An Anonymous Editable Blockchain Scheme Based on Certificateless Aggregate Signature

Kemeng Li
School of Cyberspace Security, the National Engineering Laboratory for Wireless Security
Xi'an University of Posts and Telecommunications
Xi'an 710121, China
likemeng2019@163.com

Dong Zheng
School of Cyberspace Security, the National Engineering Laboratory for Wireless Security
Xi'an University of Posts and Telecommunications
Xi'an 710121, China
zhengdong@xupt.edu.cn

Rui Guo
School of Cyberspace Security, the National Engineering Laboratory for Wireless Security
Xi'an University of Posts and Telecommunications
Xi'an 710121, China
guorui@xupt.edu.cn

***Abstract*—Blockchain technology has gradually replaced traditional centralized data storage methods, and provided people reliable data storage services with its decentralized and non-tamperable features. However, the current blockchain data supervision is insufficient and the data cannot be modified once it is on the blockchain, which will cause the blockchain system to face various problems such as illegal information cannot be deleted and breach of smart contract cannot be fixed in time. To address these issues, we propose an anonymous editable blockchain scheme based on the reconstruction of the blockchain structure of the SpaceMint combining with the certificateless aggregate signature algorithm. Users register with their real identities and use pseudonyms in the system to achieve their anonymity. If the number of users who agree to edit meets the threshold, the data on the blockchain can be modified or deleted, and our scheme has the function of accountability for malicious behavior. The security analysis show that the proposed certificateless aggregate signature algorithm enjoys the unforgeability under the adaptive selected message attack. Moreover, the method of setting the threshold of related users is adopted to guarantee the effectiveness and security of editing blockchain data. At last, we evaluate the performance of our certificateless aggregate signature algorithm and related schemes in theoretical analysis and experimental simulation, which demonstrates our scheme is feasible and efficient in storage, bandwidth and computational cost.**

Keywords—editable blockchain, certificateless aggregate signature, accountability, anonymity

I. Introduction

Blockchain is a new decentralized infrastructure and distributed computing paradigm [1], and it has been widely employed due to its decentralized, non-tamperable, and traceable features, combined with the internet of things, 5G communications, and cloud storage. With the development of smart contracts, blockchain has gradually realized the leap from the financial sector to medical, political, cultural, entertainment and other fields, promoting the diversified development of various fields.

With the continuous expansion of the application field of blockchain technology, people have gradually realized that while ensuring the security of user data, it has also caused a large amount of illegal and false information to be stored in the blockchain system. Take the Bitcoin[2] system as an example, users can conduct transactions without registering, which has better anonymity. In addition, the lack of supervision of data on the chain makes it impossible to trace and accountable for malicious behavior [3], which provides a breeding ground for illegal and criminal behavior. In addition, the blockchain technology uses a block-and-chain data storage structure. So the transaction information, smart contracts, electronic certificates and other data cannot be modified once they are released. In 2016, in the Ethereum system[4], THE DAO[5] smart contract had serious program breach, but it could not be fixed in time resulting in huge economic losses and waste of resources [6].

In order to prevent false information and contract vulnerabilities from causing huge losses to users and systems, editable blockchain has become a hot topic for scholars in recent years. At present, the editable blockchain solution provides a way to modify and delete data on the chain. However, there are still problems such as trapdoor leaks and lack of authentication of the data on the chain. Most solutions are based on a proof of work (POW)[7], which consumes more computing resources. In 2017, Park et al. proposed the SpaceMint blockchain system[8] based on the proof of space (POSpace). The nodes compete for mining based on recyclable local space, which has low computational cost and avoids the waste of resources caused by competition in computing power. At the same time, the secure proof mechanism guarantees the firmness of its credit, and realizes the balance between security and resource efficiency. In addition, the SpaceMint system adopts a new chain structure. The traditional blockchain uses a hash function to combine the block header, block body, and front and back. SpaceMint breaks the direct link between the block header and the block body by adding signature sub-blocks to build a new block link structure, which provides a new direction for blockchain data editing. In the section of related work, we provide a sufficient and detailed review of the state-of-the-art research progress in the field of editing blockchain.

A. Our Contribution

In this paper, we propose an editable blockchain scheme by reconstructing the blockchain structure and combining the certificateless aggregation signature algorithm to address the problem of uneditable blockchain data. Our main contributions are summarized as follows:

1) We restructured the structure of SpaceMint to meet the needs of editable blockchains while maintaining the original linkability of blocks.

2) We propose an anonymous certificateless aggregate signature algorithm with a fixed signature length, which can perform one-time verification of the signatures of many users instead of one by one.

3) Combining with the above two points, we propose an editable blockchain scheme. When the number of users who agree to edit meets the threshold, the data on the blockchain can be modified or deleted. Moreover, it has the function of holding accountable for illegal data.

4) We formally prove that the proposed certificateless aggregate signature algorithm is unforgeable to two types of adversaries in the random oracle model under the Computational Diffie-Hellman(CDH) assumption.

5) We simulated the proposed certificateless aggregate signature algorithm through JPBC and compared it with other schemes. The results show that this scheme has higher execution efficiency.

B. Related Work

In 2017, based on the chameleon hash function[9], Ateniese et al.[10] first proposed a scheme to modify the blockchain. Using the characteristics of the chameleon hash function, the purpose of modifying the block data to ensure that the hash value remains unchanged, and then achieving the editable function of the blockchain. In 2018, Li et al.[11] designed the chameleon hash function so that every node in each blockchain system masters the trapdoor, and only one random node can complete the modification. In 2019, Derler et al.[12] combined attribute encryption to set the access of the chameleon hash function trapdoor, so that only people who meet specific attributes can access the trapdoor and modify the blockchain data. However, all of the above solutions have the problem that the adversary can calculate the trap door through a collision, and there is a potential security risk of the trap door leakage. Marsalek et al.[13] used the double-chain model to propose a correctable blockchain scheme. Deuber et al.[14] designed a double-hash chain structure. The second chain formed is still valid by retaining a copy of the original Merkle tree. However, the double-chain mode is expensive and not suitable for practical use. Ren et al.[15] proposed a deleteable blockchain scheme based on a new block structure of space proof combined with threshold ring signatures. In Ren et al.'s scheme, the location of the original transaction sub-block is replaced by the result of threshold ring signature to delete the original content, but his scheme does not have the function of modifying the blockchain, and if the number of system users' changes, the original threshold is insufficient to represent a reasonable threshold. After that, Cai et al.[16] proposed an privacy-protected deletable blockchain with linkable multi-signature, which achieves the deletion of blockchain data and protects user's identity privacy by using their one-time addresses as pseudonyms. However, Cai et al.'s scheme lacks supervision of data uploading to the blockchain.

Aiming at the problems of the above scheme, we propose an anonymous editable blockchain scheme based on the improvement of the blockchain structure of the space proof mechanism and using certificateless aggregate signature algorithm. The certificateless public key cryptosystem[17] was first proposed by Riyami and Paterson in 2003. In that system, the key generation center only generates and masters the user's partial private key, and its complete private key is generated by combining the user's own secret information and the partial private key. In 2015, Chen et al.[18] proposed an efficient aggregate signature scheme that resists two types of adversary attacks. However, due to its large calculation overhead, it cannot meet the data verification of a large number of users. Malhi and Batra [19] proposed a certificateless aggregate signature scheme suitable for vehicular ad-hoc networks. Hang et al.[20] proposed a reattack of a certificateless aggregate signature and gave the relevant security certification. Li et al.[21] proposed an improved certificateless aggregate signature algorithm and analyzed the security of the scheme when the key generator center is malicious but passive. In 2018, Zhang et al.[22] used aggregate signatures to achieve aggregate authentication of identity data and reduce verification and communication overheads, but they did not provide security proofs for the second type of adversary. Liu et al.[23] proposed an anonymous certificateless aggregate signature scheme, but the binding relationship between the public key and the partial private key is not reflected in the signature, so the scheme can be attacked by malicious participants. In 2020, Thumbur et al.[24] presented a new certificateless signature scheme and claimed it was secure. Unfortunately, Xu et al.[25] pointed Thumbur's scheme is vulnerable to signature forgery attack and cannot achieve its stated purpose and proposed a new certificateless signature scheme without pairing operations and formally prove its security. After that, Liu et al.[26] proposed an improved security certififficateless aggregate signature and meet security requirements of Gayathri et al.'s scheme[27]. However, Zhan et al.[28] presented an attack algorithm to show that Liu et al.'s scheme cannot resist the forgery attack launched by the attackers who only can obtain public information. The certificateless aggregate signature algorithm proposed in this paper has unforgeable security properties under adaptive selection message attacks, and has higher computational efficiency than other schemes. According to the characteristics of certificateless aggregate signature algorithm, it can ensure user anonymity and improve the efficiency of user information verification, while the blockchain data can be effectively and legally deleted, modified, and accountable combining with the reconstructed blockchain structure.

II. PRELIMINARIES

A. Bilinear Pair

Suppose G and G_T are two cyclic groups with prime order p, and P is a generator of G. A bilinear map $e: G \times G \to G_T$ satisfies the following properties:

1) Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$, where $P, Q \in G$, $a, b \in Z_p^*$.

2) Non-Degeneracy: $e(P,Q) \neq 1_{G_T}$, where 1_{G_T} is the unit element of G_T.

3) Computability: For any $P,Q \in G$, there is an efficient algorithm to compute $e(P,Q)$.

B. Computational Problem and Assumption

Computation Diffie-Hellman (CDH) Problem: Given a tuple $(P, aP, bP) \in G$, compute $abP \in G$. An algorithm $\mathcal{A}$ is said to solve the CDH problem in G with an advantage ε, if $\Pr\left[\mathcal{A}(P, aP, bP) = abP\right] \geq \varepsilon$, where the probability is over the random choice of $P \in G$, $a,b \in Z_p^*$, and the random coins used by $\mathcal{A}$.

CDH Assumption: There is no probabilistic polynomial-time algorithm $\mathcal{A}$ to slove CDH successfully with a non-negligible advantage ε.

C. Algorithm Model

The certificateless aggregate signature algorithm consists of seven parts including **Setup**, **UserKeyGen**, **PartialKeyGen**, **Signing**, **Verify**, **Aggregate** and **AggregateVerify**, which are defined as follows:

Setup$(1^{\lambda}) \to (s, params)$: On input a security parameter λ, it output the master private key s and the public parameter $params$.

UserKeyGen$(params) \to (x_i, X_i)$: On input the public parameter $params$, it outputs the user private key x_i and public key X_i.

PartialKeyGen$(params, s, X_i, ID_i^{'}) \to R_i$: On input public parameter $params$, master private key s, user public key X_i and user's pseudonym $ID_i^{'}$, it ouput the partial private key R_i.

Signing$(params, m_i, x_i, X_i, ID_i^{'}, R_i) \to \sigma_i$: On input the public parameter $params$, message m_i, private key x_i and the partial private key R_i, it outputs the signature σ_i of message m_i.

Verify$(params, m_i, \sigma_i) \to (true / false)$: On input the public parameter $params$, message m_i, the signature σ_i and the user's public X_i as input. If the signature σ_i is valid signature of the m_i, it outputs $true$. Otherwise, outputs $false$.

Aggregate$\left(params, \{m_i, \sigma_i\}_{1 \leq i \leq n}\right) \to \sigma_{Agg}$: On input the public parameter $params$ and the messages and signatures pair $\{m_i, \sigma_i\}_{1 \leq i \leq n}$ from n users as input, it outputs the aggregate signature σ_{Agg} of messages $\{m_i\}_{1 \leq i \leq n}$.

AggregateVerify$\left(\text{params}, \{m_i\}_{1 \leq i \leq n}, \sigma_{Agg}\right) \to (true / false)$: On input the public parameter $params$ and the aggregate signature σ_{Agg} as input. If the σ_{Agg} is valid, it outputs $true$. Otherwise, outputs $false$.

D. Security Definition

Definition 1: The proposed certificateless aggregate signature scheme is unforgeable under the selected message attack. If the adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ can win with a non-negligible probability in **Game1** and **Game2**, the unforgeability of this scheme under adaptive selection message attack is security.

Game1:

Init: Challenger C performs the Setup algorithm and generates the system master private key s and the system public parameter $params$. Finally, challenger sends the $params$ to the adversary $\mathcal{A}_1$.

Hash_Queries: For the hash function used in the system, the adversary sends any message to the challenger C, and the challenger returns the hash value random generated.

SecretKey_Queries: When adversary $\mathcal{A}_1$ asks for the private key of user ID_i, challenger C returns the corresponding private key x_i.

PartialSecretKey_Queries: When the adversary $\mathcal{A}_1$ asks for the partial key of the user ID_i, challenger C runs the PartialKeyGen algorithm and returns the user's partial private key R_i.

PublicKey_Queries: When the adversary $\mathcal{A}_1$ asks the public key of the user ID_i, it returns the corresponding public key X_i.

ReplacePublicKey_Queries: In the public key replacement inquiry phase, the adversary $\mathcal{A}_1$ replaces the public key X_i of the user ID_i with the new public key $X_i^{'}$.

Signing_Queries: The adversary $\mathcal{A}_1$ sends user ID_i, message m_i and public key X_i. If the user ID_i public key X_i is not replaced, return the valid signature σ_i corresponding to X_i. If the public key of user ID_i is replaced with $X_i^{'}$, the signature $\sigma_i^{'}$ will be returned.

Forgery: The aggregate signature σ^* output by the adversary $\mathcal{A}_1$ can pass the AggregateVerify algorithm, where σ^* is the aggregate signature corresponding to $\{ID_i^*, X_i^*, m_i^*\}(1 \le i \le n)$.

Adversary $\mathcal{A}_1$ will win the game if the following conditions are met:

1) For user $ID_i(1 \le i \le n)$, the *PartialSecretKey_Queries* has not performed at least one user ID_j to obtain the user's partial private key.
2) The has not queried the signature of (ID^*, m^*).

Game2:

Init: Challenger C performs the Setup algorithm and generates the system master private key s and the system public parameter *params*. Finally, C sends the *params* and the system master private key s to the adversary $\mathcal{A}_2$.

In **Game2**, adversary $\mathcal{A}_2$ performs *SecretKey_Queries* , *PublicKey_Queries* and *Signing_Queries*.The three-stage is similar to each stage in **Game1**.

Forgery: The aggregate signature σ^* output by the adversary $\mathcal{A}_2$ can pass the AggregateVerify algorithm, where σ^* is the aggregate signature corresponding to $\{ID_i^*, X_i^*, m_i^*\}(1 \le i \le n)$.

Adversary $\mathcal{A}_2$ will win the game if the following conditions are met:

1) For user $ID_i(1 \le i \le n)$, the *SecretKey_Queries* has not performed at least one user ID_j to obtain the user's private key.
2) The has not queried the signature of (ID^*, m^*).

E. Blockchain Structure of SpaceMint

The node with the largest storage space is regarded as the winning mining node in SpaceMint. The winning node obtains the accounting rights of the block, and stores the corresponding space proof information, transaction information, and the signature of the winning node on the transaction information in the block. The link between the corresponding blocks before and after is guaranteed by signatures.

As shown in Figure 1, each block is divided into three sub-blocks, namely the proof sub-block φ_i , the signature sub-block θ_i and the transaction sub-block τ_i. Therefore, the block can be defined as $B_i = \{\varphi_i, \theta_i, \tau_i\}$.

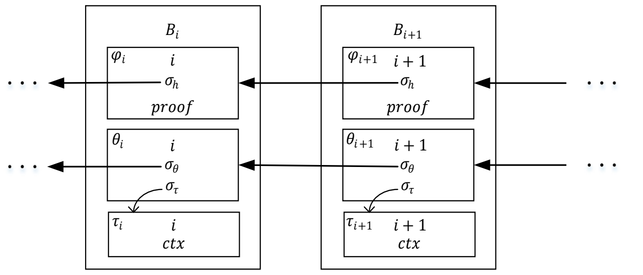

Fig. 1. The chain structure of SpaceMint.

Proof sub-block φ_i: it contains the current block number i, the miner's signature σ_h on the previous proof sub-block φ_{i-1}, and the *proof* of space proof when the miner compete for the accounting right.

Signature sub-block θ_i: it contains the current block number i, the miner's signature σ_τ on the transaction sub-block τ_i, and the miner's signature σ_θ on the previous signature sub-block θ_{i-1}.

Transaction sub-block τ_i: it contains the current block number i and transaction set ctx .

III. The Structure of Editable Blockchain

A. The Structure of Reconstructed Blockchain

This paper reconstructs the structure of the SpaceMint, while maintaining the original linkability and meets the need for modification in the future. As shown in Figure 2, this scheme adds σ_{copy} field, which is a copy of the signature σ_τ of original data, to the signature sub-block, so that the signature sub-block is reconstructed from the original $\theta_i = \{i, \sigma_\theta, \sigma_\tau\}$ to $\theta_i = \{i, \sigma_\theta, \sigma_\tau, \sigma_{copy}\}$. In addition, σ_θ is still the miner's signature on $(i-1, \{\sigma_\theta, \sigma_\tau\}_{i-1})$ in the previous signature sub-block, which means that the content of σ_θ does not contain σ_{copy}.

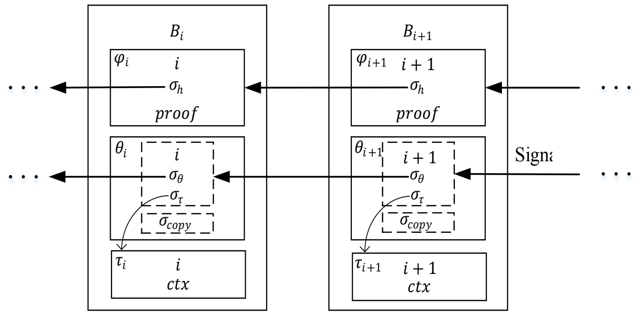

Fig. 2. The structure of reconstructed blockchain.

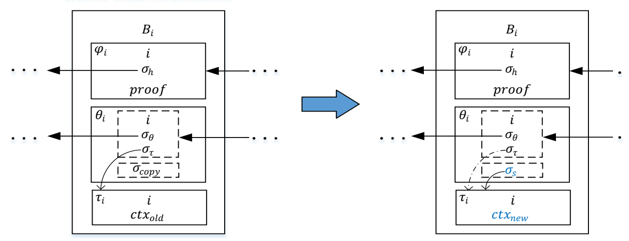

Fig. 3. Comparison before and after data modification.

As shown in Figure 3, if the original data ctx_{old} of the transaction sub-block τ_i in block i is modified to the new data ctx_{new}, where ctx_{new} is the modified new data or the reason for deleting ctx_{old}, then σ_τ is not a valid signature of ctx_{new}.

Therefore, we propose to replace σ_{copy} field with the smart contract's signature σ_s on ctx_{new} after the data is modified, so that users can verify ctx_{new} through the contract signature σ_s, while keeping the original data signature σ_τ unchanged. Note that trigger the smart contract to execute the signing algorithm to generate the contract signature σ_s on ctx_{new}, only when the number of users who agree to the modification meets the threshold, so that the contract signature σ_s has validity and legitimacy. Since σ_τ in the modified signature sub-block remains unchanged, σ_θ in the next signature sub-block is still unchanged and the signature sub-block of the next block and the signature sub-block of the modified block maintain the original linkability. Finally, during the entire data modification process, the hash sub-block φ_i is not changed, so the linkability of the entire block is not destroyed.

B. The Specific Description of Editable Blockchain

Based on the reconstruction of the signature chain, this scheme consists of four parts: System establishment, User registration, Upload data to blockchain and Blockchain editing. The system process is shown in Figure 4.

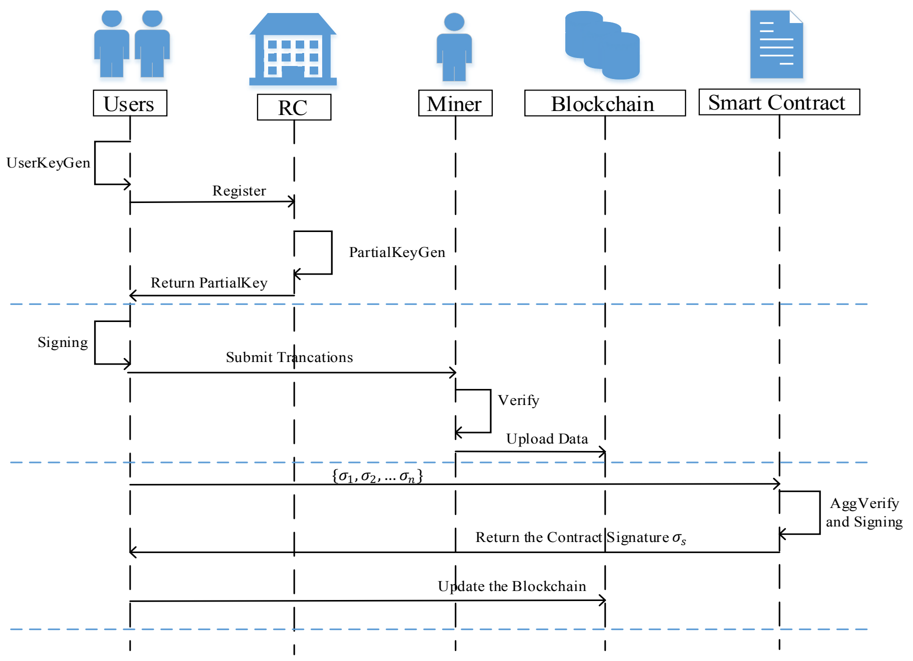

Fig.4. The system framework of editable blockchain

This system mainly consists of five entities: Users, Registration center, Miner, Blockchain, and Smart contract.

a) Users: Each user of the system first generates a user-side private key, and obtains partialprivate key after registering with a real identity, then user can publish transaction information and submit data in the blockchain system.

b) Registration center RC: RC generates a pseudonym and partial private key for users according to the real identity of users, and records the number of real-time registered users in the system.

c) Miner: Miner check the validity of the signature on the transaction information submitted by the system user. If the signature is valid, user's transaction information will be packaged and signed by miner and uploaded to the blockchain.

d) Blockchain: Decentralized distributed storage systemmaintained by all users.

e) Smart contract: The automatically executable code deployed by the authoritative center. When the number of users in the system that agree to the block modification request meet the threshold number, the smart contract will aggregate-verify the signature of the request information from each agreeing user. After the verification is passed, the smart contract generates the signature of new data and broadcasts it. Finally, each user performs block updates after receiving the signature from the smart contract.

1) System establishment

RC chooses the security parameter λ , and a secure symmetric encryption schemes (ENC, DEC), and then runs the **Setup** algorithm to output the master private key s which is kept secretly by RC and public parameters $params$.

Setup$\left(1^{\lambda}\right) \to \left(s, params\right)$: RC chooses the bilinear mapping $e: G \times G \to G_T$, where G and G_T are the additive cyclic group and multiplicative cyclic group with the same order p, respectively. Then, chooses the hash function $H: G \to \{0,1\}^*$, $H_1: G \times G \to G$, $H_2: \{0,1\}^* \times G \to G$ and $H_3: \{0,1\}^* \to Z_p^*$. Finally, randomly chooses $s \in_R Z_p^*$ as the system master private key, and computes $P_{pub} = sP$, $Q = H_1\left(P, P_{pub}\right)$, where P is a generator of G , $params = \left\{P, P_{pub}, Q, G, G_T, H_1, H_2, H_3, p\right\}$ are the system public parameters.

2) User registration

a) User-Side: User performs the **UserKeyGen** algorithm as follows, and computes $C_i = ENC\left(Key_i, ID_i\right)$, then sends C_i and X_i to RC, where ID_i is the only real identity of user.

UserKeyGen$\left(params\right) \to \left(x_i, X_i\right)$: It first randomly chooses $x_i \in_R Z_p^*$, then computes the public key $X_i = x_i P$ and the symmetric encryption key $Key_i = x_i P_{pub}$.

b) RC-Side: After receiving the information C_i and X_i from the user, RC first computes $Key_i = sX_i$, then it performs the decryption algorithm $ID_i = DEC\left(Key_i, C_i\right)$. If the user's ID_i has not been registered, RC generates user's pseudonym $ID_i' = ID_i \oplus H\left(sX_i\right)$ and performs **PartialKeyGen** algorithm as follows:

PartialKeyGen$\left(params, s, X_i, ID_i'\right) \to R_i$: RC first computes $I_i = H_2\left(ID_i', X_i\right)$ and $R_i = sI_i$. Then, computes $C_{Ri} = ENC\left(K_i, \left(R_i, I_i\right)\right)$ and sends C_{Ri} to user. Finally, RC updates the number N of user registrations.

c) User-Side: After receiving the C_{Ri} from RC, user first computes $\left(R_i, I_i\right) = DEC\left(Key_i, C_{Ri}\right)$, then verifies that if $e\left(R_i, P\right) = e\left(I_i, P_{pub}\right)$, R_i is kept secretly by user and the registration is successful.

3) Upload data to blockchain

a) User-Side: User signs the transaction data to ensure the legitimacy of the data and the follow-up accountability of the data by performing the **Signing** algorithm as follows:

Signing$\left(params, m_i, x_i, X_i, ID_i', R_i\right) \to \sigma_i$: First it randomly chooses $\alpha_i \in_R Z_p^*$, then computes $U_i = \alpha_i I_i$ and $h_i = H_3\left(ID_i', m_i, X_i, U_i, P_{pub}\right)$, where m_i is user's data. Finally computes $V_i = x_i h_i Q + \left(\alpha_i + h_i\right) R_i$ and broadcast the signature $\sigma_i = \left(V_i, U_i, X_i\right)$.

b) Miner-Side: Miner first checks the validity of the signature on user's data by performing the **Verify** algorithm as below. If the algorithm outputs $true$, which means that the user is a registered user, the verified data will be uploaded the blockchain by miner.

Verify$\left(params, m_i, \sigma_i\right) \to \left(true / false\right)$: First it computes $h_i' = H_3\left(ID_i', m_i, X_i, U_i, P_{pub}\right)$ and $I_i' = H_2\left(ID_i', X_i\right)$, if $e\left(V_i, P\right) = e\left(U_i + h_i' I_i', P_{pub}\right) e\left(h_i' Q, X_i\right)$, outputs $true$. Otherwise outputs $false$.

c) RC-Side: For illegal data existing in the blockchain, RC can restore the real identity of the data owner through the pseudonym ID_i' and the public key X_i . For $ID_i = ID_i' \oplus H\left(sX_i\right)$, only RC can compute $H\left(sX_i\right)$ to obtain the user's real identity ID_i.

4) Blockchain editing

a) User-Side: Each user in the system that agree to blockchain data modification and deletion generates the signature of the request information $request = (index, ctx_{old}, ctx_{new}, T_i)$ by performing the **Signing** algorithm in section 3.2.3, where $index$ is the block number of the modified block, ctx_{old} is the original data in the transaction block, ctx_{new} is the pre-updated new data or the reason for deleting the old data, and T_i is the current timestamp. Then each user sends their own signature $\sigma_i = (V_i, U_i, X_i)$ of $request$ to the smart contract.

b) Smart Contract-Side: The smart contract initial signature list $List$ is empty. After receiving signatures $\sigma_i = (V_i, U_i, X_i)$ of the $request$ from users, if the $List$ already contains the signature about X_i, RC keeps the list unchanged, otherwise adds σ_i to the $List$. After the response time is over, if $l / N_T > \mu$ (l is the $List$ length, μ is the system preset threshold, and N_T is the number of system users at time T), which means the number of agreed users meets the threshold, the smart contract performs the following **Aggregate** algorithm and **AggregateVerify** algorithm. If it returns $true$, the smart contract performs the **Signing** algorithm in section 3.2.3 to generate the contract signature σ_s of $m = (index, ctx_{new}, T_i)$ then to broadcast σ_s and m , where the key generation of smart contract is the same as the user registration stage.

$\textbf{Aggregate}\left(\boldsymbol{params}, \{request, \boldsymbol{\sigma_i}\}_{1 \le i \le t}\right) \rightarrow \boldsymbol{\sigma_{Agg}}$: For $1 \le i \le t$, the smart contract computes $H_3(ID_i^{'}, request, X_i, U_i, P_{pub})$, $I_i = H_2(ID_i^{'}, X_i)$, and $\sigma_{Agg} = (U, V, X, I)$, where $V = \sum_{i=1}^{t} V_i$, $U = \sum_{i=1}^{t} U_i$, $X = \sum_{i=1}^{t} h_i X_i$, $I = \sum_{i=1}^{t} h_i I_i$.

$\textbf{AggregateVerify}\left(\textbf{params}, \boldsymbol{\sigma_{Agg}}\right) \rightarrow \left(\boldsymbol{true} \,/\, \boldsymbol{false}\right)$: If $e(V, P) = e(U + I, P_{pub})e(Q, X)$, it returns $true$, otherwise return $false$.

c) User-Side: After each user receives the signature σ_s of the smart contract, performs the **Verify** algorithm in section 3.2.3 to check the validity of σ_s . If σ_s is a valid signature of $m = (index, ctx_{new}, T_i)$ from smart contract, each user updates the block B_{index} . As shown in Figure 3 in section 3.1, the original data ctx_{old} of the transaction sub-block τ_{index} is modified to ctx_{new} , and the σ_{copy} field in the signature sub-block is replaced with the contract signature σ_s . When each user of the system subsequently accesses the ctx_{new} , only needs to check the validity of σ_s to ensure the legality of ctx_{new} .

IV. SECURITY ANALYSIS

Theorem 1: Under the CDH assumption, the proposed certificateless aggregate signature scheme is unforgeable against adaptively chosen-message attacks to adversary $\mathcal{A}_1$ in the random oracle model.

Proof : Suppose a set of random CDH instances (P, aP, bP) , where $a, b \in Z_p^*$, $P \in G$, set $aP = A$ and $bP = B$. Assuming that adversary $\mathcal{A}_1$ breaks Game1 with advantage ε_1 in t_1 time, challenger C can have advantage $\varepsilon \ge \frac{\varepsilon_1}{(q_{ps}+1)e}$, time $t \le t_1 + \mathcal{O}(q_h + q_{us} + q_{ps} + q_{pk} + q_{sig} + q_{as})T_m$ to solve the CDH problem, which means it has computed abP . where q_h is the number of queries for Hash. q_{us} is the number of queries for SecretKey. q_{ps} is the number of queries for PartialSecretKey. q_{pk} is the number of queries for PublicKey. q_{sig} is the number of queries for Signing. q_{as} is the number of queries for AggSigning. T_m is the calculation time of scalar multiplication in group G .

Init : C sets $P_{pub} = aP = A$, $Q = H_1(P, P_{pub})$, then sends the public parameters $params = \{P, P_{pub}, Q, G, G_T, H_1, H_2, H_3, p\}$ to $\mathcal{A}_1$, where H_1 , H_2 and H_3 are the hash functions inquired.

Hash_Queries : C saves three lists L_1 , L_2 and L_3 . The queries corresponding to the three hash functions H_1, H_2 and H_3 are respectively stored.

H_1 _ Queries : Initial $L_1=\left(P,P_{pub},r_Q,Q\right)$ is empty. Challenger C receives $\left(P,P_{pub}\right)$ query for H_1 from adversary $\mathcal{A}_1$. If the query has ever been made, challenger C returns the same value. Otherwise, C selects the random value $r_Q\in_R Z_p^*$, return $Q=r_QP$, and adds $\left(P,P_{pub},r_Q,Q\right)$ to L_1.

H_2 _ Queries : Initial $L_2=\left(ID_i,X_i,c_i,r_i,I_i\right)$ is empty. C receives $\left(ID_i,X_i\right)$ query for H_2 from $\mathcal{A}_1$. If the query has ever been made, C returns the same value. Otherwise, C flips a coin $c_i\in\{0,1\}$ and chooses $r_i\in_R Z_p^*$, where $\Pr\left[c_i=0\right]=\mu$. If $c_i=0$, returns $I_i=r_iB$. Otherwise, returns $I_i=r_iP$ and adds $\left(ID_i,X_i,c_i,r_i,I_i\right)$ to L_2.

H_3 _ Queries : Initial $L_3=\left(m_i,X_i,ID_i,U_i,h_i\right)$ is empty. C receives $\left(ID_i,x_i,R_i,X_i\right)$ query for H_3 from $\mathcal{A}_1$. If the query has ever been made, C returns the same value. Otherwise, C returns $h_i\in_R Z_p^*$ and adds $\left(m_i,X_i,ID_i,U_i,h_i\right)$ to L_3.

SecretKey _ Queries : Initial $L=\left(ID_i,x_i,R_i,X_i\right)$ is empty. C receives $\left(ID_i,x_i,R_i,X_i\right)$ query from $\mathcal{A}_1$ .If $\left(ID_i,x_i,R_i,X_i\right)$ is included in L, returns x_i . Otherwise, C chooses $x_i\in_R Z_p^*$ and sets $X_i=x_iP$, then returns x_i and adds $\left(ID_i,x_i,R_i,X_i\right)$ to L.

PartialSecretKey _ Queries : C receives $\left(ID_i,X_i\right)$ query from $\mathcal{A}_1$. C retrieves $\left(ID_i,X_i,c_i,r_i,I_i\right)$ from L_2. If $c_i=0$, C returns failure. Otherwise, C returns $R_i=r_iP_{pub}$ and adds $\left(ID_i,x_i,R_i,X_i\right)$ to L.

PublicKey _ Queries: C receives ID_i query from $\mathcal{A}_1$. C retrieves $\left(ID_i,X_i,c_i,r_i,I_i\right)$ from L_2 . If $c_i=0$, C updates the $\left(ID_i,x_i,R_i,X_i\right)$, sets $X_i=B$, $x_i=\perp$ and returns X_i . If $c_i=1$ and $X_i=\perp$, C chooses $x_i\in_R Z_p^*$ and returns $X_i=x_iP$. If $c_i=1$ and $X_i\neq\perp$, returns X_i.

Replace*PublicKey _ Queries*: $\mathcal{A}_1$ chooses a new public key $X_i^{'}$ of ID_i. If $\left(ID_i,x_i,R_i,X_i\right)$ is included in L, C sets $X_i=X_i^{'}$, $x_i=\perp$ then updates the L. Otherwise, C sets $X_i=X_i^{'}$, $x_i=\perp$, $R_i=\perp$ and adds $\left(ID_i,x_i,R_i,X_i\right)$ to L.

Signing _ Queries : C generates signature for $\mathcal{A}_1$, When the $\mathcal{A}_1$ make a signing query on $\left(m_i,ID_i,T_x\right)$. If $\left(ID_i,x_i,R_i,X_i\right)$ is not included in L and $X_i=\perp$, C gets the $\left(x_i,X_i\right)$ through *PublicKey _ Queries* of ID_i and obtains the $\left(ID_i,X_i,c_i,r_i,I_i\right)$ from L_2.

If $c_i=1$ and $x_i\neq\perp$, C chooses $\alpha_i\in_R Z_p^*$ and computes $U_i=\alpha_iI_i$, then generates $h_i=H_3\left(ID_i,m_i,X_i,U_i,P_{pub},T_x\right)$ through H_3 _ *Queries* . Finally, C sets $V_i=x_ih_iQ+\left(\alpha_i+h_i\right)R_i$ and returns $\sigma_i=\left(V_i,U_i,X_i\right)$, where Q is obtained from L_1. Obviously, the σ_i is valid as follows:

$$e\left(V_i,P\right)=e\left(x_ih_iQ+\left(\alpha_i+h_i\right)R_i,P\right)$$
$$=e\left(U_i+h_iI_i,P_{pub}\right)e\left(h_iQ,X_i\right)$$

Otherwise, C obtains the $\left(ID_i,X_i,c_i,r_i,I_i\right)$ for L_2 , where $I_i=r_iB$, then chooses $\alpha_i,h_i\in_R Z_p^*$ and computes $U_i=\alpha_iP-r_ih_iX_i$, Denote $H_3\left(ID_i,m_i,X_i,U_i,P_{pub},T_x\right)$ as h_i . If $H_3\left(ID_i,m_i,X_i,U_i,P_{pub},T_x\right)$ already been defined as other value, C returns failure. Finally, C computes $V_i=\alpha_iP_{pub}+r_Qh_iX_i$ and returns $\sigma_i=\left(V_i,U_i,X_i\right)$, where r_Q is obtained from L_1. The σ_i is a valid signature because:

$$\begin{aligned}e\left(V_i,P\right)&=e\left(\alpha_iP_{pub}+r_Qh_iX_i,P\right)\\&=e\left(\alpha_iP,P_{pub}\right)e\left(r_Qh_iX_i,P\right)\\&=e\left(r_ih_iX_i+U_i,P_{pub}\right)e\left(X_i,r_Qh_iP\right)\\&=e\left(r_ih_iB+U_i,P_{pub}\right)e\left(X_i,h_iQ\right)\\&=e\left(h_iI_i+U_i,P_{pub}\right)e\left(X_i,h_iQ\right)\end{aligned}$$

AggSigning _ Queries : It can be obtained through ***Signing _ Queries***.

Forgery : Suppose $\mathcal{A}_1$ outputs a valid forgery $\left(m_i,V_i,U_i,X_i\right)$, C obtains $\left(ID_i,X_i,c_i,r_i,I_i\right)$ from L_2. If $c_i=1$ returns failure. Otherwise, we know that C can obtain another forgery $\left(m_i,V_i^{'},U_i,X_i\right)$.

Suppose the forgeries are valid. Therefore, $e\left(V_i, P\right) = e\left(U_i + h_i I_i, P_{pub}\right) e\left(h_i Q, X_i\right)$ and $e\left(V_i^{'}, P\right) = e\left(U_i + h_i^{'} I_i, P_{pub}\right) e\left(h_i Q, X_i\right)$. Further, we can calculate that $V_i = \alpha_i R_i + h_i R_i + x_i h_i Q$, $V_i^{'} = \alpha_i R_i + h_i^{'} R_i + x_i h_i^{'} Q$.

For $R_i = aI_i$, $I_i = r_i B$ and $V_i - V_i^{'} = \left(h_i - h_i^{'}\right)\left(R_i + x_i Q\right)$, the CDH instance can be computed as follows:

$$\text{abP} = r_i^{-1}\left(\left(h_i - h_i^{'}\right)^{-1}\left(V_i - V_i^{'}\right) - x_i Q\right)$$

If the following conditions are met, C can compute the CDH instance.

E_1: The challenger C does not fail to exit in all $PartialSecretKey\_Queries$.

E_2: The forged signatures of $\mathcal{A}_1$ are valid.

E_3: $\mathcal{A}_1$ successfully forged a valid signature and C does not fail to exit.

We know that $\Pr\left[E_1\right] \geq (1-\mu)^{q_{ps}}$, $\Pr[E_2 \mid E_1] \geq \varepsilon_1$ and $\Pr[E_3 \mid E_1 \wedge E_2] \geq \mu$ .Therefore,

$$\Pr\left[E_1 \wedge E_2 \wedge E_3\right] = \Pr\left[E_1\right] \cdot \Pr\left[E_2 \middle| E_1\right] \cdot \Pr\left[E_3 \middle| E_1 \wedge E_2\right] \geq (1-\mu)^{q_{ps}} \varepsilon \mu .$$

For $\mu = \dfrac{1}{q_{ps}+1}$, we can compute:

$$\varepsilon \geq \frac{\varepsilon_1}{\left(q_{ps}+1\right)\text{e}}$$

and, $t \leq t_1 + \mathcal{O}\left(q_h + q_{us} + q_{ps} + q_{pk} + q_{sig} + q_{as}\right) T_m$.

Theorem 2: Under the CDH assumption, the proposed certificateless aggregated signature scheme is unforgeable against adaptively chosen-message attacks to adversary $\mathcal{A}_2$ in the random oracle model.

Proof: Assuming that adversary $\mathcal{A}_2$ breaks Game2 with advantage ε_1 in t_1 time, challenger C can gain advantage $\varepsilon \geq \left(\dfrac{q_{us}}{q_{us}+1}\right)^{q_{us}} \dfrac{\varepsilon_1}{\left(q_{us}+1\right)}$ at time $t \leq t_1 + \mathcal{O}\left(q_h + q_{us} + q_{pk} + q_{sig} + q_{as}\right) T_m$ to solve the CDH problem.

Init : C sets $aP = A$, $bP = B$, and $Q = H_1\left(P, P_{pub}\right)$, then sends the public parameters $params = \left\{P, P_{pub}, Q, G, G_T, H_1, H_2, H_3, p\right\}$ and the master key s to adversary $\mathcal{A}_2$.

H_1 _ Queries : Initial $L_1 = \left(P, P_{pub}, r_Q, Q\right)$ is empty. Challenger C receives $\left(P, P_{pub}\right)$ from the adversary $\mathcal{A}_2$. If $\left(P, P_{pub}\right)$ is included in L_1 , return the corresponding Q. Otherwise, C chooses $r_Q \in_R Z_p^*$ and returns $Q = r_Q A$, then adds $\left(P, P_{pub}, r_Q, Q\right)$ to L_1.

H_2 _ Queries : Initial $L_2 = \left(ID_i, X_i, c_i, r_i, I_i\right)$ is empty. Challenger C receives $\left(ID_i, X_i\right)$ query for H_2 from adversary $\mathcal{A}_2$. If the query has ever been made, challenger C returns the same value. Otherwise, C chooses $r_i \in_R Z_p^*$, then returns $I_i = r_i P$ and adds $\left(ID_i, X_i, c_i, r_i, I_i\right)$ to L_2.

SecretKey _ Queries : The challenger C receives $\left(ID_i, x_i, R_i, X_i\right)$ from adversary $\mathcal{A}_2$. If $c_i = 0$, returns failure. Otherwise, if $\left(ID_i, x_i, R_i, X_i\right)$ is include in L, returns X_i . If L do not include $\left(ID_i, x_i, R_i, X_i\right)$, C chooses $x_i \in_R {}^1Z_p$ and returns $X_i = x_i P$ .Finally, C adds $\left(ID_i, x_i, R_i, X_i\right)$ to L.

PartialSecretKey _ QherieS : The challenger C receives $\left(ID_i, X_i\right)$ from adversary $\mathcal{A}_2$, then returns $R_i = sI_i$ and adds $\left(ID_i, x_i, R_i, X_i\right)$ to L.

PublicKey _ Queries : The challenger C receives ID_i from adversary $\mathcal{A}_2$, then gets the $\left(ID_i, X_i, c_i, r_i, I_i\right)$ from L_2 . If $c_i = 0$, C sets $X_i = x_i B$ to update the $\left(ID_i, x_i, R_i, X_i\right)$ .If $c_i = 1$ and $X_i = \perp$, C chooses $x_i \in_R Z_p^*$ and returns $X_i = x_i P$. Finally, C adds $\left(ID_i, x_i, R_i, X_i\right)$ to L.

Signing _ Queries: The Signing inquiry phase is the same as in the Game1.

Forgery: Suppose $\mathcal{A}_2$ outputs a forgery $\left(m_i, V_i, U_i, X_i\right)$, C obtains $\left(ID_i, X_i, c_i, r_i, I_i\right)$ from L_2 . If $c_i = 1$ returns failure. Otherwise, we know that C can obtain another forgery $\left(m_i, V_i^{'}, U_i, X_i\right)$.

For $e\left(V_i, P\right) = e\left(U_i + h_i I_i, P_{pub}\right) e\left(h_i Q, X_i\right)$, $e\left(V_i^{'}, P\right) = e\left(U_i + h_i^{'} I_i, P_{pub}\right) e\left(h_i Q, X_i\right)$, and $Q = r_Q A$, $X_i = x_i B$, $I_i = r_i P$, we can compute the CDH instance:

$$\text{abP} = \left(x_i r_Q\right)^{-1}\left(\left(h_i - h_i^{'}\right)^{-1}\left(V_i - V_i^{'}\right) - r_i P_{pub}\right)$$

If the following conditions are met, C can compute the CDH instance.

E_1: The challenger C does not fail to exit in all ***SecretKey _ Queries***.

E_2: The forged signatures of $\mathcal{A}_2$ are valid.

E_3: $\mathcal{A}_2$ successfully forged a valid signature and C does not fail to exit.

Similar to Game1 we know,

$$\Pr\left[E_1 \wedge E_2 \wedge E_3\right] = \Pr\left[E_1\right] \cdot \Pr\left[E_2 \middle| E_1\right] \cdot \Pr\left[E_3 \middle| E_1 \wedge E_2\right] \geq \left(1-\mu\right)^{q_{us}} \varepsilon \ \mu_1$$

For $\mu = \dfrac{1}{q_{us}+1}$, we can compute

$$\varepsilon \geq \left(\frac{q_{us}}{q_{us}+1}\right)^{q_{us}} \frac{\varepsilon_1}{\left(q_{us}+1\right)}$$

and $t \leq t_1 + \mathcal{O}\left(q_h + q_{us} + q_{pk} + q_{sig} + q_{as}\right) T_m$.

V. Performance Evaluation

This section simulates and implements the proposed certificateless aggregate signature algorithm and other schemes. Our experimental running configuration: i5-8500 CPU @ 3.00GHz, 8GB RAM HP desktop, Windows 10 operating system, IDEA development environment, JAVA 1.8.0 version, and JPBC version 2.0.0

TABLE I. Comparisons of Aggregate Schemes

Schemes	Signing	Verification	Aggregate Verification	Signature length
[Chen]	$4S + 2H$	$4P + 2S + 3H$	$4P + 2nS + (n+2)H$	$(n+1)\|G\|$
[Hang]	$5S + H$	$3P + 3S + 2H$	$3P + 3nS + (n+1)H$	$(n+1)\|G\|$
[Li]	$4S + H$	$3P + 3S + H$	$4P + 3nS + nH$	$(n+1)\|G\|$
[Malhi]	$3S + 2H$	$4P + 2S + 3H$	$4P + 2nS + (n+2)H$	$(n+1)\|G\|$
Ours	$3S + H$	$3P + 2S + H$	$3P + 2nS + nH$	$2\|G\|$

As shown in Table 1, we counts the number of basic operations used in algorithms such as literature [17-20]. P is bilinear pairing operations. S is scalar multiplication operations in group G. H is $\{0,1\}^* \rightarrow G$ hash operation. $|G|$ is the bit length of group G.

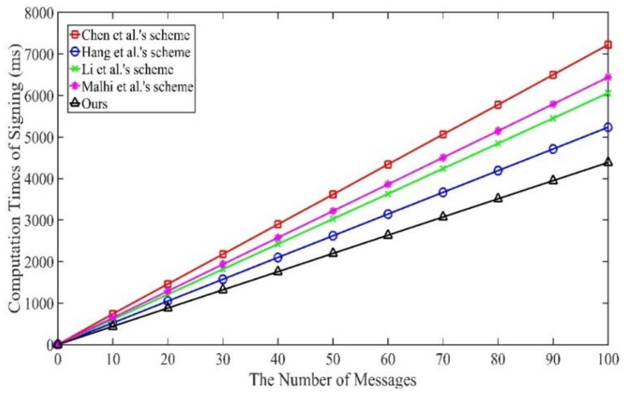

Fig. 5. Comparison of signing time in blockchain.

According to the comparison result in Figure 5, it is known that our certificateless aggregate signature algorithm has lower computational cost than other schemes in the signing phase under the different number of messages, which is beneficial to reduce the time for users to generate signature of data and improve the transaction efficiency of the blockchain system.

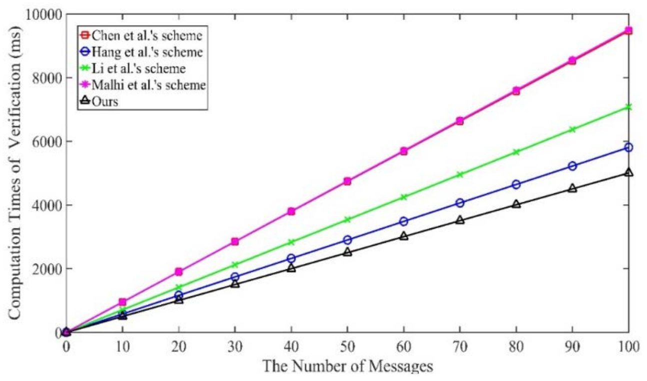

Fig. 6. Comparison of verify time in blockchain

In addition, as shown in Figure 6, the verification cost of our scheme is the lowest compared with other schemes under the different number of messages, which effectively improves the verification efficiency of user's data and ensures timely blockchain data deletion and modification.

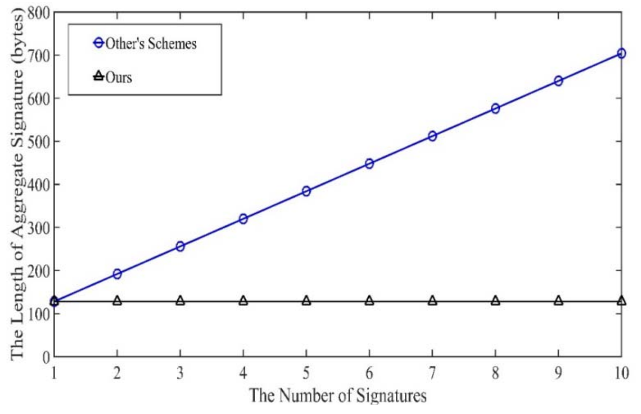

Fig. 7. Comparison of the length of aggregate signature

In the Figure 7, we compare the length of aggregate signature of our scheme with other schemes. Combining with the previous theoretical analysis, we know that the length of signature in our scheme is fixed and is only $2|G|$ with the increase of the number of signatures, which is conducive to the transmission of signatures in all kinds of applications under limited bandwidth.

VI. CONCLUSION

In this paper, we propose an anonymous editable blockchain scheme by reconstructing the blockchain structure and combining with the proposed certificateless aggregate signature algorithm. First, the miner verifies the signature of the user to ensure the subsequent accountability of the data on the chain. Secondly, only if the number of users who agree to edit meets the threshold, can the signing algorithm of smart contract be triggered to achieve legal editing of blockchain data. Note that only the signature generated by the registered user can pass the verification of the smart contract. Because the editing behavior to blockchain data is jointly determined by most legitimate registered users, the legality of blockchain data modification and deletion can be guaranteed. Finally, RC can restore the real identity of malicious users and hold them accountable. The security analysis and simulation experiments show that our scheme has certain practicability.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant 62072369, 62072371, 61802303 and 61772418.

REFERENCES

[1] Yuan Y, Wang F.Y. Current status and prospects of blockchain technology development[J]. Acta Automatica Sinica, 2016, 42(4):481-494.

[2] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system Bitcoin: A Peer-to-Peer Electronic Cash System[J]. Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper, 2009.

[3] Wang S, Ouyang L.W, Yuan Y, et al. Blockchain-enabled smart contracts: architecture, applications, and future trends[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49(11): 2266-2277.

[4] Wood G. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014): 1-32.

[5] Buterin V. Critical update re: DAO vulnerability[J]. Ethereum Blog, https://blog.ethereum.org/2016/06/17/critical- update-re-dao-vulnerabili ty/ June, 2016.

[6] Garay J, Kiayias A, Leonardos N. The bitcoin backbone protocol: Analysis and applications[C].2015 Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2015: 281-310.

[7] Ouyang L.W, Wang S, Yuan Y, et al. Smart Contract: Architecture and Progress[J]. Acta Automatica Sinica, 2019, 45(03):445-457.

[8] Park S, Kwon A, Fuchsbauer G, et al. Spacemint: A cryptocurrency based on proofs of space[C]. 2018 International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2018: 480-499.

[9] Krawczyk H M, Rabin T D. Chameleon hashing and signatures: U.S. Patent 6,108,783[P]. 2000-8-22.

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

[11] Li P.L, Xu H.X, Ma T.J, et al. Research on Modifiable Blockchain Technology [J]. Journal of Cryptography, 2018, 005(005):501-509.

[12] Derler D, Samelin K, Slamanig D, et al. Fine-Grained and Controlled Rewriting in Blockchains: Chameleon-Hashing Gone Attribute-Based[J]. IACR Cryptol. ePrint Arch., 2019: 406.

[13] Marsalek A, Zefferer T. A correctable public blockchain[C].2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications TrustCom. IEEE, 2019: 554-561.

[14] Deuber D, Magri B, Thyagarajan S A K. Redactable blockchain in the permissionless setting[C].2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 124-138.

[15] Ren Y.L, Xu D.T, Zhang X.P, et al. Deletable blockchain based on threshold ring signature[J]. Journal on Communications, 2019, 40(04):71-82.

[16] Cai X , Ren Y , Zhang X . Privacy-Protected Deletable Blockchain[J]. IEEE Access, 2019, PP(99):1-1.

[17] Al-Riyami S S, Paterson K G. Certificateless public key cryptography[C]. 9th International conference on the theory and application of cryptology and information security. Springer, Berlin, Heidelberg, 2003: 452-473.

[18] Chen H, Wei S.M, Zhu C.J, et al. A secure certificateless aggregation signature scheme [J]. Journal of Software, 2015, 26(005): 1173-1180

[19] Mal hi A K, Batra S. An efficient certificateless aggregate signature scheme for vehicular ad-hoc networks[J]. Discrete Mathematics & Theoretical Computer ence, 2015, 17(1).

[20] Hang T, He D B, Huang B J. Reattack of a Certificateless Aggregate Signature Scheme with Constant Pairing Computations[J]. The Scientific World Journal, 2014,(2014-3-13), 2014, 2014:343715.

[21] Li J, Yuan H, Zhang Y. Cryptanalysis and Improvement for Certificateless Aggregate Signature[J]. Fundamenta Informaticae, 2018, 157(1-2):111-123.

[22] Zhang Y, Deng R H, Han G, et al. Secure smart health with privacy-aware aggregate authentication and access control in Internet of Things[J]. Journal of Network and Computer Applications, 2018, 123(DEC.):89-100.

[23] Liu J, Cao H, Li Q, et al. A large-scale concurrent data anonymous batch verification scheme for mobile healthcare crowd sensing[J]. IEEE Internet of things Journal, 2018, 6(2): 1321-1330.

[24] Thumbur G , Rao G S , Reddy P V , et al. Efficient Pairing-Free Certificateless Signature Scheme for Secure Communication in Resource-Constrained Devices[J]. IEEE Communications Letters, 2020, 24(8):1641-1645.

[25] Xu Z , Luo M , Khan M K , et al. Analysis and Improvement of a Certificateless Signature Scheme for Resource-Constrained Scenarios[J]. IEEE Communications Letters, 2020, PP(99):1-1.

[26] Liu J , Wang L , Yu Y . Improved Security of a Pairing-Free Certificateless Aggregate Signature in Healthcare Wireless Medical Sensor Networks[J]. IEEE Internet of Things Journal, 2020, 7(6):5256-5266.

[27] Gayathri N B , Thumbur G , Kumar P R , et al. Efficient and Secure Pairing-Free Certificateless Aggregate Signature Scheme for Healthcare Wireless Medical Sensor Networks[J]. IEEE Internet of Things Journal, 2019, 6(5):9064-9075.

[28] Zhan Y, Wang B, Lu R. Cryptanalysis and Improvement of a Pairing-Free Certificateless Aggregate Signature in Healthcare Wireless Medical Sensor Networks[J]. IEEE Internet of Things Journal, 2020, 23(10):1-1.