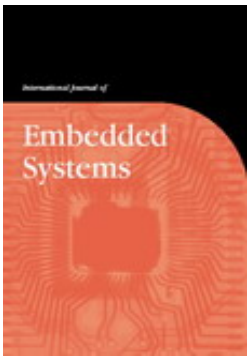


International Journal of Embedded Systems

ISSN online: 1741-1076 - ISSN print: 1741-1068
https://www.inderscience.com/ijes

The correction method of block authentication information in edge computing mode

Jianbo Xu, Wei Jian, Hongbo Zhou, Wei Liang, Meng-Yen Hsieh, Changxu Wan

DOI: 10.1504/IJES.2024.10065506

Article History:

Received:	24 October 2023
Last revised:	09 March 2024
Accepted:	10 April 2024
Published online:	10 February 2025

The correction method of block authentication information in edge computing mode

Jianbo Xu

School of Software,
Quanzhou University of Information Engineering,
Fujian, Quanzhou, China
Email: jbxu@hnust.edu.cn

Wei Jian

Hunan University of Science and Technology,
Taoyuan Road, Yuhu District,
Xiangtan City, Hunan Province, China
Email: jian2391477260@163.com

Hongbo Zhou

School of Software,
Quanzhou University of Information Engineering,
Fujian, Quanzhou, China
Email: zhb591111@163.com

Wei Liang

Hunan University of Science and Technology,
Taoyuan Road, Yuhu District,
Xiangtan City, Hunan Province, China
Email: wliang@hnust.edu.cn

Meng-Yen Hsieh*

Providence University,
Dadu Mountain in Shalu District, Taichung City, Taiwan
Email: mengyen@pu.edu.tw
*Corresponding author

Changxu Wan

Hunan University of Science and Technology,
Taoyuan Road, Yuhu District,
Xiangtan City, Hunan Province, China
Email: wan_changxu@yeah.net

Abstract: In many industrial edge computing applications, terminal devices are often remote and widely distributed. To achieve cross-domain authentication, some researchers store authentication information on a blockchain network built by edge nodes. However, since terminal equipment may be damaged or require updates to its authentication parameters, a trusted solution for deleting and modifying blockchain-stored authentication information is needed. This paper proposes using chameleon hash trap technology to maintain blockchain integrity while enabling data correction. Experimental results show that the scheme efficiently generates and distributes sub-private keys for modifying on-chain authentication data. The modifiable blockchain is well-suited for maintaining device authentication information, ensuring system security and maintainability, and improving communication efficiency.

Keywords: blockchain; edge computing; authentication; key agreement.

Reference to this paper should be made as follows: Xu, J., Jian, W., Zhou, H., Liang, W., Hsieh, M-Y. and Wan, C. (2024) 'The correction method of block authentication information in edge computing mode', *Int. J. Embedded Systems*, Vol. 17, Nos. 3/4, pp.213–223.

Biographical notes: Jianbo Xu received his MS in Department of Computer Science and Technology from the National University of Defense Technology, China, in 1994 and PhD in College of Computer Science and Electronic Engineering from Hunan University, China, in 2003. Since 2003, he has been a Professor with the School of Computer science and Engineering, Hunan University of Science and Technology. His research interests include network security and distributed computing.

Wei Jian is a postgraduate of Computer Science and Technology from Hunan University of Science and Technology.

Hongbo Zhou is a Professor in the School of Software, Quanzhou University of Information Engineering, China. He received his MS in Computer Science and Technology from the Beijing Institute of Technology in 1993. He has published more than20 journal/conference papers and PI for more than 20 large scale scientific projects. His research interests include dependable systems/networks, network security, network measurement, hardware security, and IP protection.

Wei Liang is a Professor, Dean, and a Doctoral Supervisor at the School of Computer Science and Engineering, Hunan University of Science and Technology. He is also a Yue Lu Scholar at Hunan University, a distinguished talent of the New Century in Fujian Province, and the Executive Director of the Hunan Computer Society.

Meng-Yen Hsieh is with the Department of Computer Science and Information Engineering at Providence University.

Changxu Wan is a postgraduate of Computer Science and Technology from Hunan University of Science and Technology.

1 Introduction

With the popularity of terminal equipment such as mobile phones and portable mobile computers, a large number of terminal devices are connected to the network. In this case, the cloud server needs to provide services for every user, which is bound to add a huge burden to the computing and communication capabilities of the cloud server and easily form a bottleneck. In this context, the edge computing model came into being. Edge computing's way of sinking the computing services of cloud servers located in a trusted environment to edge nodes can well alleviate the heavy computing bottleneck problem of cloud servers. However, although the edge computing model improves the system performance of the cloud computing model, the security of the system does not improve, and even brings new security risks due to the wide geographical distribution of edge nodes and terminal devices. In recent years, many scholars have applied blockchain technology to edge computing to solve the problem of device security authentication in the edge computing mode, and applying blockchain technology to edge computing can improve the service quality of edge computing in terms of credibility, data integrity and secure sharing. Zhang et al. (2023) stored the temporary public key issued by the certificate authority in the blockchain to achieve cross-domain batch authentication. That is, the certificate is written to the blockchain, because the blockchain has the characteristics of non-tampering, therefore, the temporary public key is deployed on the blockchain, which realises the purpose of temporary public key authentication in disguise. A certificate authority is not required to verify the authenticity of the certificate, but this method does not consider the issue of certificate revocation. Liu et al. (2023b) adopted a similar approach, putting the public key into the blockchain to achieve proof of validity, while writing the user login process into the smart contract to complete the transaction confirmation. Guo et al. (2019) write the user login process to the blockchain, making it impossible for users to repudiate. Wang et al. (2020) further innovated on this basis, proposing to store the public key in the blockchain. Liu et al. (2023a) write the certificate information to the blockchain, verify the validity of the public key by verifying the validity of the blockchain data, and avoid viewing the certificate revocation list.

The above methods do not take into account that the information uploaded by the terminal device is not necessarily reliable, and the changes and damage of the terminal equipment are likely to cause the data on the chain to be unavailable. Due to the immutability of the blockchain, the data after the chain cannot be deleted or modified. Therefore, we need to study a modifiable method of blockchain to conditionally break the immutability feature of blockchain. In the research of blockchain security correction method, Aizherson proposed editable blockchain technology based on the chameleon hash function. As long as you know the trapdoor of the chameleon hash function, you can find a collision of existing data, so as to realise the editing of blockchain data. In the company's solution, the trapdoor is handed over to the trusted centre, and once the trusted centre is attacked, there is a risk of the trapdoor leakage, and the scheme is not highly decentralised.

Li et al. (2023) proposed a ledger modification scheme based on gate ring signature, which relies on the block structure of the proof-of-space consensus mechanism. When the data expires or expires, one node puts forward a deletion proposal, other nodes vote for the proposal, and when the

node that agrees to the proposal exceeds the set threshold, the system generates an exclusive deletion message; then, the node that agrees to the proposal becomes the signature node, which generates a threshold loop signature on behalf of the entire system; finally, the proponent stores the generated ring signature in the original location of the transaction data, and then broadcasts it to the whole network to complete the deletion operation. Later, Ren et al. (2019, 2020) improved their scheme (Zhang and Lee, 2019), by reconstructing the signature subblock in the block body, adding the manoeuvre factor to the block so that it did not change the hash value of the block before and after the modification. However, the disadvantage of this method is that it depends on a specific block structure and is not universal to most block structures.

Based on the alliance chain, Guo et al. (2019) improved the chameleon hash algorithm, generated the chameleon hash subkey of the system on a trusted node, and then used the random number generation protocol and secret sharing technology to randomly select the modifiers to modify the block when there is a need for modification. The scheme improves the chameleon hash algorithm in several aspects, and when the ledger modification operation is required, the modification operation can be completed without additional calculations by other nodes, and the scheme is based on a trusted central node, and the modification does not skip the step of centralisation.

Wang et al. (2020) optimised the chameleon hash algorithm, which also generates the chameleon hash key of the system by a trusted node, splits the system private key into multiple private key shares through secret sharing, and distributes it to all nodes in the network or the nodes with the highest shares; when the ledger modification operation is required, the sub-private key of each node is restored to the system private key, so as to modify the transaction content. There is a problem with this scheme: although the sub-private key of the chameleon hash is stored on multiple nodes, and the ledger modification process reaches a certain degree of decentralisation, the generation and restoration of the key requires a centralised trusted node, so the degree of decentralisation of the scheme is not high.

In this paper, in the edge computing mode, the edge nodes constitute a blockchain, which solves the cross-domain authentication problem of terminal devices. The chameleon hash method is adopted to realise block modification, which solves the problem of modification of authentication information caused by terminal device changes and damage. The modification method is that the service centre stores the private key of the chameleon hash in the form of a time slice and publishes the public key of the chameleon hash, and when the edge node needs to modify the data in the blockchain, the service centre splits the private key into multiple copies and distributes it to multiple edge power nodes. After the edge node receives the sub-private key of each edge power node, it synthesises the trapdoor master private key for the edge node to modify the data.

2 Preliminaries

2.1 Chameleon hashing algorithm

The reason why blocks in a blockchain are difficult to modify between each other is that blocks are generated by hashing. Modification of one hash in the hash link causes subsequent link hash errors (Gong et al., 2023; Zhang et al., forthcoming, 2023). The chameleon hash function, also known as the trapdoor hash function, was first proposed by Krawczyk and Rabin (Ren et al., 2019; Hu et al., 2022; Zhou et al., 2023), and the corresponding collision can be easily calculated when the trapdoor key is mastered. The existing method of modifying the blockchain uses the chameleon hash function to modify the block data without changing the hash value.

The chameleon hashing algorithm consists of three main parts: hash function, chameleon trapdoor generation, and hash value generation.

- Hash function: The chameleon hash algorithm uses a basic hash function as its base. The function accepts an input message and outputs a fixed-length hash value. This hash function can be any commonly used hash function such as SHA-256 or MD5. In the chameleon hash algorithm, we use a hash function to generate a hash value h, which will be used to generate the chameleon trapdoor and hash value.
- Chameleon trapdoor generation: A chameleon trapdoor is a secret parameter that generates a hash value associated with a specific key. Specifically, the chameleon trapdoor is generated by a public parameter (hash value h) and a secret parameter (key k). In the chameleon hash algorithm, we use a special chameleon trapdoor generation algorithm to generate the chameleon trap. The algorithm first selects a random number b and computes a parameter a that is independent of the key k, and then it calculates a random number r such that $r = \frac{h-b}{a}$, while the chameleon trapdoor t can be generated by using the chameleon trapdoor generation algorithm, $t = \frac{k-b}{r}$. Its chameleon trapdoor t is a secret parameter associated with the key k.
- Hash value generation: The generation of hash value is the final step of the chameleon's hashing algorithm. In the hash generation phase, we use the chameleon trapdoor t to generate the hash value associated with the key k. Specifically, we use the common parameter h, the chameleon trapdoor t, and a random number x to generate the hash value. The hash value is calculated as: $h' = t * x + b \bmod p$, where b and p are the random numbers and large prime numbers used in the chameleon trapdoor generation algorithm.

For Krawczyk and Rabin (1998), the chameleon hash algorithm has five functions, which are described as follows:

- Initialise $Setup(\lambda)$: From the security parameter λ, calculate the public parameter $pp = \{p, q, g\}$. And p and q satisfy the equation: $p = kq + 1$, g is the generator of the multiplicative cyclic group Z_p^*.
- Generate key $GenKey\,(pp)$: According to the parameters pp generated by initialisation, output the public key h, private key s. s is the element randomly selected in the multiplicative cyclic group Z_p^* above, and the public key $h = g^s \bmod q$.
- Calculate the hash value $CalHash(h, m, t)$: Through the known public key h, plaintext m, and variadic r, the corresponding chameleon hash value $CH = g^m h^r \bmod p$ is obtained.
- Parameter t forge function $Forge(s, m, t, m')$: via the trapdoor key s, the original m and the modified original text m' and the parameter t. Output the variadic parameter t' matched by m' after modification.
- Verify the chameleon hash $Verify(h, m, t, CH)$: Verify whether the chameleon hash CH corresponds to the plaintext m, parameter t, and enter the above parameters. If output 1 is matched, output 0 is not matched.

According to the above principle and algorithm, the processing of the message using the chameleon hash requires two parameters: message m and parameter t. The parameter t is mutable, and after the message m is modified by the edge node, the parameter t can be changed so that $H(m, t) = H(m', t')$.

2.2 Shamir key distribution and recovery algorithm

The method of modifying blockchain data is to replace the traditional hash function with a chameleon hash, in this scheme, most of the research scheme is a decentralised method, each node's private key generation algorithm can only generate its own sub-private key, the modified trapdoor private key is the total private key $s = s_1 + s_2 + \ldots + s_k$.

In the mode of edge computing, the location and security status of edge nodes is complex and changeable, and the location and security of the registry can be guaranteed. A centralised approach can be adopted in this environment. That is, in the blockchain composed of edge nodes and registration centres, the registry is used as the central trusted node in the blockchain, the registry is used as the key distribution centre, and the private key of the chameleon hash is split into multiple copies for distribution.

The key splitting and recovery algorithm in this paper is the Shamir key distribution and recovery algorithm (Ren et al., 2020; Ateniese et al., 2017; Krawczyk and Rabin, 1998). Shamir's secret sharing, a cryptographic technique used to distribute secret information among multiple participants, was first proposed by Israeli cryptographer Adi Shamir in 1979 (Li et al., 2018; Rong et al., 2015; Shamir, 1979; Lv et al., 2021). It can be used to securely distribute passwords or other sensitive information so that the original secret can only be recovered if certain conditions are met.

In the chameleon hash key generation algorithm, the private key s generated by the $GenKey\,(pp)$ algorithm is a randomly selected element in the multiplicative loop group Z_p^*. Therefore, the Shamir key distribution and recovery algorithm is used to fit the key distribution and recovery scenario. The specific methods of splitting the key, distributing the key, and recovering the key are as follows.

- Initialisation: The registry CS randomly selects n (the number of power nodes) different non-zero element x_1 from the finite field $GF\,(p)$, $x_2 \ldots$, x_n, the registry CS has non-zero element information corresponding to n power nodes, and exposes its non-zero element $x_r (1 \le r \le n)$ for each power node r.
- Subkey distribution phase: The secret to be distributed by the registry is a subkey of the chameleon hashgate. Randomly select $(t-1)$ elements in $GF\,(p)$, the $a_i (i = 1, 2 \ldots, t-1)$ to form a polynomial of order $t-1$, see equation (1).

$$f\,(x) = \sum_{i=1}^{t-1} a_0 + a_i x^i \bmod p \tag{1}$$

  p is a large prime number, and $p > s$, private key $s = f\,(0) = a_0$. The registry generates a subkey for the authority edge node, as shown in equation (2), and sends it to the corresponding authority edge node in a secure manner.

$$\begin{aligned} s^r &= f\,(x^r) \\ &= \sum_{i=1}^{t-1} a_0 + a_i x^i \bmod p \quad (r = 1, 2, \ldots, n) \end{aligned} \tag{2}$$

- Chameleon hash trapdoor recovery: Nodes that hold subkeys of t or greater than t power edge nodes can recover the hash trapdoor private key S using the Lagrange interpolation formula, and nodes with less than t subkeys cannot recover the private key s, see equation (3) for the Lagrange interpolation formula.

$$\begin{aligned} s &= f\,(0) \\ &= \sum_{i=1}^{t-1} f(x^i) \prod_{v=1, v \ne l}^{t} \frac{-x_v}{x_i - x_v} \bmod p \end{aligned} \tag{3}$$

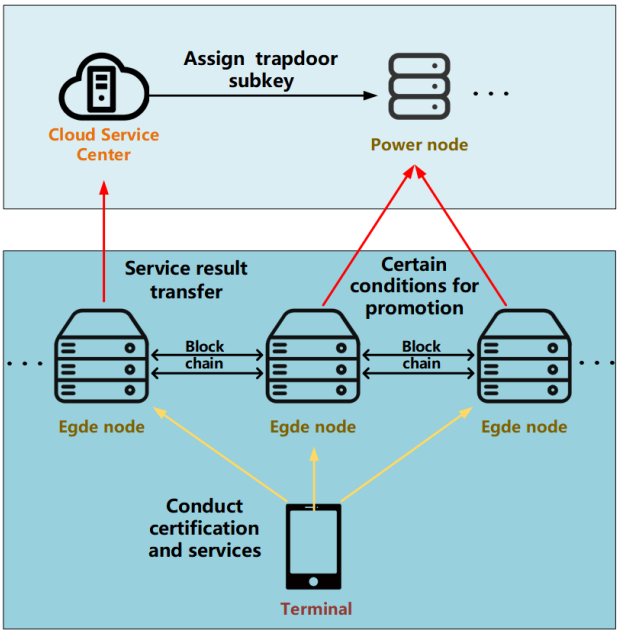
3 Method details

3.1 Network model

The model architecture is shown in Figure 1 and consists of three main parts. User devices, edge computing nodes (edge servers) and cloud service centres (registration centre). The architecture diagram is roughly the same as that studied by the majority of scholars, the only difference is the addition of the concept of power nodes, power nodes are special

edge nodes, edge nodes can be upgraded to power nodes after meeting certain conditions (see below for promotion conditions), power nodes are in charge of the sub-keys assigned by the cloud service centre. An edge computing node and several end devices form an edge service domain. In the system architecture, there is one blockchain for storing the digital certificate after the terminal device is successfully authenticated, and the other blockchain stores the device information of the active terminal.

Figure 1 Network architecture (see online version for colours)



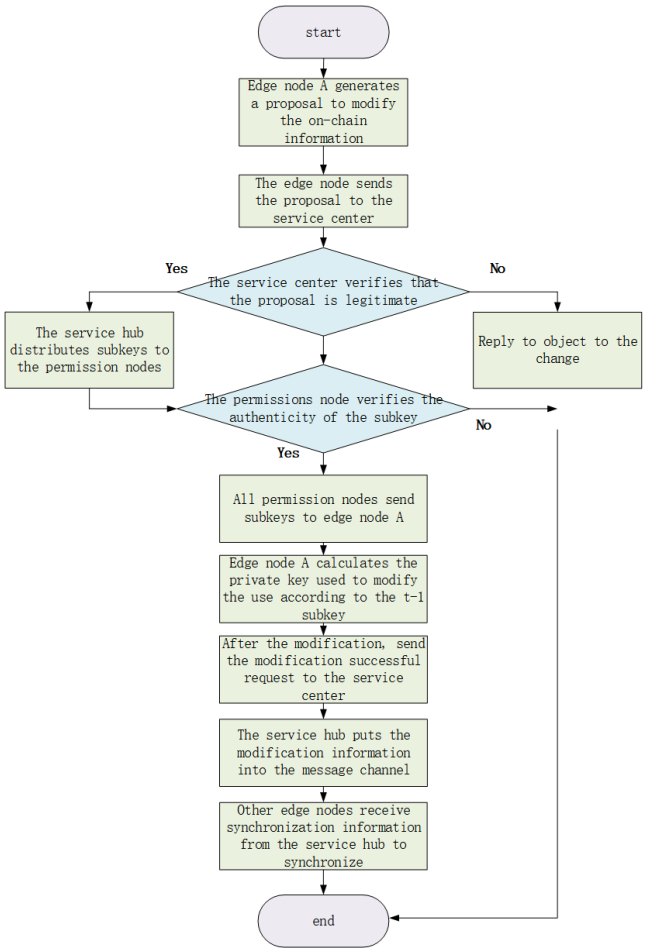
3.2 Authentication information modification methods

Terminal device authentication process: when a terminal device initiates an authentication request to an edge node, the edge node first checks whether the terminal device has ID information locally, that is, the edge node determines whether the terminal device is a recently active terminal device, and if the ID information of the terminal device is found locally on the edge node, it directly returns the authentication success. If the terminal device information does not exist at the edge node, it will go to the full node on the chain to query whether there is digital certificate information, and the authentication is successful if there is a digital certificate.

Modify the process of block authentication information: the flow chart of modifying authentication information is shown in Figure 2, and the specific modification process is as follows:

- When the terminal device changes and needs to modify the authentication information on the blockchain, we need the edge node to generate a proposal to modify the on-chain information and send the modification proposal to the service centre, and the information proposal is accompanied by the device ID of the terminal device, the reason for modifying the authentication information and the time when the information is sent.

Figure 2 Block modification flowchart (see online version for colours)



- The service centre accepts the proposal and verifies its modification request, if the verification is not passed, it sends an objection to the modification message to the proposal initiator, if the verification is passed, the service centre finds the trapdoor private key of the chameleon hash in the period (the modification of the period requires the chameleon hash key for operation), splits it into multiple sub-private keys and assigns it to the power node (the power node is generated by the edge node with a large number of service terminal devices), and the number of split sub-private keys determines the security of its chameleon hash private key. Because the private key is formed by merging sub-private keys, the greater the number of sub-private keys that are split, the lower the risk of leakage of the merged private key and the higher the system security.
- When each power node receives and verifies the message sent by the service centre, it encrypts the sub-private key it holds and sends it to the edge node that initiated the modification request, after the sub-private key is sent, the power node can choose to

destroy or retain the sub-private key, even if the power node retains the sub-private key, privately owning a sub-private key can not modify the corresponding block of the blockchain, even if it has all the sub-private keys, becomes a malicious node, and privately modifies the local blockchain node. It is also impossible to synchronise other edge nodes with their local blockchain information due to the lack of participation of the service centre.

- Once the modification operation is completed, the edge node also needs to return a reply message to the service centre, telling the service centre that it has been modified, and with the identity sent by the service centre, proving that the modification is completed by the key distributed by the service centre, then the service centre can send synchronisation requests to other edge nodes to update the blockchain information of other nodes in the entire distributed system.

3.3 A method by which an edge node is promoted to a power node

Compared with ordinary edge nodes, the only difference between the power node and ordinary edge nodes is that it needs to accept the sub-private key issued by the service centre and send its sub-private key to the edge node that made the modification request, which can judge the request issued by the edge node to modify the on-chain data, and determine that the request for joint issue will send the sub-private key it holds to the edge node. There are two more suitable scenarios for promoting an edge node to a power node:

- Vote based on the edge node. That is, the edge node that wants to be promoted to a power node initiates a promotion request to other nodes, and the other edge nodes judge the authenticity of the node after receiving the promotion request, and return the consent request if it is reliable. After receiving a consent request from a certain threshold, the edge node that wants to be promoted to an authority node is promoted to an edge node. The communication overhead caused by this method will be relatively high.
- Based on the edge node workload. This is the approach we have adopted. The workload refers to the number of service terminal devices, because the edge node will regularly transmit its successfully authenticated terminal devices to the service centre, so the service centre stores the number of terminals served by each edge node. We select the edge node with the top x of the workload to promote to the power node. Compared with the above scheme, this scheme has less communication overhead, and it does not need to vote for power nodes, because the service centre stores the workload information of each edge node, and only needs the service centre to select the top x ($0 < x < 100$) nodes with the highest workload as power nodes.

3.4 Chameleon hash key generation granularity

The chameleon hashing algorithm requires a pair of public and private keys when modifying data. In addition to the service node's need to store the private key for each modification, which adds a burden to the storage space of the service centre, the service centre's broadcast of the public key and the process of sending the sub-private key to the power node, and the service centre's acceptance of the process of initiating the modification node proposal delivery process also require additional communication costs. Therefore, in this case, different key generation granularity has a greater impact on the communication efficiency and storage space of the system, according to the research of Lv et al. (Jian et al., 2022; Liang et al., 2020; Cai et al., forthcoming) commonly used key generation granularity is divided into the following:

- Key generation granularity: Every time the data is on the chain. For each data write in the blockchain, a chameleon hash key pair needs to be generated. This method is very secure, and leaking a chameleon hash public private key will only cause the corresponding data to be tampered with, and any other on-chain data cannot be modified. But the biggest disadvantage of this method is that it brings great storage pressure and communication pressure to the entire blockchain distributed system, once the device writes a large amount of data to the blockchain, this will greatly increase the burden of the entire distributed system, because the full nodes in the blockchain will synchronise all newly added and modified data.
- Key generation granularity: Each block produced. Every time a new block is generated, the corresponding chameleon hash key pair is generated at the same time, which is equivalent to the previous key generation granularity, and its time and space cost is much less. In the Bitcoin system, a block size is about 1 MB, and a transaction data requires about 250 B of space, that is, a block can hold about 2,500 transactions, so the key generated by the block for granularity is equivalent to the granularity of each data on the chain, and the efficiency can be increased by about $2 * 10^3$ times. When an attacker obtains the key pair, it can modify any data of the corresponding block at will, and this key generation density is only suitable for cases where the number of modifications on the chain is relatively small.
- Key generation granularity: Per time period. The key generation granularity used in this article is generated for each time period, and we put the key generation density in each time period to save overhead and ensure data security. The service centre publishes a key pair per time period, and the data on the chain during that time period is hashed by the chameleon.

The specific time period size can be determined according to the frequency of specific modification operations, and the time period can be adjusted longer with high modification frequency to prevent excessive communication and storage overhead caused by the private key pair.

- Key generation granularity: Only one key is generated in the chain. That is, all data on-chain operations of the blockchain only use this chameleon hash key pair, which can modify all data on the chain. This scheme has the lowest communication overhead and storage overhead, and it has the lowest security factor, and once the attacker has its key, he can modify any data on the chain. This solution is suitable for scenarios where the value of on-chain data is low, and the communication overhead and storage overhead are extremely low.

4 Experiment details

Due to the limited experimental environment and funds, we cannot construct a real distributed system for complete experiments. Therefore, the experimental step we use is to go through the above technical route to prove that the proposed method is indeed feasible and effective.

This section will show Shamir key distribution and recovery, chameleon hash technology modification experiments, and blockchain initialisation, storage of authentication information, modification of authentication information and other experiments in a stand-alone environment. And analyse the time and performance overhead of these experiments.

4.1 Experimental environment

Our experimental environment is as follows:

- Hardware: This section conducts simulation experiments in the Intel i5 9300H processor (2.4 GHz), 16 G memory, and 1 T mechanical disk Lenovo computer.
- Software: Under the Windows 10 operating system. Use the jpbc 2.0.0 cryptography package in the Java JDK 1.8 environment.

4.2 Shamir key distribution and recovery experiment

This experiment was performed using a Shamir toolkit packaged according to the jpbc (Long et al., 2023; Diao et al., 2023; Hu et al., 2023; Liang et al., 2022, 2023) cryptography package. The Shamir key distribution algorithm requires two parameters n and t, n is the number of copies of the key distribution, t represents the minimum score required to recover the key, Figure 3 is Shamir's demo code.

Figure 3 Shamir key distribution code

Shamir key distribution code

```
public static void doIt(int n,int t) {
    final Scheme scheme = new Scheme(new
    SecureRandom(), n, t) ;
    final byte[] secret = "hello,this is
    mysercret".getBytes(StandardCharsets.UTF_8);
    finalMap<Integerbyte[]>parts=scheme.split(secret);
    Map<Integer, byte[]> user=new HashMap<Integer,
    byte[]>();
    for (int i=0;i<t;i++){
       user.put(i+1,parts.get(i+1));
    }
    final byte[] recovered = scheme.join(user);
    String sum_secret= new String(recovered,
    StandardCharsets.UTF_8);
    System.out.println("The value of
    n"+String.valueOf(n)+"    "+"The value of
    t"+String.valueOf(t));
    System.out.println("Post-restore information：
    "+sum_secret);
}
```

In the demo code, we use a simple English paragraph to represent our private key, and then pass in the required parameters n, t in the doIt method, where we get n is 10 and t is 8. It represented the distribution of ten different paragraphs of information in English and the restoration of the eight paragraphs of information that needed to be distributed in English. After this code passes through the compiler runner, it can be found that the value of the sum_secret is summarised by 8 pieces of information and successfully restored to the beginning of the English paragraph, that is, our private key.

4.3 Chameleon hash technique modification experiment

First, after encapsulating the chameleon hash class according to the cryptography JPBC library in Java, then implementing the initialisation $Setup(\lambda)$, generating the key $GenKey\,(pp)$, trapdoor $Forge(s, m, t, m')$ and other methods required to achieve the chameleon hash, and finally start the corresponding experiment.

At the beginning of the experiment, the hash value of key generation and initial information was calculated through the encapsulated chameleon hash class, and then the initial value of the initial value was changed, and then the hash value was calculated.

This is known from the compiler result, strMsg1 and strMsg2 are two completely different initial information, calculated by the Forge trapdoor function, and their hash values are found separately, and it can be found that the hash values of the two are exactly the same. Therefore, for the chameleon hash, even if two completely different pieces of information, after mastering the trap, the same chameleon hash value can be constructed.

Compared with the traditional hashing algorithm, the chameleon hashing algorithm mainly has the following disadvantages:

- Key compromise risk: Traditional hash functions do not have a key, and if the key of the chameleon hash

is compromised, an attacker can use the key to forge the hash value, thereby compromising the security of the system.

- Attacks that can target keys: The security of chameleon hashes relies on the confidentiality of the keys. If an attacker is able to crack the key, they can impersonate a legitimate user by generating a valid hash value to access sensitive data or perform other malicious behaviour, threatening the security of the entire system.
- Computationally inefficient: Chameleon hashes are slower to compute compared to traditional hash functions, which can negatively impact the performance of the system.
- Chameleon hashing implementation complexity: Because the algorithm for chameleon hashing is complex, the software or hardware that implements the algorithm may require more time and resources.

4.4 Blockchain experiment in a stand-alone environment

Chain initialisation: the initialisation process of a blockchain typically consists of the following three steps:

- Create a genesis block: A genesis block is the first block of the blockchain and contains some initial parameters and states of the blockchain.
- Define blockchain rules: Define the transaction rules, consensus algorithm, block size, mining reward, etc. of the blockchain to ensure the stability and security of the network.
- Setup nodes: Start nodes and connect them to the network, ensuring that nodes can receive and broadcast transaction information.

During chain initialisation, the main work is to complete the genesis block generation step and generate the chameleon hash key on a regular basis. The genesis block is the starting point of the entire blockchain. It is generated by the creator or initial team of the blockchain and usually contains some initial metadata such as initial parameters, initial state, etc. The genesis block can also be thought of as the ‘initial state’ of a blockchain network, as all subsequent blocks are built on top of it. The genesis block header JSON format of the blockchain in our stand-alone environment in Figure 4.

The meaning of the symbols shown in Figure 4 is as follows, ‘id’ is 0 means that it is the genesis block, the ‘hash’ value is the hash value of all parameters of the current block together to calculate the hash value, ‘previous_hash’ represents the hash value of the previous block, the above value is 0 because the genesis block does not have a predecessor block, ‘Merkel’ represents the last hash obtained after many authentication information is obtained after seeking hash separately, and ‘Merkel’ in the genesis block the root value is still empty because the authentication information has not been stored in the genesis block body.

Figure 4 Genesis block header


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
{
"id":0,
"hash":"00001628294da6d40e72be413f975d23dd776cc224
        2997ddb06438482c03fb04",
"previous_hash":"0",
"merkle_root":null,
"time_stamp":"2023-01-02 11:03:31",
"nonce":132397
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```


Since the chameleon hash key is required to modify the authentication information, we write a method to generate the chameleon hash key, the main code of which is shown in Figure 5.

Figure 5 Generate key code

```
Generate key code
public void GenKey(String StrNow)
    throws IOException
{
    this.chameleonService=ChameleonService.getIn
    stance();
    //Generate a chameleon private key
    ChameleonHashKey key =
    chameleonService.keyGenerator();
    chameleonHashKeyService.
    SaveChameleonHashKeyToKeyToData( key,S
    trNow);
    System.out.println("Generate the current
    chameleon hash key : " +
    LocalDateTime.now().toLocalTime()
    +"\r\nthread : "
    +Thread.currentThread().getName());
}
```

According to the concept of key generation granularity mentioned in the above subsection, the granularity we took in the experiment for each time period to generate a chameleon hash key, in the service centre we must generate a chameleon hash key at regular intervals, the key includes the public key and the private key, and broadcast the public key in the key to other edge nodes, so that the hash value of all authentication information during this period is obtained by the chameleon hash algorithm with the public key as a parameter. Therefore, in order to generate the corresponding chameleon hash key, we set the corresponding timing task in the system, the timing time is set by ourselves, I set it to an hour here, let the code run every hour to generate the corresponding chameleon hash key, and then we use class serialisation to store the chameleon hash key into the service centre database.

Stores authentication information: To store authentication information, you need to obtain the chameleon hash public key for the chameleon hash operation in the current time period, which is broadcast by the service centre regularly, and the edge node needs to

listen to the channel at regular intervals to accept the public key broadcast by the service centre in order to calculate the chameleon hash value of the authentication information.

Figure 6 Stored authentication information

In the authentication information, the authentication information needs to specify the block number stored, and the authentication information stored in each block has an upper limit, that is, the authentication information that can be stored in a block is limited, and the upper limit needs to be specified by the system, and the upper limit is specified as four pieces of information in this experiment. After storing the information, the block header and block body are shown in Figure 6.

The block header is mainly composed of its own block ID, block hash value, as well as the four parts of the precursor block hash value and Merkel root value, and the relevant information of block generation, such as timestamp and block generation difficulty coefficient nonce two parts. The block body is mainly composed of authentication information and key information used to store authentication information, and hash and r are the trapdoor information generated by the chameleon hash, which is used to obtain its chameleon hash trap when modifying the authentication information. Each time the authentication information is stored, the Merkle root value in the block body will be changed, and when the block authentication information storage reaches its storage limit, the Merkle root value information will not be modified.

Modify the authentication information: the chameleon hash is used in the block structure as shown in Figure 7, the lock on the authentication data represents that the hash function of the transaction data hash is replaced with the chameleon hash function, and the change chameleon hash trap and the chameleon hash private key need to be obtained to modify the authentication information.

Figure 7 The position of the chameleon hash in the block (see online version for colours)

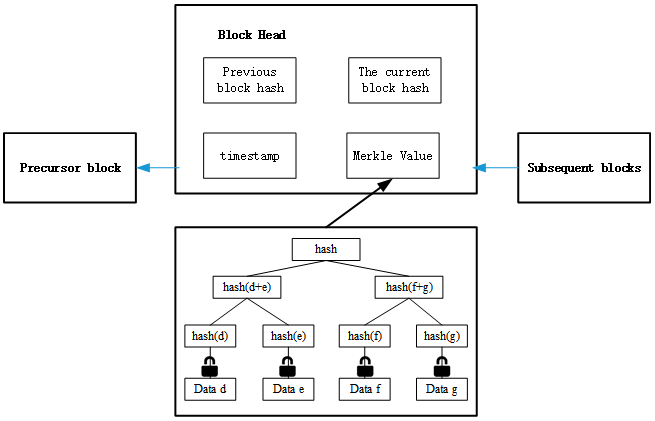


Figure 8 Modified authentication information

Chameleon hash trap: The chameleon hash trap is calculated when the authentication information is found when the chameleon hash, it is stored in the relevant place of the authentication information, that is, in the authentication information in the block body can find its chameleon hash trap, whenever the authentication information is modified, directly make these three parameters 'modified information, chameleon hash trap, chameleon hash private key' can find the hash value of the original information. Although the chameleon hash trap can be calculated by the public key and the original information, the amount of computation and time overhead associated with the latter method is significantly reduced compared to the method of directly storing in the block and obtaining it directly from the block information.

Chameleon hash private key: The chameleon hash private key requires the edge node to query the current block generation time key_gen_time. The generation time is attached to the information modification proposal and sent to the service centre, which verifies the proposal and distributes the chameleon hash private key to the corresponding permission node. When the edge node

merges the chameleon hash private key from the permission node through the key merging algorithm, it can modify the data in the block body together with the chameleon hash trap.

The specific modification of the block structure before and after the change is shown in Figure 8, see that the information in the 'message' has changed before and after, after modifying the single authentication data, due to the use of the chameleon hash trap, the single data hash value will not change, the hash value of the two hashes will not change, see Merkle root structure diagram, so the hash value and Merkle root value before and after modification will not change, and the block structure of the precursor block will not change with modification. The rear-drive block will also not change.

For the question of how to synchronise blocks, the updated block will be sent to the service centre by modifying the node, and after the block is verified by the service centre, each edge node in each domain is notified to synchronise its own local block.

So, when the authentication device is damaged or changed, the invalid authentication information of the endpoint can be deleted or changed, saving the storage space of the edge device and facilitating the management and maintenance of the authentication information.

5 Conclusions

In the edge computing environment, in order to solve the problem of editing and modifying the authentication information in the blockchain formed by edge nodes, this paper adopts the method of chameleon hash and key generation time slice to edit and modify the authentication data in the blockchain. For the distribution and recovery of chameleon hash keys during the modification process, we adopt the policy of using the service centre as the centre of key distribution, distributing the key through the service centre, and restoring the key at the modification node.

In the experimental part, this paper tests the Shamir key distribution and recovery algorithm and tests the time overhead of the algorithm. The algorithm has less time overhead when the number of distribution keys is small and the number of subkeys required for merging is small. After the chameleon hash and key distribution experiments, it is verified that there are no technical obstacles to the method of chameleon hashing and key distribution technology to modify the on-chain data, and finally, this paper conducts the storage and modification of authentication information experiments in the stand-alone blockchain to simulate the storage and modification process in the actual situation.

After the above experiments, it is proved that when the authentication equipment is damaged or changed, that is, when the corresponding authentication data on the edge node needs to be modified, the proposed scheme can reliably and effectively modify the authentication information in the actual environment, and will not change the block structure before and after, and adapt to a variety of consensus algorithms.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant 61872138, 61572188) and the Natural Science Foundation of Fujian Province, China (Grant 2023J011800).

References

Ateniese, G., Magri, B., Venturi, D. et al. (2017) ' Redactable blockchain-or-rewriting history in bitcoin and friends', *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, pp.111–126.

Cai, J., Liang, W., Li, X., Gui, Z., Li, K-C., Khan, MK. (forthcoming) 'GTxChain: a secure IoT smart blockchain architecture based on GNN', *IEEE Internet of Things Journal*, IEEE, DOI: 10.1109/JIOT.2023.3296469.

Diao, C., Zhang, D., Liang, W., Li, K-C., Hong, Y. and Gaudiot, J-L. (2023) 'A novel spatial-temporal multi-scale alignment graph neural network security model for vehicles prediction', *IEEE Transactions on Intelligent Transportation Systems*, Vol. 24, No. 1, pp.904–914, January, DOI: 10.1109/TITS.2022.3140229.

Gong, Y., Li, K., Xiao, L., Cai, J. et al. (2023) 'VASERP: an adaptive, lightweight, secure, and efficient RFID-based authentication scheme for IoV', *Sensors*, Vol. 23, No. 11, p.5198, DOI: 10.3390/s23115198.

Guo, S., Hu, X., Guo, S. et al. (2019) 'Blockchain meets edge computing: a distributed and trusted authentication system', *IEEE Transactions on Industrial Informatics*, Vol. 16, No. 3, pp.1972–1983.

Hu, N., Zhang, D., Xie, K., Liang, W., Diao, C. and Li, K-C. (2022) 'Multi-range bidirectional mask graph convolution based GRU networks for traffic prediction', *Journal of Systems Architecture*, Vol. 133, p.102775, Elsevier, DOI: 10.1016/j.sysarc.2022.102775.

Hu, N., Zhang, D., Xie, K., Liang, W., Li, K. and Zomaya, A. (2023) 'Multi-graph fusion based graph convolutional networks for traffic prediction', *Computer Communications*, Vol. 210, pp.194–204, Elsevier, DOI: 10.1016/j.comcom.2023.08.004.

Jian, W., Xu, J., Liang, W. and Li, K-C. (2022) 'Dual chain authentication and key agreement protocol based on blockchain technology in edge computing', *2022 IEEE 24th Int. Conf. on High Performance Computing & Communications; 8th Int. Conf. on Data Science & Systems; 20th Int. Conf. on Smart City; 8th Int. Conf. on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, Hainan, China, 2022, pp.396–401, DOI: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00082.

Krawczyk, H. and Rabin, T. (1998) *Chameleon Hashing and Signatures*, Cryptology ePrint Archive.

Li, P., Xu, H., Ma, T. et al. (2018) 'Research on fault-correcting blockchain technology', *Journal of Cryptologic Research*, Vol. 5, No. 5, pp.501–509.

Li, Y., Liang, W., Xie, K., Zhang, D., Xie, S. and Li, K-C. (2023) 'LightNestle: quick and accurate neural sequential tensor completion via meta learning', *IEEE INFOCOM 2023*, IEEE, DOI: 10.1109/INFOCOM53939.2023.10228967.

Liang, W., Li, Y., Xie, K., Zhang, D., Li, K-C., Souri, A. and Li, K. (2022) 'Spatial-temporal aware inductive graph neural network for C-ITS data recovery', *IEEE Transactions on Intelligent Transportation Systems*, Vol. 24, No. 8, pp.8431–8442, IEEE, DOI: 10.1109/TITS.2022.3156266.

Liang, W., Yang, Y., Yang, C., Hu, Y., Xie, S., Li, K-C. and Cao, J. (2023) 'PDPChain: a consortium blockchain-based privacy protection scheme for personal data', *IEEE Transactions on Reliability*, Vol. 72, No. 2, pp.586–598, DOI: 10.1109/TR.2022.3190932.

Liang, W., Zhang, D., Lei, X., Tang, M. and Zomaya, Y. (2020) 'Circuit copyright blockchain: block chainbased homomorphic encryption for IP circuit protection', *IEEE Transactions on Emerging Topics in Computing*, DOI: 10.1109/TETC.2020.2993032.

Liu, S., Xiao, L., Han, D., Xie, K., Li, X. and Liang, W. (2023a) 'HCVC: a high-capacity off-chain virtual channel scheme based on bidirectional locking mechanism', *IEEE Transactions on Network Science and Engineering*.

Liu, Y., Liang, W., Xie, K., Xie, S., Li, K. and Meng, W. (2023b) 'LightPay: a lightweight and secure off-chain multi-path payment scheme based on adapter signatures', *IEEE Transactions on Services Computing*.

Long, J., Liang, W., Li, K-C., Wei, Y. and Marino, M.D. (2023) 'A regularized cross-layer ladder network for intrusion detection in industrial internet-of-things', *IEEE Transactions on Industrial Informatics*, February, Vol. 19, No. 2, pp.1747–1755, DOI: 10.1109/TII.2022.3204034.

Lv, W., Wei, S., Yu, M. et al. (2021) 'Research on verifiable blockchain ledger redaction method for trusted consortium', *Chinese Journal of Computer*, Vol. 44, No. 10.

Ren, Y., Xu, D., Zhang, X. et al. (2019) 'Deletable blockchain based on threshold ring signature', *Journal on Communications*, Vol. 40, pp.71–82.

Ren, Y., Xu, D., Zhang, X. et al. (2020) 'Scheme of revisable blockchain', *Journal of Software*, Vol. 31, No. 12.

Rong, H., Mo, J., Chang, B. et al. (2015) 'Key distribution and recovery algorithmbased on Shamir's secret sharing', *Journal on Communications*.

Shamir, A. (1979) 'How to share a secret', *Communications of the ACM*, Vol. 22, No. 11, pp.612–613.

Wang, J., Wu, L., Choo, K.K.R. et al. (2020) 'Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure', *IEEE Transactions on Industrial Informatics*, Vol. 16, No. 3, pp.1984–1992.

Xu, Z., Liang, W., Li, K-C., Xu, J., Zomaya, A.Y. and Zhang, J. (2022) 'A time-sensitive token-based anonymous authentication and dynamic group key agreement scheme for Industry 5.0', *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 10, pp.7118–7127, DOI: 10.1109/TII.2021.3129631.

Zhang, S. and Lee, J.H. (2019) 'A group signature and authentication scheme for blockchain-based mobile-edge computing', *IEEE Internet of Things Journal*, Vol. 7, No. 5, pp.4557–4565.

Zhang, S., Hu, B., Liang, W., Li, K-C. and Gupta, B.B. (2023) 'A caching-based dual K-anonymous location privacy-preserving scheme for edge computing', *IEEE Internet of Things Journal*, Vol. 10, No. 11, pp.9768–9781, DOI: 10.1109/JIOT.2023.3235707.

Zhang, S., Hu, B., Liang, W., Li, K-C. and Pathan, A-S.K. (forthcoming) 'A trajectory privacy-preserving scheme based on transition matrix and caching for IIoT', *IEEE Internet of Things Journal*, IEEE, DOI: 10.1109/JIOT.2023.3308073.

Zhang, S., Yan, Z., Liang, W., Li, K-C. and Dobre, C. (2023) 'BAKA: biometric authentication and key agreement scheme based on fuzzy extractor for wireless body area networks', *IEEE Internet of Things Journal*, IEEE, DOI: 10.1109/JIOT.2023.3302620.

Zhou, S., Li, K., Xiao, L., Cai, J. et al. (2023) 'A systematic review of consensus mechanisms in blockchain', *Mathematics*, Vol. 11, No. 10, p.2248, DOI: 10.3390/math11102248.