Constraints-based and One-time Modification Redactable Blockchain

Chunli Wang, Yuling Chen, Wensheng Jia*

College of Mathematics and Statistics, Guizhou University, China
13476006558@163.com, ylchen3@gzu.edu.cn, wsjia@gzu.edu.cn

Abstract

Immutability enables the blockchain to store data permanently, which is considered to be the most essential property to protect the security of blockchain technology. However, illegal content stored on the blockchain cannot be modified either. "The right to be forgotten" stipulates that the data subject has the right to request the data controller to delete personal data about him, and the controller is obliged to delete the personal data in a timely manner in such circumstances. Therefore, redactable blockchains are proposed to solve the aforementioned problems. The redactable blockchain relaxes the immutability of blockchains in a controlled manner. However, a participant who is granted the privilege to modify the blockchain, he/she may add inappropriate or malicious modifications to the content of historical blocks. In this article, we introduce a redactable blockchain where transaction owners restrict what editors can rewrite in transactions. When posting an editable transaction, the transaction owner indicates what can be edited and what cannot be edited in order to restrict a transaction modifier from making malicious changes to the content of the transaction. The modifier's changes to the transaction are then validated by the validator and any malicious changes that do not meet the requirements will fail to be validated. To counteract collusive attacks between modifiers and validators, they will be held accountable and penalized.

Keywords: Chameleon hash, Redactable, Blockchain rewriting

1 Introduction

Since Bitcoin [1] was proposed by Satoshi Nakamoto in 2008, the blockchain, which is the underlying technology of Bitcoin [2], has received extensive attention from scholars. So far, scholars have given the characteristics of blockchains and studied aspects such as selfish mining attacks [3] and collusion attacks [4] to which they are potentially vulnerable. With the continuous development of blockchain technology, its application has been extended to digital finance [5-6], Internet of Things [7-8], intelligent manufacturing [9-10], supply chain management [11-12], digital asset trading [13-14] and other fields, showing broad application prospects. Applying blockchain technology to the digital financial framework and establishing a financial asset management system based on digital technology can automatically connect assets of different industrial forms, thereby creating an economic network that integrates financial asset digitization and asset management. By combing the application of blockchain technology in digital finance, analyzing and comparing related cases in multiple digital asset transactions, Y. Zhu et al. [15] summarized the advantages of blockchain technology in digital finance. Moreover, various difficulties and obstacles are encountering in the application of blockchain technology under the current circumstances. The author's analysis of the application of blockchain technology in digital finance provide more inspiration for the application of this technology in digital assets.

Blockchain technology applied to the Internet of Things (IoT) can protect transactions in the IoT, while blockchain deployment can eliminate central institutions in the IoT. C. Xu et al. [16] use cryptographic tools to design new consensus algorithms for the attack problems encountered in blockchain technology in IoT applications. The authors not only selected suitable methods to quantify the security protection level of the consensus algorithm, but also demonstrated the superiority of the new consensus algorithm in terms of security and scalability through extensive experimental evaluations [17].

Smart manufacturing is part of the smart city entity and the two collaborate to make processes more efficient through the use of technological information. Y. Teng et al. [18] have designed a generic configurable blockchain smart manufacturing system using the de-trusting features of blockchain. Using this system, complex manufacturing structures can be handled agreeably. At the same time, the authors addressed the efficiency and latency issues of blockchain in real-time manufacturing processes by formulating optimizations such as block size and task scheduling.

Blockchain technology has great potential for deployment in supply chain operations management because of its decentralized, transparent and tamper-evident properties. Any data in a blockchain-based supply chain system cannot be altered, enabling dynamic connectivity between network stakeholders and mitigating threats of security and fraud. R. W. Ahmad et al. [19] proposed a blockchain-based decentralized solution to the problem of methodological and technical inefficiencies, deficiencies in reliability, security and traceability of COVID-19 medical devices in terms of supply and subsequent production of waste disposal. The authors used the Ethernet blockchain and the InterPlanetary File

*Corresponding Author: Yuling Chen; E-mail: ylchen3@gzu.edu.cn
DOI: 10.53106/160792642023122407005

System to store and share data related to COVID-19 medical devices and developed algorithms to define interaction rules for COVID-19 waste disposal and establish a specification of penalties for those involved in non-compliance.

Applying blockchain technology to digital asset trading can solve the problems of tampering, forgery and profit distribution in the digital content trading environment. G. Heo et al. [20] proposed a blockchain based on secret blocks to address the problems of difficult digital content dissemination, limited capacity and privacy leakage encountered by blockchain technology in digital asset trading. This kind of blockchain is proposed to solve the current privacy and storage capacity limitation problems of blockchain technology in digital asset transactions. In this paper, the authors proposed a weight-based consensus algorithm for a reliable consensus in an environment where authenticated and non-authenticated users coexist.

It is thus clear that blockchain technology holds considerable promise for application in a wide range of industries. And specifically, blockchain is a new type of database that integrates various technologies such as distributed networks, encryption technology, and smart contracts. The ecological environment of the blockchain system is orderly [21]. In a nutshell, a blockchain is an immutable and unforgeable distributed ledger that combines data blocks in a sequential manner using a hash function by each node in chronological order. The data block contains the specific content of each transaction and various auxiliary information. Immutability, as one of the basic properties of blockchain, ensures that transactions added to historical blocks cannot be tampered, and is the cornerstone of blockchain security. Because any modification to a block will affect the related block and all subsequent blocks. Therefore, blocks confirmed by a series of subsequent blocks cannot be tampered. However, due to various needs of practical applications and the provisions of relevant laws, immutability has became an important obstacle to restrict the development of blockchain technology. In the following, we will describe the specific reasons that hinder the development of blockchain technology.

For example, some people use the immutability of blockchain technology to spread inappropriate content on the blockchain (eg: personal privacy, child pornography and illegal transactions, etc.). If the blocks on the chain contain illegal content, chain participants may inadvertently become disseminators of illegal content due to their inability to recognize the legitimacy of the transaction content. If chain participants can be identified, they will refuse to participate in and download the chain for fear of being accused of possessing illegal content.

In addition, according to the "right to be forgotten (RtbF)" [22] of the new European data protection regulations, data owners have the right to request that their private information be deleted from the Internet or search engines in order to protect their privacy from being violated. However, the immutability of the blockchain makes it impossible for messages that have been added to the blockchain to be modified or deleted. This feature violates the "right to be forgotten" of data regulations. Inevitably, various legal disagreements and incompatibilities have arisen between blockchain technology and data protection regulations. Therefore, it is necessary to edit the information on the blockchain in a controlled way. To that end, scholars have proposed methods and related technologies that can modify the blockchain to realize the editing, modification or deletion of transaction content while satisfying the security of the blockchain.

In our paper [23], we proposed an application of editable blockchain in the housing rental market, in which we hope the content that can be edited by the modifier to be restricted. At the same time, it is rare for modifiers to edit transactions throughout the housing rental process. Accordingly, in order to limit the malicious modification of users, we introduce revision control rules and related descriptions to limit the transaction content that users can modify and guide them to modify correctly. We introduce a one-time chameleon hash function to assign the modifier one modification permission, so as to avoid central authority assigning users the number of modifications arbitrarily. Specifically, this paper makes the following contributions.

1) Introduce revision control rules to restrict the modifiers from making arbitrary changes. Users specify the content of the transaction that modifiers can modify when they publish the transaction, that is, they introduce revision control rules to limit the content of the transaction that modifiers can rewrite. The modifiers can only modify the contents in the revision control rules. The contents outside the revision control rules are not allowed to be modified.
2) One time chameleon hash forbids the authority to grant modifiers any number of changing permissions. The modification authority of the modifier is granted by the certification authority. Central authority may arbitrarily grant modifiers the number of revisions, resulting in indiscriminate revisions. Therefore, we use a one-time chameleon hash so that the authority can only deliver the modifier to modify one time. If the modifier makes more than one modification or does not modify according to the relevant description of the revision control rules, the modifier's deposit will be withdrawn as a punishment for his malicious behavior.
3) In this paper, we present an instantiated construction of a constraints-based and one-time modification redactable blockchain ($\mathcal{CORB}$).

The structure of this paper is as follows: In section 2, we introduce the related work of this paper. Preliminary knowledge is introduced in the section 3. In section 4 we give the formal definition of a constraints-based and one-time modification redactable blockchain, the formal definition, the threat model and the instantiation of $\mathcal{CORB}$. Finally, we conclude the article in section 5.

2 Related Work

In 2017, G. Ateniese et al. [24] proposed the first editable blockchain based on a permissioned environment. This editable blockchain enables block-level rewriting in a controlled way by introducing chameleon hashes [25].

A chameleon hash is a cryptographic hash function that contains trapdoors, and users who hold the trapdoor can effectively generate collisions without changing the hash value. Therefore, the authors introduced the chameleon hash function to divide trapdoors to specific modifiers through certification authority. Deletion, modification, and insertion of blocks are performed by modifiers. The same year, I. Puddu et al. [26] proposed an editable blockchain called u-chain, which distributes keys among validators through dynamic secret sharing in a permissionless environment. However, there is a lot of performance overhead and multi-round interaction overhead between chain participants. Based on the problems of u-chain, D. Deuber et al. [27] proposed an editable blockchain that does not rely on heavy cryptography tools, which is also suitable for a permissionless environment. In this scheme, any user can make modification, but only edition voted by the majority are approved. Therefore, any modification operation on the chain can be publicly verified with minimal overhead for this step. However, this scheme assumed that most miners are honest and rational when voting. Dishonest and irrational miners may refuse to verify. Moreover, any party on the chain can make edit operations and there may be a denial of service attack. In 2020, the blockchain-based IoT system by G. Yu [28] and others introduced chameleon hash and attribute-based encryption to achieve fine-grained access control for attribute updates. The proposal of this scheme aims to solve the incompatibility problem between the immutability of the blockchain and the property revocation of ABE. In this paper, the data on the history chain can only be accessed by new members after the attribute is updated and cannot be accessed by members after the attribute is revoked.

To enable transaction-level blockchain rewriting in a permissioned environment in a controlled way, D. Derler et al. [29] proposed a solution based on chameleon-hashes with ephemeral trapdoors [30] and attribute-based encryption [31] in 2019. In this scheme, the authors proposed a new cryptographic primitive called policy-based chameleon hash (PCH). In PCH, only modifiers who know about long-term trapdoors and temporary trapdoors can modify transactions. Among them, the temporary trapdoor can only be obtained by the modifier whose attributes satisfy the editable transaction access structure, that is, only the trapdoor holder who meets the specific access structure can rewrite the transaction. However, this scheme requires the certification authority to be fully trusted. And a malicious central authority may do inappropriate behavior. Especially, certification authority is a very obvious target and disadvantage. Inspired by the PCH scheme proposed by D. Derler et al. In 2022, J. Ma et al. [32] proposed a new cryptographic primitive called Distributed Property-Based Chameleon Hash (DPCH) based on the PCH scheme to address the above problems. In this solution, where the authority of the modifier is issued by multiple authorities, there is no need for a fully trusted authority. At the same time, DPCH is resistant to collusion attacks between authorities and modifiers, and does not require any interaction between the various authorities.

In addition to the above-mentioned various schemes for editable blockchains, scholars have also worked on the number of modifications [33], accountability [34], and revocability [35] of redactable blockchains. S. Xu et al. [33] proposed k-time modifiable and epoch-based redactable blockchain (KERB) in 2022. This scheme rewrites the transaction by using the common chameleon hash, and the central authority decides the maximum number of modification k of the modifier according to the deposit submitted by the modifier. Trapdoors are exposed in KEBR and any party in the chain can modify the transaction content. However, only modifications made by modifiers authorized by the central authority signature can be verified by miners. Hence, the security of the scheme depends on the security of the underlying digital signature, which greatly reduces the overhead of the scheme. However, this scheme requires the central authority to be fully trusted, which hinders the application of KEBR.

Policy-Based chameleon hash is a useful tool for building redactable blockchains. However, the holder of the chameleon hash trapdoor may abuse the rewrite privilege to maliciously modify the transaction without being identified in the process of modifying the transaction. Thence, to hold accountable modifier who maliciously modify and abuse modification privileges, Y. Tian et al. [34] proposed a policy-based chameleon hash with black-box accountability (PCHBA) in 2020. Black-box accountability enables the attribute authority to link the modified transaction with the malicious modifier if malicious modification or abuse of rewriting privileges is detected by the modifier. Any party in the chain can then identify malicious transaction modifiers through black box interactions.

Users whose attributes in the PCH satisfy the access policy can obtain temporary trapdoors, and users can permanently access temporary trapdoors. However, in practical applications, we need to revoke the user's access rights so that they cannot obtain temporary trapdoors. Wherefore, to solve the above problems, Gaurav Panwar et al. [35] built a revocable and traceable blockchain rewrite scheme. In this scheme, the authors construct a revocable chameleon hash function with temporary trapdoors (RCHET) based on the ordinary chameleon hash function, and a revocable attribute-based encryption (RFAME) based on attribute-based encryption. Then, using the above methods and dynamic group signature, the user can immediately revoke the access of the authorized user to the temporary trapdoor as needed.

3 Preliminaries

In this section, we review access structures and several alternatives, including digital signatures, one-time chameleon hash and collision-resistant hash function.

3.1 Annotates

Let $\mathbb{N}$ denote the set of all natural numbers. We use the Greek letter λ for security parameters and capital letters such as A, B for algorithms. Unless otherwise stated, all algorithms need to run in probabilistic polynomial time (PPT), i.e. the running time is determined by a polynomial in their input length. Also, a special symbol $\perp$ is returned when the algorithm fails. If A is a probabilistic algorithm, then y

$\leftarrow A(x, r)$ represents x and random value r as the input of A, and calculates the output result y. We assume that 1^{λ} is the implicit input to all algorithms. For a function φ, we say that φ is negligible if $\forall\, k\, \exists\, \lambda_0\, \forall \lambda \geq \lambda_0$: $|\varphi(\lambda)| \leq 1/\lambda^k$.

3.2 Access Structure

Let $\mathbb{U}$ denote an attribute domain. A non-empty set $\mathbb{A} \in 2^{\mathbb{U}} \setminus \{\phi\}$ is an access structure on $\mathbb{U}$. A set in $\mathbb{A}$ is called an authorized set, and a set not in $\mathbb{A}$ is called an unauthorized set. A is said to be monotonic if $\forall B, C \in \mathbb{A}$ if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.

3.3 Digital Signature

The digital signature $\mathcal{DS}$ with message space $\mathcal{M}$ includes four algorithms $\{\mathcal{DS}\_Setup, \mathcal{DS}\_KeyGen, \mathcal{DS}\_Sign, \mathcal{DS}\_Verify\}$.

$\mathcal{DS}\_Setup(1^{\lambda}) \rightarrow pp$: Take a security parameter $\lambda \in \mathbb{N}$ as input and output a public parameter pp, where the public parameter pp is the implicit input of the other three algorithms.

$\mathcal{DS}\_KeyGen(pp) \rightarrow (pk, sk)$: Take the public parameter pp as input, and output a signing key pair (pk, sk), where pk is the signature public key and sk is the signature private key.

$\mathcal{DS}\_Sign(sk, m) \rightarrow \sigma$: Take a private key sk and a message $m \in \mathcal{M}$ as input, and outputs a signature σ.

$\mathcal{DS}\_Verify(pk, \sigma, m) \rightarrow b$: Taking the public key pk, a signature σ and the message $m \in \mathcal{M}$ as input, the output is a decision bit $b \in \{0,1\}$. If $b = 0$, the verification fails; if $b = 1$, the verification succeeds.

The correctness of digital signature requires that for all security parameters $\lambda \in \mathbb{N}$, for all public parameters pp $\leftarrow \mathcal{DS}\_Setup(1^{\lambda})$, for all $(pk, sk) \leftarrow \mathcal{DS}\_KeyGen(pp)$, for all messages $m \in \mathcal{M}$, We have $\mathcal{DS}\_Verify\ (pk, \mathcal{DS}\_Sign(sk, m), m) = 1$.

The EUF-CMA security of digital signatures is based on the following experiments.

$\text{Experiment } \text{Exp}_{\mathcal{DS},\mathcal{A}}^{EUF-CMA}(1^{\lambda})$

$p \leftarrow \mathcal{DS}.Setup(1^{\lambda}), \mathcal{L}_{key}, \mathcal{L}_{sign}, \mathcal{L}_{corr};$

$\mathcal{Q} \leftarrow \varnothing;$

(pk^*, m^*, σ^*)

$\leftarrow \mathcal{A}^{\mathcal{O}_{KeyGen}(\cdot), \mathcal{O}_{Sign}(\cdot), \mathcal{O}_{Corrupt}(\cdot)}(pp);$

where $\mathcal{O}_{KeyGen}(\tau)$:

$(sk, pk) \leftarrow \mathcal{DS}\_KeyGen(pp);$

$\mathcal{L}_{key} \leftarrow \mathcal{L}_{key} \cup \{(\tau, sk, pk)\};$

return pk;

and $\mathcal{O}_{Sign}(m)$:

$\sigma \leftarrow$

$\mathcal{DS}\_Sign(sk, m);$

$\mathcal{L}_{sign} \leftarrow \mathcal{L}_{sign} \cup \{m\};$

return σ;

and $\mathcal{O}_{Corrupt}(pk)$:

$\mathcal{L}_{corr} \leftarrow \mathcal{L}_{corr}\{pk\};$

return sk;

return 1 if $pk^* \notin \mathcal{L}_{corr} \wedge (pk^*, m^*) \notin \wedge\, \mathcal{L}_{sign} \wedge \mathcal{DS}_{Verify}(pk^*, m^*, \sigma^*) = 1$;

else return 0;

3.4 One-time Chameleon Hash Function

A one-time chameleon hash [36] with message space $\mathcal{M}$ consists of the following five algorithms $\{CH\_Setup, CH\_KeyGen, CH\_Hash, CH\_Verify, CH\_Adapt\}$.

$CH\_Setup(1^{\lambda}) \rightarrow pp$: Taking a security parameter $\lambda \in \mathbb{N}$ as input, it outputs a public parameter pp, where pp is the implicit input to the other four algorithms.

$CH\_KeyGen(pp) \rightarrow (sk, pk)$: Take the public parameter pp as input, and output the public-private key pair (sk, pk), where sk is the trapdoor private key and pk is the public key.

$CH\_Hash(pk, \tau, m) \rightarrow (h, r)$: Taking public key pk, label τ and message $m \in \mathcal{M}$ as input, output a hash value h and a random value r.

$CH\_Verify(pk, \tau, m, h, r) \rightarrow b$: Taking public key pk, label τ, message $m \in \mathcal{M}$, hash value h and random value r as output, output a decision bit $b \in \{0,1\}$. If $b = 0$, the verification fails; if $b = 1$, the verification passes.

$CH\_Adapt(sk, \tau, m, m', h, r) \rightarrow r'$: Input private key sk, label τ, message $m \in \mathcal{M}$, message $m' \in \mathcal{M}$, hash value h and random value r, and output another random value r'.

The correctness of one-time chameleon hash requires that for all security parameters $\lambda \in \mathbb{N}$, for all public parameters $pp \leftarrow CH\_Setup(1^{\lambda})$, for all $(sk, pk) \leftarrow CH\_KeyGen(pp)$, for all messages $m, m' \in \mathcal{M}$, for all $(h, r) \leftarrow CHET\_Hash(pk, \tau, m)$, for all $r' \leftarrow CH\_Adapt(sk, \tau, m, m', h, r)$, we have $H\_Verify(pk, \tau, m, h, r) = 1 \wedge CH\_Verify(pk, \tau, m', h, r') = 1$.

3.5 Collision-resistant Hash Function

A hash function that satisfies the following properties is called a collision-resistant hash function. For all probabilistic polynomial-time adversary $\mathcal{A}$, its probability of successfully finding a collision $\Pr(m_0, m_1) \leftarrow \mathcal{A}\ (1^{\lambda}, h)$: $m_0 \neq m_1 \wedge h(m_0) = h(m_1)$ is negligible.

4 Constraints-based and One-time Modification Redactable Blockchain (CORB)

4.1 Formal Definition

A constraints-based and one-time modification redactable blockchain involves four types of participating entities: certification authority (CA), user, modifier, and miner. It consists of the following eight algorithms:

$\mathcal{CORB}\_Setup(1^{\lambda}) \rightarrow (pp, msk, mpk)$: The probabilistic setting algorithm is run by the certification authority. It takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs a public parameter pp, a public-private key pair (msk, mpk), where msk is the master private key and mpk is the master public key. The public parameters pp and the master public key mpk in this algorithm are implicit inputs to other algorithms.

$\mathcal{CORB}\_Setup_u(pp) \rightarrow (sk_u, pk_u)$: The probabilistic user setting algorithm is run by each user. It takes the public parameter pp as input and outputs the user's public-private

key pair (sk_u, pk_u), where sk_u is the user's private key and pk_u is the user's public key.

$\mathcal{CORB}\_Setup_m(pp) \rightarrow (sk_m, pk_m)$: The probabilistic modifier setting algorithm is run by each modifier. It takes the public parameter pp as input, and outputs the modifier's public-private key pair (sk_m, pk_m), where sk_m is the modifier's private key, and pk_m is the modifier's public key.

$\mathcal{CORB}\_MAKGen(msk, pk_m, S) \rightarrow mak$: The probabilistic modifier authorized key generation algorithm is run by the certificate authority. It takes the master public key msk, the modifier public key pk_m and the attribute set S as input, and outputs a modifier authorization key mak.

$\mathcal{CORB}\_Hash(mpk, sk_u, (\tau, tx_\tau, ADM, desc), \mathbb{A}) \rightarrow (h, r, \sigma_\tau)$: Probabilistic hashing algorithms are run by user. He takes the master public key mpk, the user's private key sk_u, a message including transaction label τ, transaction content tx_τ, revision control rule ADM and related description $desc$, and an access structure $\mathbb{A}$ as input. Output hash value h, random value r and signature σ_τ, where ADM is a collection of subsets that can be modified in the transaction content tx_τ, which is called revision control rules. And $desc$ is the explanation and description of ADM given by the user.

$\mathcal{CORB}\_Adapt(mpk, sk_m, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau) \rightarrow (\tau, tx_\tau', ADM, desc), (r', \sigma_\tau')$: Probabilistic adaptive algorithms are run by modifiers. It takes the master public key mpk, the modifier's private key sk_m, a message including transaction label τ, transaction content tx_τ, revision control rule ADM and related description $desc$, hash value h, random value r, signature σ_τ and another message including transaction label τ, transaction content tx_τ', revision control rule ADM and related description $desc$ as input. Output another random value r' and signature σ_τ'.

$\mathcal{CORB}\_Verify(mpk, pk, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau) \rightarrow b$: Deterministic verification algorithms are run by chain participants. It takes the master public key mpk, the public key pk, a message containing the transaction label τ, the transaction content tx_τ, the revision control rule ADM and the related description $desc$, the hash value h, the random value r and the signature σ_τ as input. Output a decision bit b, where $b \in \{0,1\}$. If $b = 0$, it means that the transaction verification failed; if $b = 1$, it means that the transaction verification is successful. There are two types of public keys here: if the transaction has not been modified, the public key is pk_u; if the transaction has been modified, the public key is (pk_u, pk_m, mak).

$\mathcal{CORB}\_Extract(pk_m, (\tau, tx_\tau, ADM, desc), (\tau, tx_\tau', ADM, desc), \sigma_\tau, \sigma_\tau') \rightarrow sk_m$: Determinisic extraction algorithms are run by CA. It takes modifier's public key pk_m, one contains the transaction label τ, the transaction content tx_τ, the revision control rule ADM and a message describing the $desc$, and the other contains the transaction tag τ', the transaction content tx_τ', the revision control rule ADM, the associated message describing $desc$ and two signatures (σ_τ, σ_τ') as input. Output the modifier's signature private key sk_m.

4.2 System Model

Our constraints-based and one-time modification redactable blockchain consists of four participating entities, namely the certification authority, the user, the modifier, and the miner, as shown in Figure 1 below. The specific description is as follows.

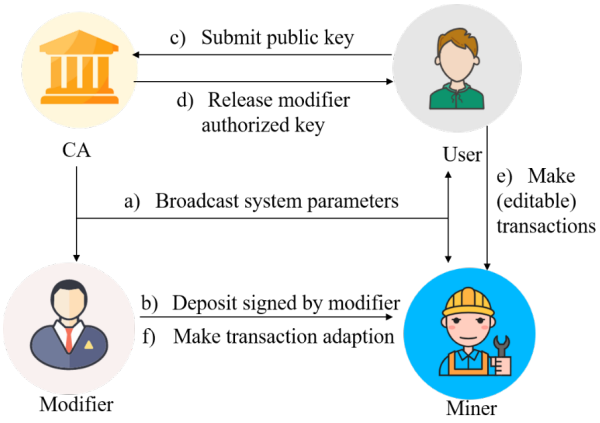

Figure 1. System model

The certification authority is the administrator of the blockchain and has the following responsibilities.

1) The certification authority initializes the system parameters of $\mathcal{CORB}$ and broadcasts the relevant parameters of the system to other parties (see a)).
2) After receiving the authorization key request from the modifier (see c)), the certification authority checks the signature of the modifier's deposit (see b)) and issues the authorization key for the modifier (see d)).

Users are allowed to release two types of transactions on-chain. One is an immutable transaction that cannot be modified; the other is a mutable transaction that can be modified by authorized modifiers (see e)).

Modifiers are chain participants who are issued modification privileges by the certified authority. Modifiers are required to pay a certain amount of deposit when they receive modification privileges. If the modified transaction passes the verification, the modifier's deposit will be returned; if the modifier is found to maliciously modify the transaction content, the modifier will lose his deposit. Miners validate transactions and add validated transactions to the global public ledger (see e), f)). In the ledger, blocks are fixed by miners and connected end to end to form a chain.

The workflow of $\mathcal{CORB}$ consists of five steps: system initialization, transaction generation, mutable transaction rewriting, transaction verification, and malicious punishment.

4.2.1 System Initialization

Figure 2 below shows the specific steps of system initialization. This stage can be divided into three steps: system parameter initialization (see 11), user parameter initialization and modifier parameter initialization (see 22 33 44).

System parameter initialization: CA generates public parameters pp and master public key mpk by running the $\mathcal{CORB}\_Setup(1^\lambda)$ algorithm, and broadcasts pp and mpk to other parties on the chain (see 11).

User parameter initialization: After receives the public parameter pp and the master public key mpk from the CA, each user run the modifier setting algorithm $\mathcal{CORB}\_Setup_u(pp)$ to generate its own public-private key pair (sk_u, pk_u) (see 55).

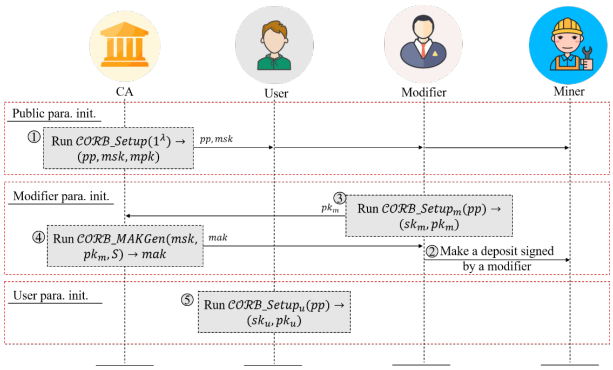

Figure 2. System initialization of $\mathcal{CORB}$

Modifier parameter initialization: After each modifier receives the public parameter *pp* and the master public key *mpk* from the CA, he runs the modifier setting algorithm $\mathcal{CORB}\_Setup_m$ (*pp*) to generate his own public-private key pair (sk_m, pk_m). Then, the modifier submits a deposit after signing sk_m with the modifier's private key. If the modifier has malicious modifications or more than one modification, the deposit can be extracted through using the CA and the modifier's private key (see 22). After that, the modifier sends his public key pk_m to the CA (see 33). The CA specifies the modifier's attribute set *S* and runs the modifier authorization key generation algorithm $\mathcal{CORB}\_MAKGen$ (*msk*, pk_m, *S*) to generate the modifier authorization key *mak* to the modifier (see 44).

4.2.2 Transaction Generation

Figure 3 below shows the specific steps of transaction generation. As mentioned above, users can generate two types of transactions: immutable transactions and mutable transactions, where variable transactions can be modified by authorized modifiers. Therefore, this phase can be divided into two steps: immutable transaction generation and mutable transaction generation.

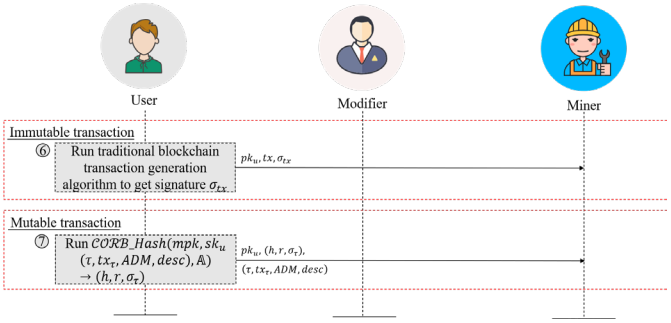

Figure 3. Transaction generation of $\mathcal{CORB}$

Immutable transaction generation: Users are allowed to generate immutable transactions in traditional blockchains. The user generates transaction *tx* and signature σ_{tx} using his own private key sk_u and traditional hash function, and broadcasts his own public key pk_u transaction *tx* and signature σ_{tx} to the blockchain system (see 66).

Mutable transaction generation: The user is allowed to generate mutable transactions, that is, the user runs the hash algorithm $\mathcal{CORB}\_Hash$(*mpk*, sk_u, (τ, tx_τ, *ADM*, *desc*), $\mathbb{A}$) to generate a hash value *h*, a Random value *r* and a signature σ_τ. Then, the user broadcasts the public key pk_u, the transaction (τ, tx_τ, *ADM*, *desc*) with the label τ and the signature (*h*, *r*, σ_τ) to the blockchain system (see 77).

4.2.3 Mutable Transaction Rewriting

Figure 4 below shows the specific steps of mutable transaction modification. As mentioned above, after receiving a variable transaction, authorized modifiers rewriting the mutable transaction.

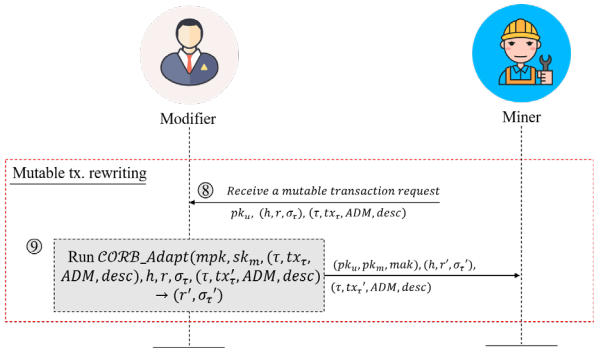

Figure 4. Mutable transaction rewriting

Mutable transaction rewriting: When the transaction modifier receives a variable transaction modification request (see 88), the authorized modifier checks whether its attribute set *S* satisfies the access structure $\mathbb{A}$. If the attribute set *S* satisfies the access structure $\mathbb{A}$, the modifier runs the adaptive

algorithm ($\mathcal{CORB}\_Adapt(mpk, sk_m, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau, (\tau, tx_\tau', ADM, desc))$ to generate a random value r' and signature σ_τ'. Then, the modifier broadcast the public key $pk = (pk_u, pk_m, mak)$, the modified transaction $(\tau, tx_\tau', ADM, desc)$ with the label τ and signature (h, r', σ_τ') to the blockchain system (see 99).

4.2.4 Transaction Verification

Transaction verification is divided into immutable transaction verification, mutable transaction verification and transaction rewrite verification. Figure 5 below shows the specific steps of immutable transaction verification.

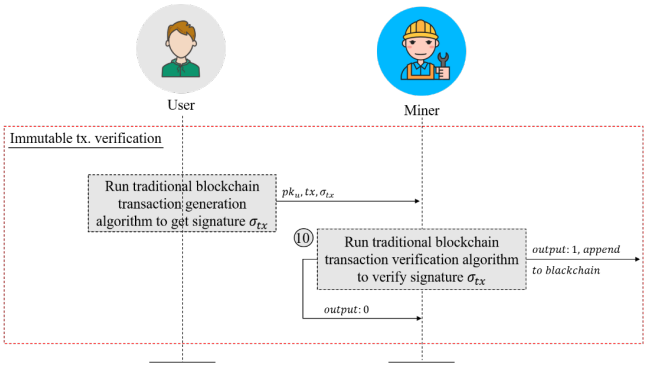

Figure 5. Immutable transaction verification

After receiving an immutable transaction whose public key is pk_u, the transaction content is tx, and the transaction signature is σ_{tx}, the miner runs the traditional immutable blockchain verification algorithm to verify the signature σ_{tx}. If the output of the algorithm is 1, the miner adds the transaction to the blockchain. Otherwise, the miner refuses to add the transaction (1010).

Figure 6 below shows the specific steps of variable transaction verification.

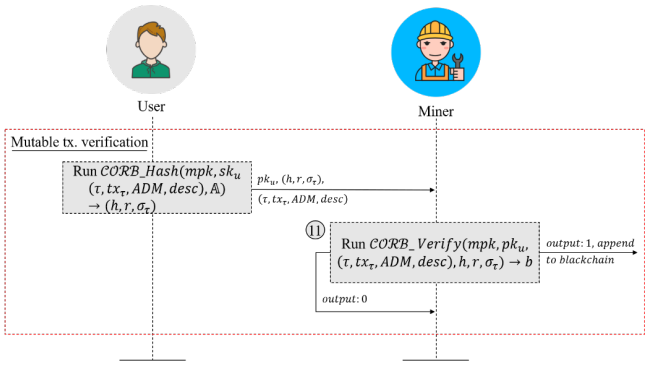

Figure 6. Mutable transaction verification

After receiving a mutable transaction whose public key is pk_u , the transaction is $(\tau, tx_\tau, ADM, desc)$ and the signature is (h, r, σ_τ), the miner runs the $\mathcal{CORB}$ verification algorithm $\mathcal{CORB}\_Verify(mpk, pk, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau)$ verify the mutable transaction. If the algorithm output is 1, the miner adds the transaction to the blockchain. Otherwise, the miner refuses to add the transaction (see 1111).

Figure 7 below shows the specific steps of transaction rewrite verification.

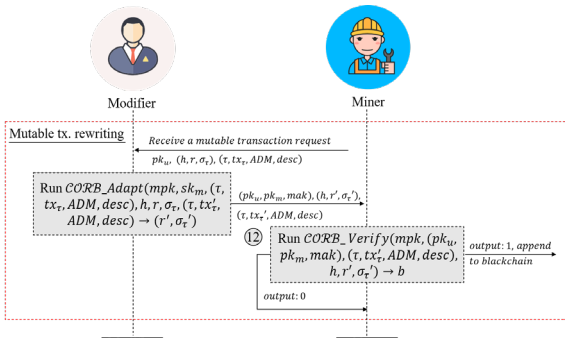

Figure 7. Transaction rewrite verification

After receiving a rewriting transaction whose public key is (pk_u, pk_m, mak), the transaction is $(\tau, tx_\tau', ADM, desc)$ and the signature is (h, r', σ_τ'), the miner run CORB verification algorithm $\mathcal{CORB}\_Verify(mpk, pk, (\tau, tx_\tau', ADM, desc), h, r', \sigma_\tau')$ to verify the rewriting transaction. If the output of the algorithm is 1, the miner adds the rewrite transaction to the blockchain. Otherwise, the miner refuses to add the rewrite transaction (see 1212).

4.2.5 Malicious Punishment

Figure 8 below shows the specific steps of malicious punishment. This stage can be divided into two steps: signature key extraction and malicious punishment.

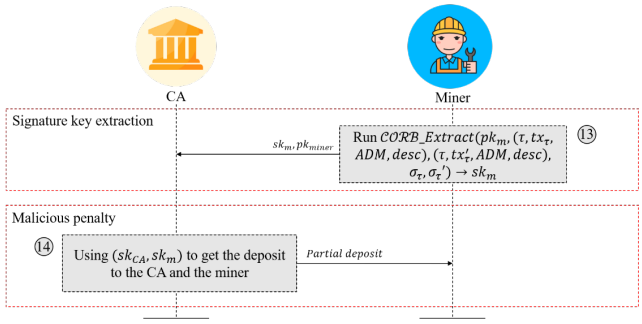

Figure 8. Malicious punishment

Signature key extraction: If the authorized modifier violates the revision control rule ADM given by the user or there is more than one modification, the miner runs the key extraction algorithm $\mathcal{CORB}\_Extract(pk_m, (\tau, tx_\tau, ADM, desc), (\tau, tx_\tau', ADM, desc), \sigma_\tau, \sigma_\tau')$ to obtain the modifier's signature key sk_m. The miner then sends the obtained signature key sk_m and his own public key pk_{miner} to the CA.

Malicious punishment: After the CA receives the malicious modifier's signature key sk_m and the miner's public key pk_{miner}, the CA uses its own private key sk_{CA} and the malicious modifier's private key sk_m to obtain the modifier's deposit. Then, the CA distributes part of the deposit to the miners as a reward.

4.3 Threat Model

In our threat model, we assume that the authority (CA) is fully trusted, i.e. the CA is honest. In a blockchain system, the number of modifiers is only a small fraction. Therefore, we assume that modifiers and miners are mostly honest. However, there are a small number of dishonest modifiers

and miners in the system. They may get together to launch a collusion attack, that is, they would rewrite an unauthorized transaction without the authorization key or make more than one modification without any penalty. In our scenario, modifications made by unauthorized users cannot pass validation. Therefore, if the CA's private key is kept secret, our scheme is secure. We give users the right to rewrite transactions through digital signatures by authoritative organizations. Unauthorized users cannot effectively edit transactions. Therefore, the security of our scheme relies on the security of the underlying digital signatures. Here we give a formal proof of the EUF-CMA security of the scheme, as shown in Figure 9 below.

Let the modifier oracle be $\mathcal{O}_{Setup_m}(\cdot)$, modifier corruption oracle be $\mathcal{O}_{Corrupt_m}(\cdot)$, user oracle be $\mathcal{O}_{Setup_u}(\cdot)$, user corruption oracle be $\mathcal{O}_{Corrupt_u}(\cdot)$, modifier authorized key oracle be $\mathcal{O}_{MAKGen}(\cdot,\cdot)$, hash oracle be $\mathcal{O}_{Hash}(\cdot,\cdot,\cdot)$, adaptive oracle be $\mathcal{O}_{Adapt}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$.

$\mathrm{Exp}^{EUF-CMA}_{CORB,\mathcal{A}}(1^\lambda)$
$(pp, msk, mpk) \leftarrow CORB\_Setup(1^\lambda), \mathcal{L}_{s_m}, \mathcal{L}_{Corr_m}, \mathcal{L}_{s_u}, \mathcal{L}_{Corr_u}, \mathcal{L}_{MAK}, \mathcal{L}_H, \mathcal{L}_{Apt};$
$(pk^*, (\tau^*, tx_{\tau^*}, ADM^*, desc^*), \mathbb{A}^*, h^*, r^*, \sigma_{\tau^*}) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, mpk);$
where $\mathcal{O}_{Setup_m}(\tau)$:
$(sk_m, pk_m) \leftarrow CORB\_Setup_m(pp);$
$\mathcal{L}_{s_m} \leftarrow \mathcal{L}_{s_m} \cup \{(\tau, sk_m, pk_m)\};$
return pk_m;
and $\mathcal{O}_{Corrupt_m}(pk_m)$:
$\mathcal{L}_{Corr_m} \leftarrow \mathcal{L}_{Corr_m} \cup \{pk_m\};$
return sk_m;
and $\mathcal{O}_{Setup_u}(\tau)$:
$(sk_u, pk_u) \leftarrow CORB\_Setup_u(pp);$
$\mathcal{L}_{s_u} \leftarrow \mathcal{L}_{s_u} \cup \{(\tau, sk_u, pk_u)\};$
return pk_u;
and $\mathcal{O}_{Corrupt_u}(pk_u)$:
$\mathcal{L}_{Corr_u} \leftarrow \mathcal{L}_{Corr_u} \cup \{pk_u\};$
return sk_u;
and $\mathcal{O}_{MAKGen}(pk_m, S)$:
$mak \leftarrow CORB\_MAKGen(msk, pk_m, S);$
$\mathcal{L}_{MAK} \leftarrow \mathcal{L}_{MAK} \cup \{(pk_m, S)\};$
return mak;
and $\mathcal{O}_{Hash}(sk_u, (\tau, tx_\tau, ADM, desc), \mathbb{A})$:
$(h, r, \sigma_\tau) \leftarrow CORB\_Hash(mpk, sk_u, (\tau, tx_\tau, ADM, desc), \mathbb{A});$
$\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{(pk_u, (\tau, tx_\tau, ADM, desc), \mathbb{A}, h, r, \sigma_\tau)\};$
return$(h, r, \sigma_\tau);$
and $\mathcal{O}_{Adapt}(sk_m, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau, (\tau, tx_\tau', ADM, desc))$:
$(r', \sigma_\tau') \leftarrow CORB\_Adapt(mpk, sk_m, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau, (\tau, tx_\tau', ADM, desc));$
$\mathcal{L}_{Apt} \leftarrow \mathcal{L}_{Apt} \cup \{(pk_m, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau, (\tau, tx_\tau', ADM, desc))\};$
return$(r', \sigma_\tau');$
return 1, if
$CORB\_Verify(mpk, pk^*, (\tau^*, tx_{\tau^*}, ADM^*, desc^*), h^*, r^*, \sigma_{\tau^*}) = 1 \wedge$
$(pk^* = pk_u^* \wedge pk_u^* \notin \mathcal{L}_{Corr_u} \wedge (pk_u^*, (\tau^*, tx_{\tau^*}, ADM^*, desc^*), \mathbb{A}^*, h^*, r^*, \sigma_{\tau^*}) \notin \mathcal{L}_H) \vee$
$(pk^* = (pk_u^*, pk_m^*, mak^*) \wedge (pk_m^* \notin \mathcal{L}_{Corr_m} \vee mak^* \notin \mathcal{L}_{MAK}) \wedge$
$(pk_m^*, (\tau^*, tx_{\tau^*}, ADM^*, desc^*), (\cdot,\cdot,\cdot,\cdot), h^*, \cdot, r^*, \cdot, \sigma_{\tau^*}) \notin \mathcal{L}_H$

Figure 9. The EUF-CMA safety experiment of $\mathcal{CORB}$

4.4 An Instantiation of CORB

In this scheme, we use EUF-CMA secure digital signature $\mathcal{DS}$ = {$\mathcal{DS}$_*Setup*, $\mathcal{DS}$_*KeyGen*, $\mathcal{DS}$_*Sign*, $\mathcal{DS}$_*Verify*} and a collision-resistant one-time chameleon hash *OCH* = {*CH*_*Setup*, *CH*_*KeyGen*, *CH*_*Hash*, *CH*_*Adapt*} to instantiate our $\mathcal{CORB}$. The specific instantiation process is as follows.

$\mathcal{CORB}$_*Setup*$(1^\lambda) \rightarrow$ (*pp*, *msk*, *mpk*): Run the digital signature parameter initialization algorithm $\mathcal{DS}$_*Setup*$(1^\lambda) \rightarrow pp$ to generate the digital signature parameters $pp_{\mathcal{DS}}$. Run the key generation algorithm $\mathcal{DS}$_*KeyGen*$(pp) \rightarrow$ (*pk*, *sk*) to obtain the digital signature key pair (sk_{ca}, pk_{ca}). Choose a collision-resistant one-time chameleon hash function H:$\{0,1\}^* \rightarrow \mathbb{Z}_p$. The parameter $pp_{\mathcal{CH}}$ of the one-time chameleon hash is generated by running the one-time chameleon hash parameter initialization algorithm $CH_{Setup(1^\lambda)} \rightarrow pp$. Select two large prime numbers p and q of the same length, and calculate $N = pq$. Choose a sufficiently large prime number e and compute $ed = 1 \bmod (p-1)(q-1)$. Randomly select $x_0 \in \mathbb{Z}_N$ and calculate $X_0 = x_0^e \bmod N$. Then, choose two secure hash functions: H_e: $\{0,1\}^* \rightarrow \mathbb{Z}_e^*$ and H_N:$\{0,1\}^* \rightarrow \mathbb{Z}_N^*$. Run the key generation algorithm $CH_{KeyGen(pp)} \rightarrow$ (*sk*, *pk*) to get the public key $pk_{ch} = (N, e, X_0, H_e, H_N)$ and the private key $sk_{ch} = d$ for a chameleon hash. The algorithm returns public parameters $pp = (pp_{\mathcal{DS}}, pp_{\mathcal{CH}})$, master private key $msk = sk_{ca}$ and master public key $mpk = (sk_{ch}, pk_{ca}, pk_{ch}, H)$.

$\mathcal{CORB}$_$Setup_u(pp) \rightarrow (sk_u, pk_u)$: The user sets the algorithm by running the signature generation algorithm $\mathcal{DS}$_$KeyGen(pp_{\mathcal{DS}}) \rightarrow (sk_u, pk_u)$ to return the user's private key sk_u and public key pk_u.

$\mathcal{CORB}$_$Setup_m(pp) \rightarrow (sk_m, pk_m)$: The modifier sets the algorithm by running the digital signature key algorithm $\mathcal{DS}$_$KeyGen(pp_{\mathcal{DS}}) \rightarrow (sk_m', pk_m')$ to initialize the signature key pair (sk_m', pk_m'). Then, choose a random exponent $\alpha \in \mathbb{Z}_p$ and randomly choose terms $\beta, r \in \mathbb{Z}_p$, and calculate $c = g^\alpha$. Return the modifier's private key $sk_m = (sk'_m, \{\beta, r\})$ and public key $pk_m = (pk'_m, c)$.

$\mathcal{CORB}$_$MAKGen(msk, pk_m, S) \rightarrow mak$: The modifier authorization key algorithm returns $\perp$ when the deposit signed by the modifier has not been received, otherwise the authority CA runs the digital signature algorithm $\mathcal{DS}$_$Sign(sk_{ca}, (pk_m, S)) \rightarrow mak$ to get the authorization key *mak*.

$\mathcal{CORB}$_$Hash(mpk, sk_u, (\tau, tx_\tau, ADM, desc), \mathbb{A}) \rightarrow (h, r, \sigma_\tau)$: The hash algorithm runs the one-time chameleon hash algorithm $CH\_Hash(pk, \tau, m) \rightarrow (h, r)$ to calculate the hash value $h = \overline{r}^e X_0^{H_e((\tau, tx_\tau, ADM, desc), \hat{r}, 0)} X_1^{H_e((\tau, tx_\tau, ADM, desc), \hat{r}, 1)} \bmod N$ and random value $r = (\overline{r}, \hat{r})$, where $X_1 \leftarrow H_N(\tau)$, $\overline{r} \leftarrow \mathbb{Z}_N^*$, $\hat{r} \leftarrow \{0,1\}^k$. Then, the access policy $\mathbb{A}$ is defined by generating the signature $\mathcal{DS}$_$Sign(sk_u, (\tau, ADM, desc, r, \mathbb{A})) \rightarrow \sigma_\tau$. In our scheme, our hash value h and signature σ_τ are linked by transaction label τ. Moreover, our signature σ_τ contains revision control rules *ADM* and related description *desc*, but does not contain transaction specific information tx_τ to prevent any party from deducing previous transaction information. The algorithm returns the hash value h, the random value r and the signature σ_τ.

$\mathcal{CORB}$_$Adapt(mpk, sk_m, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau, (\tau, tx_\tau', ADM, desc)) \rightarrow (r', \sigma_\tau')$: The adaptive algorithm is adapted by running the one-time chameleon hash adaptive algorithm $CH\_Adapt(sk_{ch}, (\tau, tx_\tau, ADM, desc), h, r, (\tau, tx_\tau', ADM, desc)) \rightarrow r'$ to generate another random value $r' = \overline{r}(X_0^{H_e((\tau, tx_\tau, ADM, desc), \hat{r}, 0) - H_e((\tau, tx_{\tau'}, ADM, desc), \hat{r}, 0)} \times X_1^{H_e((\tau, tx_\tau, ADM, desc), \hat{r}, 1) - H_e((\tau, tx_{\tau'}, ADM, desc), \hat{r}, 1)}) \bmod N$. Then, through the digital signature algorithm $\mathcal{DS}$_$Sign(sk'_m, (tx'_\tau, r', \mathbb{A})) \rightarrow \sigma'$ to generate another signature σ_τ'. The algorithm returns another random value r' and another signature $\sigma_\tau' = (\sigma_\tau', z)$, where $z = \beta \cdot H(\tau, r', \mathbb{A}) + sk_m'$.

$\mathcal{CORB}$_$Verify(mpk, pk, (\tau, tx_\tau, ADM, desc), h, r, \sigma_\tau) \rightarrow b$: If the transaction is not rewrite, that is, $r = r'$ and the public key $pk = pk_u$. If $\mathcal{DS}$_$Verify(pk_u, \sigma_\tau, \tau, r, \mathbb{A}) = 1 \wedge CH_Verify(pk_{ch}, (\tau, tx_\tau, ADM, desc), h, r) = 1$, the algorithm returns 1. Otherwise, the algorithm returns 0. If the transaction is edited, that is, $r \neq r'$ and the public key $pk = (pk_u, pk_m, mak)$. If $\mathcal{S}$_$Verify(pk_u, \sigma_\tau, \tau, r, \mathbb{A}) = 1 \wedge CH_Verify(pk_{ch}, (\tau, tx_\tau, ADM, desc), h, r) = 1 \wedge \mathcal{DS}$_$Verify(pk_m', \sigma_\tau', \tau, r', \mathbb{A}) = 1 \wedge \mathcal{DS}$_$Verify(pk_{ca}, mak, (pk_m, S)) = 1 \wedge S \vDash \mathbb{A}$, then the algorithm returns 1. Otherwise, the algorithm returns 0.

$\mathcal{CORB}$_*Extract* $(pk_m, (\tau, tx_\tau, ADM, desc), (\tau, tx_\tau', ADM,$

desc), σ_τ, σ_τ') $\rightarrow sk_m$: If $\tau = \tau' \wedge (\tau, r, \mathbb{A}) = (\tau, r', \mathbb{A}) \wedge \mathcal{DS}\_Verify(pk_m', \sigma_\tau, \tau, r, \mathbb{A}) = 1 \wedge \mathcal{DS}\_Verify(pk_m', \sigma_\tau', \tau, r', \mathbb{A}) = 1$, then the key extraction algorithm returns $pk_m' = z \cdot H(\tau', r', \mathbb{A}') - z' \cdot H(\tau, r, \mathbb{A})/ H(\tau', r', \mathbb{A}') \cdot H(\tau, r, \mathbb{A})$. Otherwise, the algorithm returns the failure symbol $\perp$.

5 Conclusion

In this paper, we proposed a constraints-based and one-time modification redactable blockchain, which aims to limit the malicious modification of the transaction content by the modifier and the malicious authorization of the authority. We prohibit the modifiers from making arbitrary changes to the content of the transaction through the revision control rules and related description. By introducing a one-time chameleon hash function, the authority issued by the authority to the modifier for any number of modifications is limited. In future work, we will consider a set of authorities to issue modification authority to avoid corruption of a single authority.

Acknowledgements

This research is funded by the National Natural Science Foundation (12061020, 61962009).

This work is supported by the Top Technology Talent Project from Guizhou Education Department (Qian jiao ji [2022] 073).

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008, *https://bitcoin.org/bitcoin.pdf*.

[2] T. Li, Y. Chen, Y. Wang, Y. Wang, M. Zhao, H. Zhu, Y. Tian, X. Yu, Y. Yang, Rational protocols and Attacks in blockchain system, *Security and Communication Networks*, Vol. 2020, pp. 1-11, September, 2020.

[3] T. Li, Z. Wang, G. Yang, Y. Cui, Y. Chen, X. Yu, Semi-selfish mining based on hidden Markov decision process, *International Journal of Intelligent Systems*, Vol. 36, No. 7, pp. 3596-3612, July, 2021.

[4] X. Wang, Z. Zhou, X. Luo, Y. Xu, Y. Bai, F. Luo, A Blockchain-Based fine-grained access data control scheme with attribute change function, *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation*, Atlanta, GA, USA, 2021, pp. 348-356.

[5] M. Du, Q. Chen, J. Xiao, H. Yang, X. Ma, Supply chain finance innovation using blockchain, *IEEE transactions on Engineering Management*, Vol. 67, No. 4, pp. 1045-1058, November, 2020.

[6] A. Basumatary, S. Joshi, Adoption of blockchain in trade finance and its impact on financial decision making, *2022 International Conference on Decision Aid Sciences and Applications*, Chiangrai, Thailand, 2022, pp. 556-559.

[7] G. Shi, H. Hao, J. Lei, Y. Zhu, Application security system design of Internet of Things based on blockchain technology, *2021 International Conference on Computer, Internet of Things and Control Engineering*, Guangzhou, China, 2021, pp. 134-137.

[8] Y. Chen, J. Sun, Y. Yang, T. Li, X. Niu, H. Zhou, PSSPR: A source location privacy protection scheme based on sector phantom routing in WSNs, *International Journal of Intelligent Systems*, Vol. 37, No. 2, pp. 1204-1221, February, 2022.

[9] Surjandy, E. Fernando, Meyliana, R. Kosala, H. L. H. S. Warnars, E. Abdurach, S. H. Supangkat, Success factors of the blockchain adoption for smart manufacture, *2018 International Seminar on Research of Information Technology and Intelligent Systems*, Yogyakarta, Indonesia, 2018, pp. 617-621.

[10] C. K. M. Lee, Y. Z. Huo, S. Z. Zhang, K. K. H. Ng, Design of a Smart Manufacturing system with the application of multi-access edge computing and blockchain technology, *IEEE Access*, Vol. 8, pp. 28659-28667, February, 2020.

[11] S. Esfandiari, The effect of blockchain technology on supply chain management: its potential to prevent fraud and reduce risks to food safety and its effects on the relationships between supply chain actors in the Mexican food processing industry, *2022 IEEE Technology and Engineering Management Conference*, Izmir, Turkey, 2022, pp. 179-183.

[12] I. A. Omar, R. Jayaraman, M. S. Debe, H. R. Hasan, K. Salah, M. Omar, Supply chain inventory sharing using ethereum blockchain and smart contracts, *IEEE Access*, Vol. 10, pp. 2345-2356, December, 2021.

[13] T. Chomsiri, D. Pansa, JSP digital asset trading system, *2019 23rd International Computer Science and Engineering Conference*, Phuket, Thailand, 2019, pp. 255-260.

[14] G. Heo, D. Yang, I. Doh, K. Chae, Design of blockchain system for protection of personal information in digital content trading environment, *2020 International Conference on Information Networking*, Barcelona, Spain, 2020, pp. 152-157.

[15] Y. Zhu, Research on digital finance based on blockchain technology, *2021 International Conference on Computer, Blockchain and Financial Development*, Nanjing, China, 2021, pp. 410-414.

[16] C. Xu, Y. Qu, T. H. Luan, P. W. Eklund, Y. Xiang, L. Gao, A lightweight and attack-proof bidirectional blockchain paradigm for internet of things, *IEEE Internet of Things Journal*, Vol. 9, No. 6, pp. 4371-4384, March, 2022.

[17] L. Cotugno, F. Mazzenga, A. Vizzarri, R. Giuliano. The major opportunities of Blockchain for Automotive Industry: a Review, *2021 AEIT International Conference on Electrical and Electronic Technologies for Automotive*, Torino, Italy, 2021, pp. 1-6.

[18] Y. Teng, L. Li, L. Song, F. R. Yu, V. C. M. Leung, profit maximizing smart manufacturing over AI-Enabled configurable blockchains, *IEEE Internet of Things Journal*, Vol. 9, No. 1, pp. 346-358, January, 2022.

[19] R. W. Ahmad, K. Salah, R. Jayaraman, I. Yaqoob, M. Omar, S. Ellahham, Blockchain-based forward supply

chain and waste management for COVID-19 medical equipment and supplies, *IEEE Access*, Vol. 9, pp. 44905-44927, March, 2021.
[20] G. Heo, D. Yang, I. Doh, K. Chae, Efficient and secure blockchain system for digital content trading, *IEEE Access*, Vol. 9, pp. 77438-77450, May, 2021.
[21] T. Li, Z. Wang, Y. Chen, C. Li, Y. Jia, Y. Yang, Is semi-selfish mining available without being detected? *International Journal of Intelligent Systems*, Vol. 37, No. 12, pp. 10576-10597, December, 2022.
[22] E. Politou, E. Alepis, C. Patsakis, Forgetting personal data and revoking consent under the GDPR: challenges and proposed solutions, *Journal of Cybersecurity*, Vol. 4, No. 1, pp. 1-20, March, 2018.
[23] C. Wang, W. Jia, Y. Chen, Housing rental scheme based on redactable blockchain, *Wireless Communications and Mobile Computing*, Vol. 2022, pp. 1-10, March, 2022.
[24] G. Ateniese, B. Magri, D. Venturi, E. Andrade, Redactable block-chain–or–rewriting history in bitcoin and friends, *Proc. 2017 IEEE European Symposium on Security and Privacy*, Paris, France, 2017, pp. 111-126.
[25] H. Krawczyk, T. Rabin, Chameleon hashing and signatures, *IACR Cryptology ePrint Archive*, Article ID. 1998/010, 1998.
[26] I. Puddu, A. Dmitrienko, S. Capkun, uchain: How to forget without hard forks, *IACR Cryptology ePrint Archive*, Article ID. 2017/106, 2017.
[27] D. Deuber, B. Magri, S. A. K. Thyagarajan, Redactable blockchain in the permissionless setting, *2019 IEEE Symposium on Security and Privacy*, Francisco, CA, USA, 2019, pp. 124-138.
[28] G. Yu, X. Zha, X. Wang, W. Ni, K. Yu, P. Yu, J. A. Zhang, R. P. Liu, Y. J. Guo, Enabling attribute revocation for fine-grained access control in blockchain-IoT systems, *IEEE Transactions on Engineering Management*, Vol. 67, No. 4, pp. 1213-1230, November, 2020.
[29] D. Derler, K. Samelin, D. Slamanig, C. Striecks, Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based, *The Network and Distributed System Security Symposium*, San Diego, California, USA, 2019, pp. 1-15.
[30] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, D. Slamanig, Chameleon hashes with ephemeral trapdoors-and applications to invisible sanitizable signatures, *20th IACR International Conference on Practice and Theory in Public-Key Cryptography*, Amsterdam, The Netherlands, 2017, pp. 152-182.
[31] V. Goyal, O.t Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, *Proceedings of the 13th ACM conference on Computer and Communications Security*, Alexandria, VA, USA, 2006, pp. 89-98.
[32] J. Ma, S. Xu, J. Ning, X. Huang, R. H. Deng, Redactable blockchain in decentralized setting, *IEEE Transactions on Information Forensics and Security*, Vol. 17, pp. 1227-1242, March, 2022.
[33] S. Xu, J. Ning, J. Ma, X. Huang, R. H. Deng, K-time modifiable and epoch-based redactable blockchain, *IEEE Transactions on Information Forensics and Security*, Vol. 16, pp. 4507-4520, August, 2021.
[34] Y. Tian, N. Li, Y. Li, P. Szalachowski, J. Zhou, Policy-based chameleon hash for blockchain rewriting with black-box accountability, *Annual Computer Security Applications Conference*, Austin, TX, USA, 2020, pp. 813-828.
[35] G. Panwar, R. Vishwanathan, S. Misra, ReTRACe: Revocable and traceable blockchain rewrites using attribute-based cryptosystems, *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, Virtual Event, Spain, 2021, pp. 103-114.
[36] W. Gao, L. Chen, C. Tang, G. Zhang, F. Li, One-time chameleon hash function and it application in redactable blockchain, *Journal of Computer Research and Development*, Vol. 58, No. 10, pp. 2310-2318, October, 2021.

Biographies

Chunli Wang is currently studying for a master's degree in the School of Mathematics and Statistics at Guizhou University in China. Her research interests include applied statistics and blockchain technology.

Yuling Chen is a Professor and Ph.D. adviser. Her current research interests include cryptography, big data security and privacy protection, blockchain technology, etc.

Wensheng Jia received his M.S. degree and Ph.D. degree from Department of Mathematics, Guizhou University, China. His research interest covers immune optimization, evolutionary computation deep learning and game theary.