) can guarantee multiple backup data of the same source node will not be stored in the same key wallet. When the model satisfies all the constraints above while the flow of the *Sink* is the largest, the output satisfies MCD. In this paper, we improve Edmonds-Karp [27] algorithm to solve the maximum network flow, and the solution process is shown in Algorithm 1.

B. IMPLEMENTATION OF DMCD

The most critical problem of the scheme in this paper is to complete MCD in a decentralized network, which is DMCD. Assuming that each node must take part in the mapping path generation to contribute its computing capacity. At the same time, the master node can select the slave node autonomously to prevent the backup path from being inferred. Based on the above premises and characteristics of derangement, the MCD can be improved to DMCD, and the specific implementation process is as follows:

1) The whole negotiation process uses the anonymous communication protocol D-Hordes described in section III D.
2) Each m_i in $M = \{m_1, m_2, \ldots, m_n\}$ runs the MCD algorithm to generate a mapping path table $T_i = \{< m_1, s_{j_1} >, < m_2, s_{j_2} >, \ldots, < m_n, s_{j_n} >\}$ and saves its own mapping path $< m_i, s_{j_i} >$.
3) m_i distributes the remaining items to the corresponding nodes. Using the public key PK_k of m_k to encrypt $< m_k, s_{j_k} >$ and timestamp $Time_k$, m_i sends $E(PK_k, [< m_k, s_{j_k} > ||Time_k])$ to m_k through instruction *SEND_MAP*.
4) m_i receives (n-1) *SEND_MAP* and attach item $< m_i, s_{j_i} >$ to form a backup path candidate table $D_i = \{< m_i, s_{j_1} >, < m_i, s_{j_2} >, \ldots, < m_i, s_{j_k} >\}$.
5) m_i selects the lowest frequency item $< m_i, s_k >$ from D_i and then sends $[timestamp_i || optional\_num_i]$ to s_k through instruction *LOCK_MAP*, where $optional\_num_i$ is the total item number in backup path candidate table of node i.

Algorithm 1 Maximum Network Flow to Solve MCD

Input: Node Set Scale (*numuber*), Constrain Condition (*limit*)
Output: Maximum Flow network Solution (*maxflowgraph*)
1. **function** MCD Solution(*numuber*, *limit*)
2. *residual* ← ModelInitalization(*numuber*, *limit*)
3. *sink* ← *number* × 2 + 1, *source* ← 0
4. **while** TRUE **do**
5. *queue* ← *source*, *pre*[*source*] ← −1, *flow*[*source*] ← ∞
6. **while** *queue* **and** *sink* ≠ *index* ← *queue*.*Pop*(0) **do**
7. **for** *i* ← 0 **to** *sink* **do**
8. **if** *i* ≠ *source* **and** *residual*[*index*][*i*] > 0 **and** *pre*[*i*] = −1 **then**
9. *pre*[*i*] ← *index*
10. *flow*[*i*] ← *min*(*flow*[*index*], *residual*[*index*][*i*])
11. *queue*.*Append*(*i*)
12. **end for**
13. *random*.*Shuffle*(*queue*)
14. **end while**
15. **if** *augmentflow* ← *flow*[*sink*] ≠ 0 **then**
16. *k* ← *sink*
17. **while** *k* ≠ *source* **do**
18. *prev* ← *pre*[*k*]
19. *maxflowgraph*[*prev*][*k*] ← *maxflowgraph*[*prev*][*k*] + *augmentflow*
20. *residual*[*prev*][*k*] ← *residual*[*prev*][*k*] − *augmentflow*
21. *residual*[*k*][*prev*] ← *residual*[*k*][*prev*] + *augmentflow*
22. *k* ← *prev*
23. **end while**
24. **else break**
25. **end while**
26. **return maxflowgraph**
27. **end function**

6) If m_i receives more than one *LOCK_MAP* instruction, m_i preferentially accepts request with small *optional_num*. If *optional_num* is the same, m_i accepts request with the early *timestamp*. Then m_i sends an acknowledgment message to the node whose request is accepted.
7) When receiving an acknowledgment message, m_i adds item $< m_i, s_k >$ to backup routing table $BRT_i = < m_i, s_k >$ and maintains the table. Nodes that do not receive the acknowledgment message, return to step 4).

Fig. 3 shows a simple example of fulfilling DMCD. Firstly, each node runs the MCD algorithm to generate a mapping path table. Secondly, the node distributes the items of the mapping path table to the corresponding nodes. Thirdly, each node picks up the lowest frequency item from its backup path candidate table as the candidate slave node. Finally, each node takes part in the negotiation to lock their slave node.

The sign of fulfilling the negotiation is that each master node successfully locks a slave node. Assume the network constraint condition set is $C = \{C_1, C_2, \ldots, C_n\}$, where C_i is the constraint condition of node i. The network' s output mapping path set is $S = \{S_1, S_2, \ldots, S_n\}$ and S_i is the mapping path item of node i.

Definition 2: if $\forall i \in 1, 2, 3, \ldots, n$, there is S_i meeting the conditions C, which is denoted as $S_i \models C$, then we define S as one of the outputs of the flow network and the result is denoted as $C \mapsto S$.

Based on the above definition, we prove that the negotiation result is a feasible solution to the MCD:

1) Suppose that the flow network gets n outputs and one of the outputs is S^k. Then the result can be denoted as : $\forall k \in K$, there is $C \mapsto S^k$, where $K = \{k | 1 \le k \le n\}$. Therefore, $\forall i \in 1, 2, \ldots, n$, there is $S_i^k \models C$.
2) Assume that after the negotiation is completed, all the mapping path set is $B = \left\{S_1^{k_1}, S_2^{k_2}, \ldots, S_n^{k_n}\right\}$, where $i \in 1, 2, \ldots, n, k_i \in K$.
3) Since $\forall i \in 1, 2, \ldots, n, S_i^{k_i} \in B$, and according to (1), there is $S_i^{k_i} \models C$, so there is $C \mapsto B$.

C. REDUNDANCY STRATEGY

In a decentralized network, nodes are frequently online and offline, which has a great impact on the keys backup and recovery. A stable and efficient redundant backup strategy is the key to ensure the availability of wallets. To deal with this problem, combining the Secret sharing [24] scheme

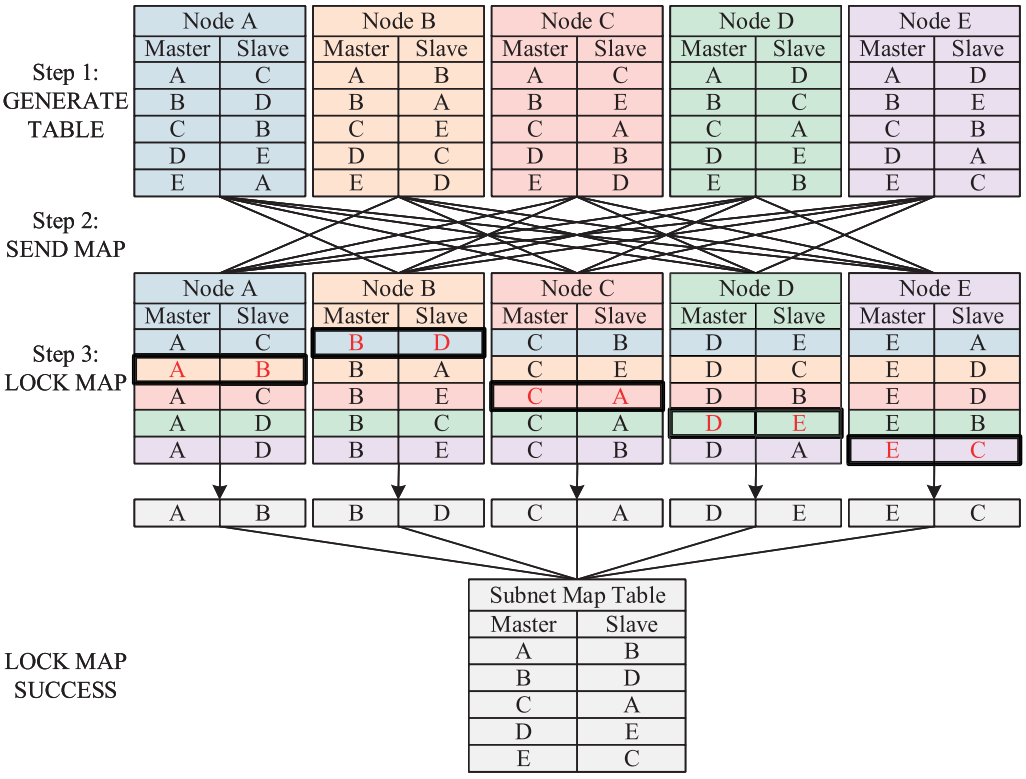


FIGURE 3. An implementation example of DMCD.

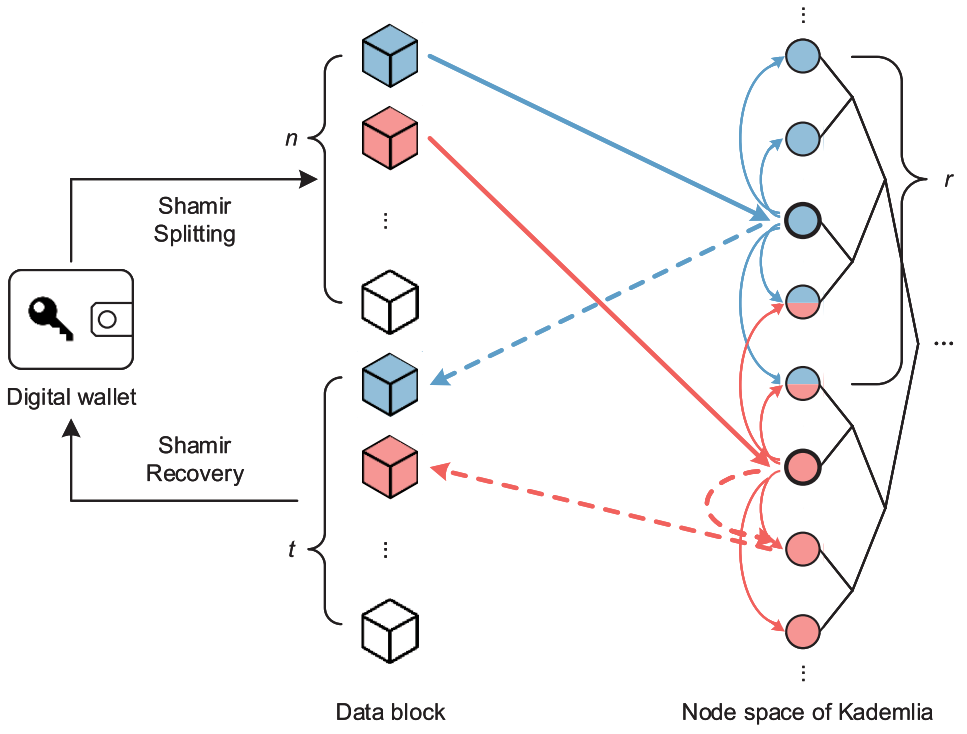


FIGURE 4. SNK redundancy strategy.

with the Kademlia XOR distance metric [15], we propose a Shamir-Kademlia-Neighbor (SKN) redundancy strategy. And the process of SKN is shown as Fig. 4.

Secret sharing [24] scheme privates a way to divide a secret into n parts and it can rebuild the secret through the reconstruction function R if any t or more parts are retrieved.

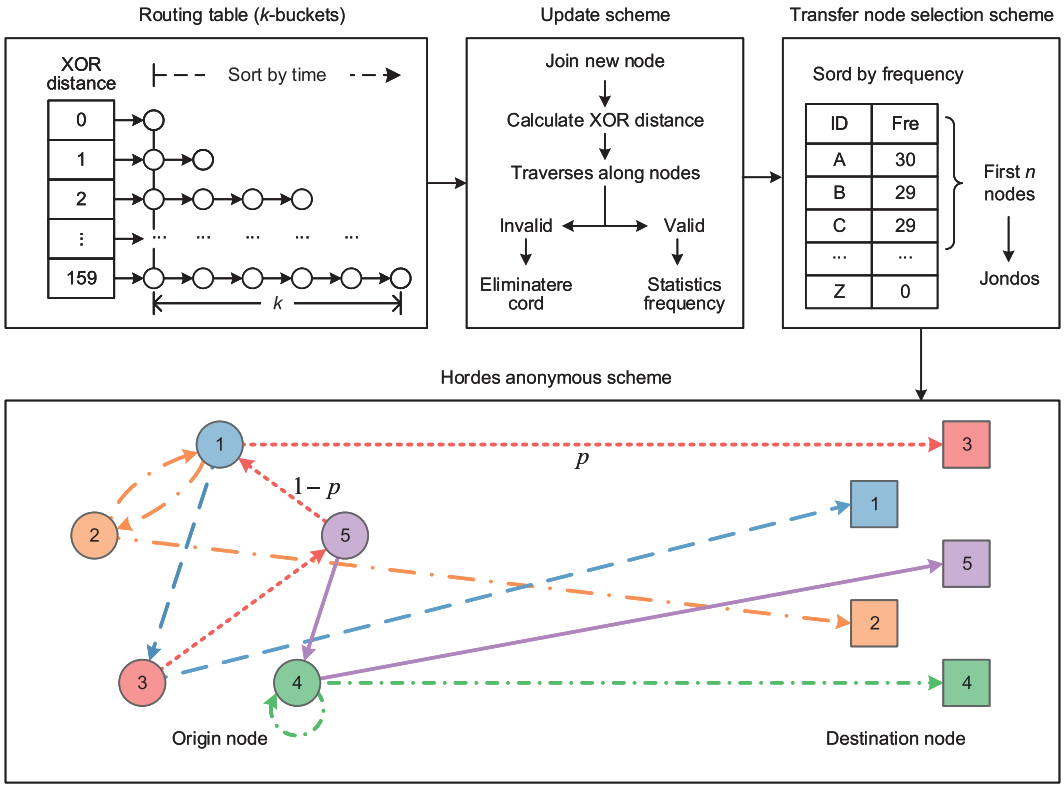


FIGURE 5. An overview of the D-Hordes anonymous communication protocol.

We definite the function $shamir(t, n, secret, R)$ to represent the secret sharing scheme and $R(w_1, w_2, \ldots, w_t) \Rightarrow W_U$ to restore the secret.

When a node joins the network, it obtains a fixed route ID through the *identity authentication module*. Then it finds its r neighbors through the Kademlia XOR metric [15]. The source node will back up its encrypted data to r neighbor nodes of the target node. In this way, when the target node is offline, the source node can find the neighbors of its target node to request the data. Kurihara *et al*. [29] proposed a more suitable Secret Sharing Scheme and studied the influence of parameter (t, n) on the performance of the scheme. Base on their work, in our scheme, the data block parameter n is set as 11, the reconstruction parameter t is set as 6. Considering that the number of data blocks is already enough, the redundancy parameter r is set as 2.

D. DECENTRALIZED HORDES

To prevent the source address of the backup data from being inferred, an anonymous communication protocol is needed during the data distribution. The anonymous communication system is mainly including C/S architecture, P2P with a central and non-central P2P architecture. Bitmessage [25] is an anonymous communication protocol based on the decentralized network. Its anonymous communication is based on the blockchain workload proof mechanism. Structural-based tunneling [28] is used to provide recipient anonymity for the circular structured P2P network. Hordes protocol has the lowest latency, which relies on the directory server to coordinate the jump agent node resources. Considering that we utilize the Kademlia protocol to organize the network in the redundancy strategy, we propose to change the C/S mode of the original Hordes protocol into the decentralized mode based Kademlia protocol. In this paper, we denote our new anonymous communication as D-Hordes.

The Kademlia routing protocol records the network nodes based on the K-bucket routing table. And the K-bucket routing table sorts the network nodes according to their survival time. According to the K-bucket update mechanism, the highly available nodes are more easily stored in the routing table, so they can serve as a stable anonymous agent. Based on the K-bucket update mechanism, we design a routing jump strategy and then change the centralized architecture service mode of the Hordes to a decentralized mode: Hordes(D-Hordes). As Fig. 5 shown, the process of D-Hordes protocol is as follows:

1) When U joins the network, it runs the *identify registration module* described in section IV and get its routing *ID*. Then U initializes the K-bucket routing table based on the XOR distance and records each surviving node in K-bucket table when update.
2) When the K-bucket table is stable, U randomly selects the node with the highest frequency of survival from

the K-bucket routing table as the transit node set $J_U = \{j_1, j_2, \ldots, j_k\}$. U periodically updates the K-bucket table and J_U.

3) For each j_i in J_U, U generates a communication key SK_i, encrypts it with the public key of j_i, and sends the encrypted communication key to j_i.
4) U selects a multicast group address M and a proxy node j_i. U encrypts both the data and M, then sends $E(SK_i, [data||M])$ to j_i.
5) When j_i receives the message, it will send the data to the destination node with the probability of p or send the data to another proxy with the probability of $(1-p)$.
6) After receiving the data, the destination node sends the acknowledgment packet through the multicast address M. U listens to the multicast address M to receive the acknowledgment data packet.

IV. BACKUP AND RESTORE PROCESS

A. IDENTITY REGISTRATION MODULE

The identity registration module needs to generate stable identity information for users. The necessary identity information includes private key AK_U, public key PK_U, identity authentication information C_U, and unique route ID. Firstly, we utilize the existing biometric key generation technique [26] to generate an initial inherent key. Then, using the initial inherent key to further generate the necessary identity information. The process of identity registration module is as follows:

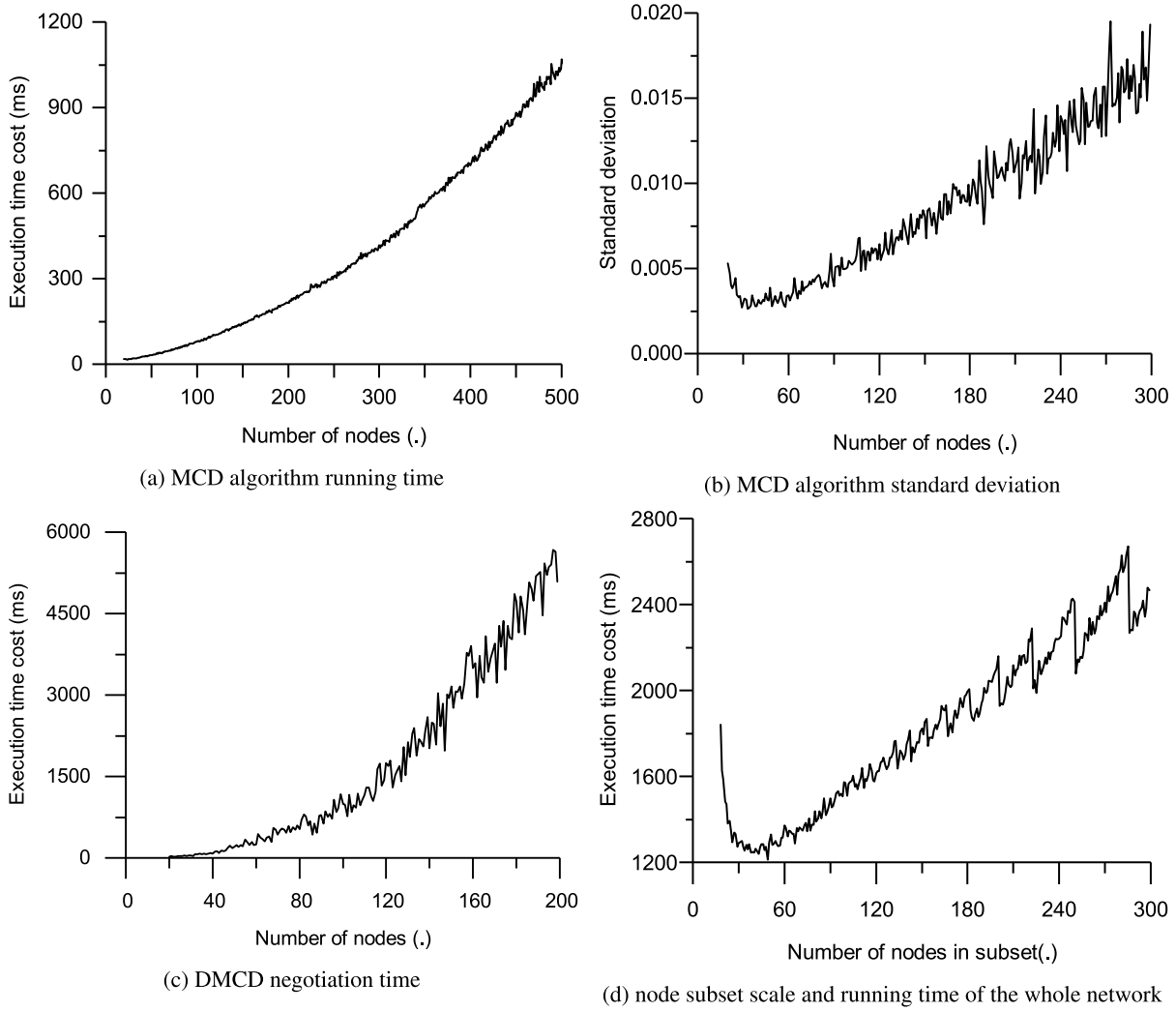
1) Using the fingerprint or voiceprint to generate initial inherent key through the biometric key generation technique [26]:
$Inherent(U) \Rightarrow IK_U$.
2) Hashing IK_U with the user's password information K_U or device encoding information O_U to generate an identity authentication private key AK_U:
$SHA256(IK_U||K_U||O_U) \Rightarrow AK_U$.
3) Using the $SECP256$ elliptic curve to generate the public key :
$SECP256(AK_U) \Rightarrow PK_U$
4) Generating the routing ID in decentralized network:
$ID = RIPEMD160(SHA256(publicKey))$.
5) Obtaining the authentication information by calculating the hash value of the authentication private key:
$SHA256(AK_U) \Rightarrow C_U$.

B. BACKUP PROCESS

1) Using the *identity registration module* to generate the identity authentication public key PK_U, identity authentication private key AK_U, authentication information C_U and routing ID.
2) Encryption key split. Splitting the Encryption asset key into sub-asset key sets $\{w_1, w_2, w_3, \ldots, w_n\}$ by the $shamir(t, n, W_U, R)$ function:
$shamir(t, n, W_U, R) \Rightarrow \{w_1, w_2, w_3, \ldots, w_n\}$.
3) Back up key and mapping path:
   a) As for each w_i in $\{w_1, w_2, w_3, \ldots, w_n\}$, U uses DMCD to generate the mapping path $< U, s_i >$.
   b) U encrypts the data w_i and $< U, s_i >$ with public key PK_U:
   $e\_w_i = E(PK_U, w_i)$, $e\_path = E(PK_U, < U, s_i >)$.
   c) U encrypts the data e_w_i, the public key PK_U, and the certification information C_U with the public key of node s_i :
   $e\_e\_w_i = E(PK_{s_i}, [e\_w_i||PK_U||C_U])$.
   Then U sends $e\_e\_w_i$ to node s_i with D-Hordes protocol.
   d) U encrypts the data e_path and the public key PK_U with the public key PK_N of its neighbor nodes:
   $e\_e\_path = E(PK_N, [e\_path||PK_U])$.
   Then U sends e_e_path to its neighbor nodes with D-Hordes protocol.

C. RESTORE PROCESS

1) If the key data does not exist when users reconnect to the network, using the *identify registration module* to quickly regenerate the public key PK_U, private key AK_U, authentication information C_U, and routing ID.
2) Restore mapping path
   a) U issues a data request to its neighbor nodes to obtain the backup path data and identifies this request using its routing ID and the temporary interaction number N_1:
   $E(PK_N, [ID||N_1])$.
   b) After receiving the request, the neighbor node N encrypts a temporary interaction number N_2 with the user's public key PK_U, and attaches N_1 to show it has received the request. Then N sends the data to U:
   $E(PK_U, [N_2||N_1])$.
   c) U verifies N_1 and then the temporary interaction number N_2 encrypted with PK_N is returned to N:
   $E(PK_N, N_2)$.
   d) N determines the identity of U by verifying N_2 and sends the backup path data e_path to U.
3) Restore backup cryptocurrency key
   a) For each target node S in backup path, U randomly generates a session key SK with a temporary interaction number N_3 and sends to S: $E(PK_S, [SK||N_3])$.
   b) After S receives the request, it encrypts a temporary interaction number N_4 and N_3 with the session key SK: $E(SK, N_3||N_4)$.
   c) U authenticates N_3 to confirm identity S and returns the temporary interaction number N_4 and authentication information C_U to S.
   d) S determines that U holds the data by verifying the authentication information C_U, and sends the encrypted key data to the U using the session key SK:
   $E(SK, [e\_w_i||N_3])$.
   e) U retrieves the t backup data and uses the private key AK_U to decrypt the data: $\{w_1, w_2, ...w_t\} =$



(a) MCD algorithm running time

(b) MCD algorithm standard deviation

(c) DMCD negotiation time

(d) node subset scale and running time of the whole network

FIGURE 6. Time performance of DMCD.

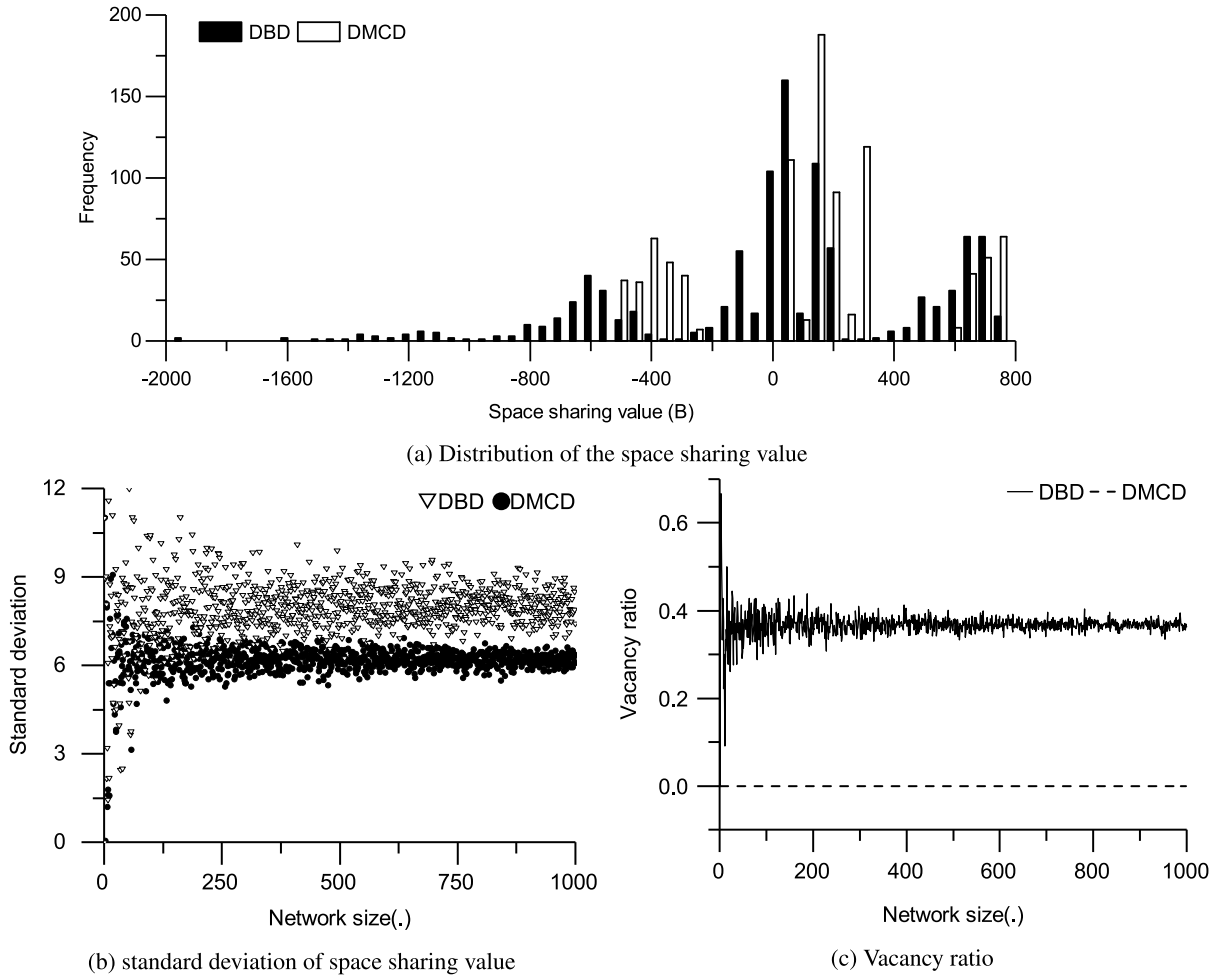
$D(AK_U, \{e\_w_1, e\_w_2, ...e\_w_t\})$. Then U uses recovery function $R(\{w_1, w_2, ...w_t\})$ to get the final backup data W_U: $R(\{w_1, w_2, \ldots, w_t\}) \Rightarrow W_U$.

V. EXPERIMENT AND EVALUATION

A. EFFICIENCY EVALUATION

This section tests the efficiency of DMCD based key distribution strategy. Since the DMCD is based on the MCD and achieved through negotiation, we test the running time of the MCD algorithm and the negotiation time to fulfill the DMCD algorithm separately. The experiment simulates the running time and stability of the MCD under different network scales. The solution for MCD is based on breadth-first and its time complexity is $O(V^2 E)$, where V is the number of vertices and E is the number of lines. As shown in Fig. 6 (a), the execution time of the algorithm increases with the node subset scale. Fig. 6 (b) shows that the standard deviation of the running time increases with the number of nodes, so the stability decreases. And as Fig. 6 (c) shown, the negotiation time increase with the nodes number due to the increasing probability of selection conflicts. When the number of nodes participating in the backup process is less than 200 each time, the negotiation time is maintained within 6s, which is acceptable.

When the number of nodes participating in backup within a time slice is too large, it will cost more negotiation time and reduce the performance of the scheme. To solve this critical problem, we design a subset organization mechanism and explore a suitable node scale to finish DMCD. Firstly, the backup nodes initiate the *BACKUP* instruction and wait for the next time slice. Then the nodes initiate a backup request with a timestamp and start listening for backup requests from other nodes. Each node that requests backup will maintain a backup request table and records all backup requests within this time slice according to the timestamp. We assume that the time slice is t, the total number of nodes that require backup is N, and the best subset scale is $[N_{min}, N_{max}]$. The steps for subset formation are as follows:



(a) Distribution of the space sharing value

(b) standard deviation of space sharing value

(c) Vacancy ratio

FIGURE 7. Spatial performance of DMCD.

1) If $N < N_{min}$, it indicates that a subset cannot be formed during this time slice. The nodes need to wait for the next time slice, initiate the *BACKUP* instruction, and request a backup again. Meanwhile, the backup request table is cleared.
2) If $N_{min} \leq N \leq N_{max}$, the nodes can format a subset to start a backup. At the same time, the backup request table is cleared.
3) If $N > N_{max}$, every time picking up N_{max} nodes from the backup request table sequentially to form a subset until the number of remain nodes less than N_{min}. And the remain nodes will come back to step 1. At the same time, the backup request table is cleared.

We decompose a larger set of backup nodes into several same subsets with this subset formation mechanism. To determine the best subset scales, we calculate the serial running time of all subsets to represent the running time of the larger one. We simulate the time of a network with 3000 nodes completing the DMCD through different scale subsets. As it is shown in Fig. 6 (d), the running time of the same network (with 3000 nodes) will increase with the subset scale. Combining with the running time of MCD and the negotiation time, we set the node subset scale $[N_{min}, N_{max}]$ as [20, 200] to guarantee the whole backup time within 6 seconds. According to the node subset scale, the time slice interval t for a backup request is set to 6s. And all the subsets run in parallel. Therefore, our scheme can perform well in terms of time efficiency.

B. SPACE STORAGE STABILITY EVALUATION

We conduct a simulation experiment of actual cryptocurrency wallet data storage to evaluate the spatial performance of the proposed scheme. Three kinds of common asset data in cryptocurrency wallet and their data size are shown in Table 1. Assume that during each data backup process, the data type is randomly obtained from Table 1. In order to quantitate the rate of storage utilization and contribution, we define a space sharing value. The initial space sharing value of a node is 0. Its space sharing value will increase when storing data of other nodes. When using the space of other nodes, its space sharing the value will decrease. The mainstream distribution strategies mainly include Distributed Hash Table (DHT) and directory-based random distribution (DBD), while DHT-based distribution

TABLE 1. Three common digital asset data sizes.

Type	Data Size
Original private key	32B
Mnemonic	133B - 163B
Keystore	592B - 724B

strategies are rarely used, so we compare our scheme with directory-based random distribution (DBD) strategies. Our experiment simulates the backup process in a network composed of 1000 nodes and records the space sharing value of each node.

According to the definition of the space sharing value, when the space sharing value is near 0, the space utilization and contribution of the node can be balanced, which is conducive to the stability of the network. Fig. 7(a) shows the distribution of space sharing value for the two distribution schemes. In the DMCD based distribution scheme, the space sharing value is concentrated in the interval [-400,800]. In the DBD scheme, the space sharing value is dispersed in the interval $[-2000, 800]$. Compared with DBD, the distribution of the space sharing value in DMCD scheme is more focused and closer to zero, which means that the DMCD is more balanced between the space utilization and contribution. Fig. 7 (b) shows that the DMCD scheme has a lower deviation than the DBD scheme. To more intuitively reflect the balance between the two schemes in terms of storage space utilization and contribution, we count the ratio of nodes whose storage space contribution is 0B during the backup process, which is denoted as vacancy ratio. When the vacancy ratio is large, it means that many nodes only consume storage space and do not providing corresponding storage space, which is not conducive to the stability and balance of the scheme. The simulation results are shown in the Fig. 7 (c). It can be seen that the node vacancy ratio of the DBD strategy ranges from 35% to 40%. As the network scale increases, the vacancy ratio gradually stable at 38%, which means that most nodes only consume storage space of other nodes without providing corresponding storage space. The vacancy ratio of DMCD is 0% because the DMCD strategy can make full use of the storage space of each node. Thus, we conclude that our scheme is stable and balanced in spatial storage.

C. AVAILABILITY EVALUATION

The SKN redundancy strategy proposed in this paper can guarantee the data persistence and availability when the slave nodes are offline. In this section, we compare with the Erasure Code (EC) on data availability. Assume that the available probability of node is $p = 0.5$, the data redundancy is $r = 2$, the threshold $t = \frac{n}{r}$ where n is the number of the data block, and the size of the K-bucket is 2. As for EC, its availability is:

$$P = \sum_{i=t}^{n} C_n^i p^i (1-p)^{n-i} \tag{1}$$

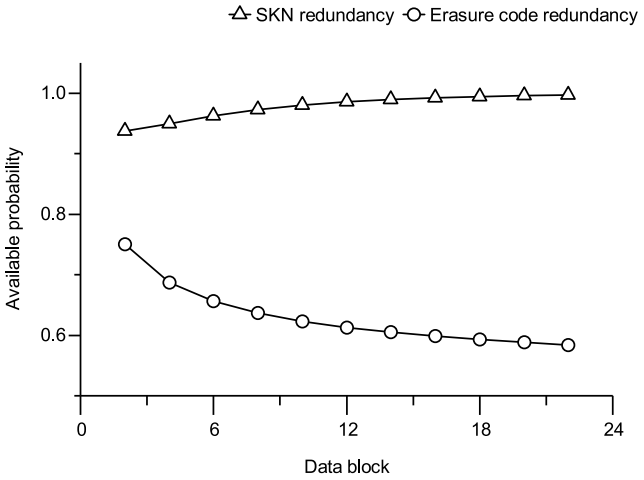


FIGURE 8. Availability of SKN and EC redundancy strategy.

And for SKN, its data availability is:

$$P = \sum_{i=t}^{n} C_n^i (1-(1-p)^t)^i (1-(1-(1-p)^t))^{n-i} \tag{2}$$

As Fig. 8 shows, the SKN redundancy has higher availability than the EC redundancy. What's more, the data availability of SKN is over 90 percent, which can guarantee the availability of our scheme.

D. FEASIBILITY ANALYSIS BASED ON KEY MANAGEMENT FRAMEWORK

This section uses Eskandari's Bitcoin Wallet Management Method Evaluation Framework [9] to analyze the availability and security of our scheme. This evaluation framework has ten dimensions, focusing on two aspects. The first aspect is the security assessment. It focuses on the attack methods that will be suffered in real networks, such as malware, physical theft, physical observation, etc. The ease of use of the digital wallet solution mainly focuses on whether the solution is user-friendly, and whether the novice user can conveniently perform key management.

In the evaluation, we compare our scheme with seven mainstream schemes. The seven wallet schemes are key mixing scheme (SNBCwallet [14]), local storage scheme (Bitcoincor), and password-protected scheme (MultiBit), offline storage scheme (Bitaddress), key equipment scheme (Brainwallet), managed wallet scheme (Blockchain) and cash scheme. The evaluation results are shown in Table 2. The black solid point indicates the system can satisfy this attribute. The black hollow point indicates the system can partially satisfy the corresponding attribute. No mark indicates that the system cannot satisfy this attribute.

The experimental results show that the proposed scheme outperforms the SNBCwallet and Blockchain schemes in decentralization performance. It is superior to MultiBit and BrainWallet schemes in resistant to password loss.

TABLE 2. Comparison of our scheme of and the mainstream scheme.

Example	Malware resistant	Key kept offline	None trusted third party	Resistant to physical theft	Resistant to physical observation	Resistant to password loss	Resistant to key churn	Immediate access to funds	No new user software	Cross-device portability
Ours	○	○	●	●	●	●	●	●		●
SNBCwallet	○	○		●	●	●	●	●		●
Bitcoincore			●		●	●	●	●		
MultiBit		○	●	○	●		●	●		
Bitaddress	○	●	●		●	●	●			
BrainWallet		●	●	○			●	●	●	●
Blockchain		○	○			●	●	●	●	●
cash	●	●	●		●	●	●	●	●	●

And it is superior to Bitcoincore and Bitaddress schemes in cross-device portability.

E. SECURITY EVALUATION

1) SECURITY OF KEY STORAGE

The data dispersion is used to measure the distribution of data storage across the network. High dispersion can make the entire network more stable when being attacked, which can guarantee the stability and security of the key storage. To evaluate the data storage dispersion of our scheme, we analyze the recoverability of backup data when nodes in the network are attacked. We define a data storage matrix S:

$$S = \begin{pmatrix} node_{j_1} & node_{j_2} & \dots & node_{j_m} \\ \dots & \dots & \dots & \dots \\ node_{k_1} & node_{k_2} & \dots & node_{k_m} \end{pmatrix} \tag{3}$$

Each row corresponds to all the backup data storage addresses of a source node. Defining recoverability Rt:

$$Rt = \frac{\sum_{i=1}^{N} D_i}{N}, \tag{4}$$

where D_i represents whether the source node can restore backup data. $D_i = 1$ indicates that the backup data can be restored while $D_i = 0$ indicates cannot be restored. N is the number of nodes. For each element in S, we use $s_{ij} = 1$ to represent that the node is destroyed and $s_{ij} = 0$ to represents the node is not destroyed. Combining with the secret sharing scheme (t,m), if $\sum_j s_{ij} > m - t$, then set $D_i = 0$, otherwise, set $D_i = 1$. The parameters are the same in both DBD and DMCD, and they are set as follows: $N = 100$, $(t, m) = (6, 11)$, and the final result is as Fig. 9. The experiment result shows the DMCD scheme has better performance in terms of recoverability than the DBD. When the network is attacked, the key distributed by the DMCD mode is more recoverable than the key distributed by the DBD mode, which ensures that the key is more secure.

2) SECURITY OF KEY RECOVERY

We analyze fake attacks and replay attacks during the key recovery process. The fake attack includes four scenarios

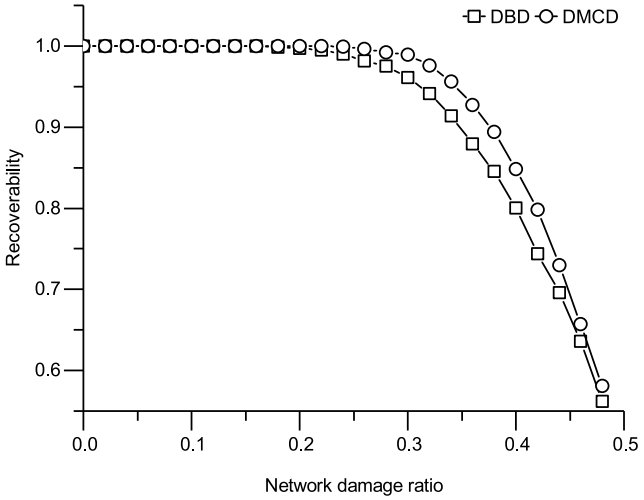


FIGURE 9. Data recoverability performance of DBD and DMCD scheme.

while the replay attack includes 2 scenarios. Suppose A is a malicious node, U is a legal user, S is a backup data storage node, and N is a U neighbor node.

a: FAKE ATTACKS

a) In the case shown in Fig. 10 (a), A sends its public key and the routing ID of U, and the neighbor node can verify that the public key does not correspond to U, so it refuses the request.
b) In the case shown in Fig. 10 (b), A gets the public key and ID of U. The neighbor node verifies the data, and returns a random number N_2 encrypted with the public key of legal user to identity its verification. A does not have the correct private key and therefore cannot return valid confirmation information.
c) In the case shown in Fig. 10 (c), A attaches the instant-generated communication key SK, and after passing identity verification, A sends the certificate C_A. But the certificate C_A is different from C_U.
d) In the case shown in Fig. 10 (d), assuming that A obtains the key data by a certain method, but A cannot reconstruct the whole key with only one data block.

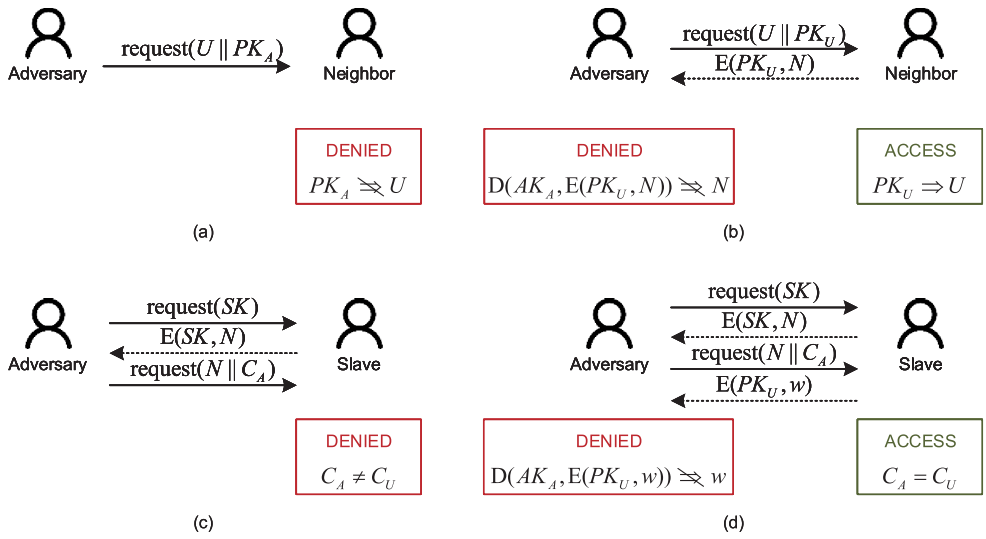


FIGURE 10. Fake attack scenarios.

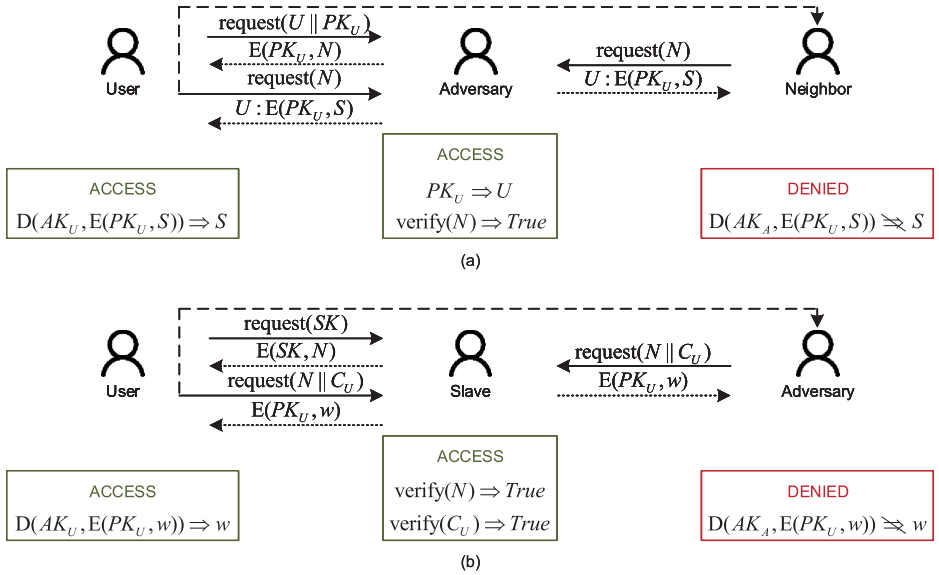


FIGURE 11. Replay attack.

b: REPLAY ATTACK

a) In the case shown in Fig. 11 (a), A steals the verification credential of U, and passes the verification of the neighbor node. But A cannot get the authentication private key of U, and cannot decrypt the backup path data S.

b) In the case shown in Fig. 11 (b), A steals the verification credential of U and successfully pass the verification of the slave node. Since A cannot know the authentication private key of U, and cannot decrypt the asset data w.

The simulation experiment indicates that our scheme can guarantee high data recoverability when the storage key is attacked. At the same time, it can resist common attacks during the key recovery process. Thus, we claim that our solution satisfies security.

VI. CONCLUSION

As the cryptocurrency market continues to expand, secure and stable key management is becoming more and more important. This article focuses on designing a decentralized cryptocurrency key management scheme. Compared with local management and central management, decentralized management can avoid risk aggregation and make use of the whole network storage resources. In this paper, we propose a scheme based on DCMD to manage the key efficiently, securely, and stably, herein using the flow network to solve the MCD problem. Meanwhile, we propose the SKN redundancy strategy based on the Kademlia XOR metric and a decentralized anonymous communication protocol D-Hordes based on Kademlia protocol to ensure the scheme can work

well in the decentralized network. Our research demonstrates that when DMCD serves as a data distribution strategy, its high data dispersion can guarantee the security and stability of the scheme. The final experiments and evaluations prove that the proposed scheme is safe, stable and effective.

REFERENCES

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.
[2] J. Mazer. (2018). Demystifying Cryptocurrencies, Blockchain, and ICOs. Toptal Website. [Online]. Available: https://www.toptal.com/finance/market-research-analysts/cryptocurrency-market
[3] S. L. Garfinkel, D. Margrave, J. I. Schiller, E. Nordlander, and R. C. Miller, "How to make secure email easier to use," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2005, p. 701.
[4] S. Sheng, L. Broderick, and C. Koranda, "Why Johnny still can't encrypt: Evaluating the usability of email encryption software," in *Proc. Symp. Usable Privacy Secur.*, 2006, pp. 3–4.
[5] S. L. Garfinkel and R. C. Miller, "Johnny 2: A user test of key continuity management with S/MIME and Outlook Express," in *Proc. Symp. Usable Privacy Secur.*, 2005, pp. 13–24.
[6] H. Gjermundrød and I. Dionysiou, "Recirculating lost coins in cryptocurrency systems," in *Business Information Systems Workshops* (Lecture Notes in Business Information Processing), vol. 183. 2014, pp. 229–240.
[7] S. Baum, "Cryptocurrency fraud: A look into the frontier of fraud," Tech. Rep., 2018.
[8] *Infostealer. Coinbit*, Symantec, Mountain View, CA, USA, 2011.
[9] S. Eskandari, J. Clark, D. Barrera, and E. Stobert, "A first look at the usability of bitcoin key management," 2018, *arXiv1802.04351*. [Online]. Available: https://arxiv.org/abs/1802.04351
[10] Bitcoin Forum. (2014). *List of Major Bitcoin Heists, Thefts, Hacks, Scams, and Losses*. [Online]. Available: https://bitcointalk.org/index.php?topic=576337
[11] A. Harney and S. Stecklow, "Twice burned—How Mt. Gox's bitcoin customers could lose again," Tech. Rep., 2017.
[12] Y. Nakamura, "Bitfinex comes back from $69 million bitcoin heist," Tech. Rep., 2017.
[13] Z. Zheng, C. Zhao, H. Fan, and X. Wang, "A key backup scheme based on bitcoin," Tech. Rep.
[14] S. He, "A social-network-based cryptocurrency wallet-management scheme," *IEEE Access*, vol. 6, pp. 7654–7663, 2018.
[15] P. Maymounkov and D. Maziéres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. Int. Workshop Peer-Peer Syst.*, 2002, pp. 53–65.
[16] B. N. Levine and C. Shields, "Hordes: A multicast based protocol for anonymity," *J. Comput. Secur.*, vol. 10, no. 3, pp. 213–240, Jul. 2002.
[17] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.
[18] I. Homoliak, D. Breitenbacher, A. Binder, and P. Szalachowski, "SmartOTPs: An air-gapped 2-factor authentication for smart-contract wallets," 2018, *arXiv:1812.03598*. [Online]. Available: https://arxiv.org/abs/1812.03598
[19] S. Goldfeder, "Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Cham, Switzerland: Springer, 2016, pp. 156–174.
[20] M. Bosamia and D. Patel, "Current trends and future implementation possibilities of the merkel tree," vol. 6, no. 8, 2018.
[21] M. Nolan, D. Carboni, and N. M. Smith, "Securing distributed electronic wallet shares," Google Patent, Jan. 31, 2019.
[22] O. Hosam, "Hiding bitcoins in steganographic fractals," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2018, pp. 512–519.
[23] J. O. Abe, "BitCoin, Wallet management and network security management with storage components: A model," Tech. Rep., Jun. 2018.
[24] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[25] J. Warren. (2012). *Bitmessage: A Peer-to-Peer Message Authentication and Delivery System*. Accessed: Nov. 27, 2012. [Online]. Available: https//bitmessage.org/bitmessage.pdf
[26] G. Panchal, D. Samanta, and S. Barman, "Biometric-based cryptography for digital content protection without any key storage," in *Proc. Multimedia Tools Appl.*, 2017, pp. 1–22.
[27] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.
[28] A. Naghizadeh, S. Berenjian, E. Meamari, and R. E. Atani, "Structural-based tunneling: Preserving mutual anonymity for circular P2P networks," *Int. J. Commun. Syst.*, vol. 29, no. 3, pp. 602–619, 2016.
[29] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A new (k, n)-threshold secret sharing scheme and its extension," in *Proc. Int. Conf. Inf. Secur.*, 2008, pp. 455–470.
[30] W. Dai, J. Deng, Q. Wang, C. Cui, D. Zou, and H. Jin, "SBLWT: A secure blockchain lightweight wallet based on trustzone," *IEEE Access*, vol. 6, pp. 40638–40648, 2018.

XIAOJIAN HE received the Ph.D. degree in computer software and theory from Shanghai Jiao Tong University, China. He is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, China. His research interests include distributed computing, business intelligence, machine learning, and software development.

JINFU LIN received the B.S. degree from South China Normal University, in 2018. He is currently pursuing the M.S. degree in computer technology with the South China University of Technology, Guangzhou, China. His research interests include the areas of decentralized technology application and software development.

KANGZI LI received the B.S. and M.S. degrees in computer technology from the South China University of Technology, Guangzhou, China, in 2016 and 2019, respectively. Her research interests include the areas of blockchain application and distributed key management.

XIMENG CHEN received the B.S. and M.S. degrees in computer technology from the South China University of Technology, in 2015 and 2018, respectively. His research interests include blockchain and software development.