Front. Comput. Sci., 2019, 13(6): 1182–1197
https://doi.org/10.1007/s11704-017-6338-8

RESEARCH ARTICLE

New instant confirmation mechanism based on interactive incontestable signature in consortium blockchain

Yan ZHU (✉)[1], Khaled RIAD[1,2], Ruiqi GUO[1], Guohua GAN[1], Rongquan FENG[3]

1 School of Computer & Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
2 Mathematics Department, Faculty of Science, Zagazig University, Zagazig 44519, Egypt
3 School of Mathematical Sciences, Peking University, Beijing 100871, China

Abstract The blockchain is a radical innovation that has a considerable effect on payments, stock exchanges, cybersecurity, and computational law. However, its limitations in terms of the uncertainty involved in transaction confirmation are significant. In this paper, we describe the design of a decentralized voting protocol for the election of a block generator in a consortium blockchain and propose a new system framework that allows fast and exact confirmation of all transactions. In addition, to replace a transaction's owner signature, a new interactive incontestable signature between the dealer and owner is used to confirm a transaction. By means of this signature, the dealer can assure the owner that a transaction will be permanently included in the blockchain in a non-repudiation manner. Moreover, the signatures of all transactions in a block share only one witness that provides membership proof between the block and these transactions. Finally, a security and performance analysis shows that the proposed schemes are provably secure and highly efficient.

Keywords security, blockchain, signature, consortium, interactive proof

1 Introduction

Over the last years, the concept of a "*private blockchain*" has become very popular in banks and financial institutions (FIs). The source of its popularity is the great success of the blockchain, which is the core technology behind Bitcoin. Essentially, rather than a fully public and uncontrolled network and a state machine secured by cryptoeconomics (e.g., proof of work and proof of stake), a fully private blockchain is a blockchain where the write permissions are kept centralized to one organization and the read permissions may be public or restricted to an arbitrary extent. Such systems have been a primary focus of interest of FIs. For example, venture capital invested in blockchain-related companies has increased considerably over the past three years and is on track to exceed $600 million in 2015 [1]. Currently, the foundational layer and infrastructure necessary to support a rich ecosystem of blockchain-based applications and services is being established.

The key to Bitcoin's security (and success) is that it is "fully decentralized" [2]. However, a private blockchain can be considered a traditional centralized system with a degree of cryptographic auditability attached. It has lost the advantage of *decentralization and consensus* that a public blockchain provides. On the one hand, without a central point, decentralization could provide a significantly more durable setting than today's centralized systems, which would be better able to withstand any threats to its functioning from malicious network attacks to power outages. On the other hand, while anomalies, e.g., conflicts, fails, and attacks, occur in the case of a few participants, the consensus mechanism has the ability to cause a distributed network

Received June 28, 2016; accepted December 22, 2016

E-mail: yanzhu@ustb.edu.cn; khaled.riad@science.zu.edu.eg

to come to an agreement regarding the state of data without involving a trusted third party (TTP) or super-nodes. Therefore, there is an urgent need to integrate the backbone of public blockchains, decentralization and consensus, into private blockchains.

To achieve this goal, the consortium blockchain was introduced recently (see the Ethereum's website). This new type of blockchain is constructed on a consortium that consists of some pre-selected FIs, where each FI has many branches for collecting clients? transactions (as shown in Fig. 1). In this consortium, all FIs cooperate to operate and control the blockchain; however, the right to read the blockchain may be public or restricted to specified clients. The consortium blockchain, therefore, creates a new system where access permissions are more tightly controlled, with rights to modify or even read the blockchain state being restricted to a few clients, while many types of partial guarantees of authenticity and decentralization that blockchains provide are still maintained. Hence, consortium blockchain can be considered "partially decentralized."

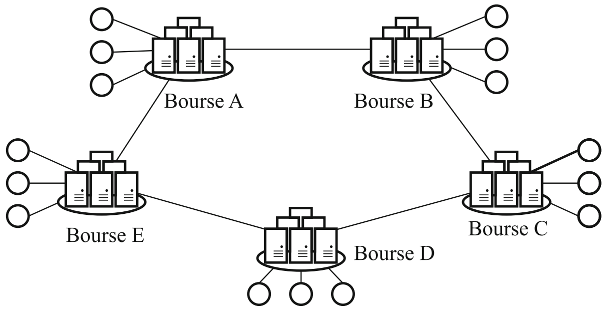


Fig. 1 Consortium blockchain

• **Motivation** At present, the major type of fault produced in a blockchain is *block conflict*. The so-called conflict is that a *fork* in a blockchain can occur if two blocks are published nearly simultaneously, as shown in Fig. 2(a). The public blockchain is a large network with ten thousand nodes, each of which is eligible to generate a block; therefore, a conflict cannot be avoided. The current solution for such a conflict is based on the "*Longest Chain Rule*" (LCR) [3]: if there exist multiple blocks, treat the longest chain as legitimate. This means that nodes follow the protocol rule that they will attempt to extend only the longest branch of which they have knowledge, as shown in Fig. 2(b).

This rule causes a few transactions on the wrong side of the fork to be delayed. It also faces risks if double spending is attempted. In the current implementation, a new block is generated approximately every 10 min [3]. In addition, the uncertainty as to whether an established block is in the prevailing branch leads to a common rule that a given transaction is not confirmed until it is at least six blocks deep in the chain. In practice, Bitcoin has adopted six blocks as the standard confirmation period [4]. This can be problematic in certain real-time application scenarios.

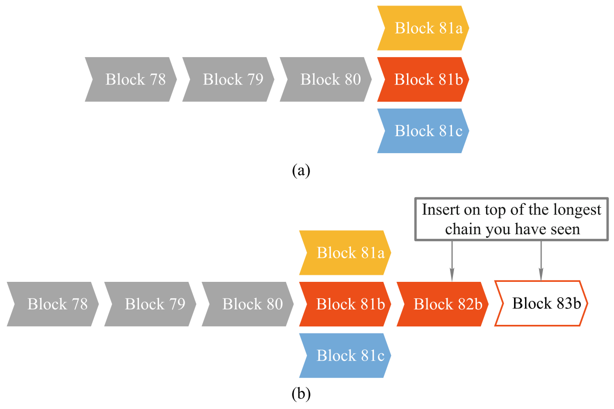


Fig. 2 Block conflict and current solution. (a) Conflict occurs due to block fork; (b) current solution on the "longest chain rule"

Although the conflict can be resolved, let us show whether instant confirmation can be implemented efficiently. Confirmation is a verification process that offers a final proof after validating a certain transaction. It is necessary, whenever a transaction is received, to obtain confirmation from other nodes on the network that the transaction is indeed valid. In Bitcoin, however, there is no notion of "instant confirmation," for the following reasons.

1) A transaction is implicitly confirmed through its inclusion in a block that is followed up by approximately six blocks (as described above). This process, taking at least 1 hr, causes a high confirmation latency.
2) Even after such a long wait, the transaction is not confirmed with total finality, which would mean it is permanently included in the blockchain. In fact, the blockchain offers 99.9999% finality [5] only after 2 hr, as does Bitcoin.
3) To determine whether a transaction has been validated, the client needs to search the most recent six blocks (six additional blocks for an implicit confirmation, which is also called confirmed by six blocks) 1 hr after he/she has published this transaction. This incurs a large computational overhead, because the search volume is about 24,000 transactions.

The search volume mentioned above is computed as $24,000 = 6 \times 4,000$ according to a maximum of roughly

4,000 transactions per block (the average transaction size is around 200–250 bytes and the block size is currently at most 1 MB in Bitcoin).

Therefore, we can never state with certainty that a transaction is "confirmed," because it is always possible that a transaction will apparently be included in the blockchain but be replaced by a competing block. This kind of replacement, called blockchain reorganization, is exponentially unlikely, but possible. For example, a 24-block reorganization due to a technical glitch was found in March 2013. As shown in Fig. 3, although it looks like the Branch A, including Block 3′ and Block 4′, has already been discarded after Block 5 is inserted, the confirmation of its actual discarding still needs to be determined by a number of subsequent blocks. This makes block confirmation with a large degree of uncertainty.

To conclude the above discussion, "instant confirmation" is still a challenging problem for the current blockchain. The important point to be remembered, however, is that there is no absolute notion of "permanently included" and the blockchain simply uses a reasonably safe policy of considering transactions confirmed when they are included with very high probability. The time spent is quite variable: sometimes confirmation may take tens of minutes and sometimes it may take more than 2 hr; however, on average it will take approximately 1 hr. Obviously, this is a fatal problem in blockchains? future application, given that the clients might be told "your yesterday's deal was confirmed without success for technical reasons."

• **Goal and approaches** In this study, our objective was to implement a new notion of "*Instant Confirmation with Incontestability*". It clear that instant confirmation [6] signifies an ability to confirm promptly whether a transaction has been permanently included in the blockchain. Incontestability is also an important ability, in that it signifies that the transaction's owner will not be able to successfully challenge the validity of an associated contract. Sometimes, a method that provides incontestability is considered to provide non-repudiation, that is, it guarantees the validity of a contract via digital signature and/or encryption. Instant confirmation with incontestability is a necessary feature for financial systems and the economy and is a key technical challenge in designing any cryptocurrency system.

Our goal is undoubtedly a great challenge. Fortunately, the consortium blockchain provided significant inspiration for achieving our goal. The consortium blockchain, proposed in August 2015, has attracted broad attention. The reason is that this infrastructure not only is a new concept, but also presents some new features.

1) It can be trusted and very well-connected and faults can quickly be fixed by allowing the use of consensus algorithms (e.g., Byzantine agreement (BA) and zero-knowledge proof), which offer "instant confirmation";
2) It uses cryptography and digital signatures to prove identity and authenticity and enforce read/write access rights; and
3) It ensures that the confirmation of transactions is prompter and cheaper, since they need to be verified by only a few nodes that can be trusted to have very high processing power and do not need to be verified by all (ten thousand) nodes.

These features of a consortium blockchain, in our opinion, can facilitate the resolution of the "slow and inexact" confirmation problem arising in the existing blockchains.

• **Contribution** In this paper, we address the problem of implementing instant confirmation with incontestability for a large number of financial transactions in the blockchain framework. First, we determined that the most essential reason for the "slow and inexact" confirmation problem is the emergence of the fork problem. However, a consortium blockchain having an adequate scale, which can be trusted and very well-connected, can easily solve the problem. Based on this new framework, we propose some new notions and

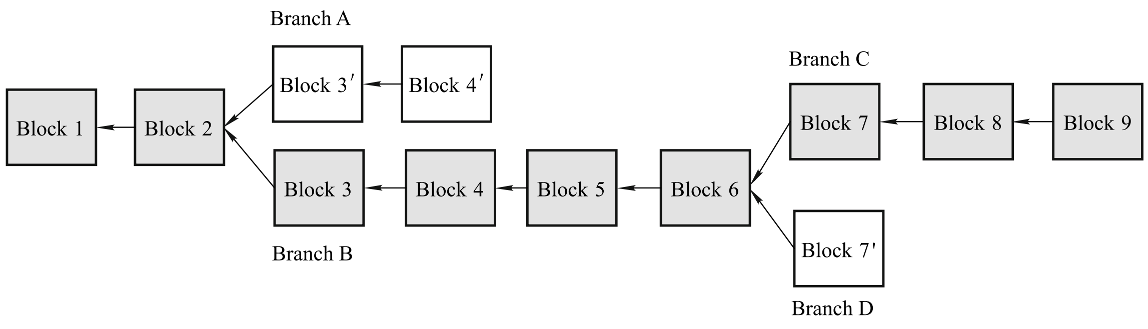


Fig. 3 Example of a blockchain fork

schemes, including an accounting cycle, dealer election protocol, and interactive group-oriented signature, to achieve our goal. In detail, our contributions are as follows.

1) We propose a new framework for fast and exact confirmation of all transactions. In this framework, a working period, called an accounting cycle, involves two parallel tasks: dealer election and block generation. For dealer election, we design a voting scheme involving all participants, which has the properties of uniqueness, agreement, and unpredictability. This scheme resolves the blockchain fork problem. In block generation, we introduce a witness module into the block structure, which is considered verifiable evidence of the confirmation.
2) Replacing the transaction's owner signature, a new interactive incontestable signature (IIS) scheme between the dealer and owner is presented for implementing confirmation with incontestability. By means of this signature, the dealer can assure the owner that a transaction will be permanently included in the blockchain in a non-repudiation manner. Moreover, the signature of all transactions in a block shares only one witness, which provides membership proof between the block and these transactions. In addition, this signature is short and easily built in a 3-move simple process.

Our signature scheme is constructed on a general bilinear map group system. We also prove the security of the scheme under the unforgeability of a signature by an owner and the incontestability of a dealer, based on two extended computational bilinear Diffie-Hellman assumptions, eCBDH_1 and eCBDH_2, respectively. Our experimental results show that the properties of the scheme are good: a short signature, high performance, and low-key storage.

• **Organization** The rest of this paper is organized as follows. Section 2 introduces the preliminaries and existing work. Section 3 presents our system model and requirements. Our signature construction is presented in Section 4. The performance evaluation is provided in Section 5. Finally, the conclusions and future extensions of our model are discussed in Section 6.

2 Preliminary and existing work

2.1 Blockchain

The blockchain [7, 8], which ensures the security of distributed data storage, is the core of Bitcoin. A blockchain is essentially a ledger of all transactions. It is append-only, meaning new data can be added to the end of the ledger, but data can never be removed once included. This ledger is necessary to prevent double-spending, which is a key technical challenge in designing any cryptocurrency system. The basic structure of a blockchain, which consists of a series of blocks connected in a hash chain, is shown in Fig. 4. Transactions become effective after they have been referenced in a block in the blockchain, which serves as the official records of executed transactions.

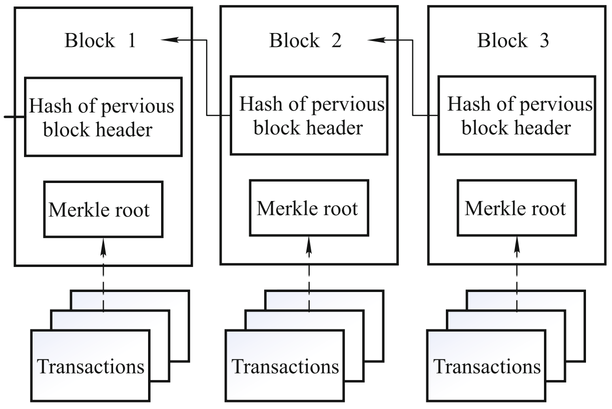


Fig. 4 Basic structure of a blockchain

The running setting of a blockchain is a peer-to-peer (P2P) network, and it provides an interface that broadcasts data to all nodes in the network. There are two types of objects that are broadcast: transactions and blocks. Both object types are addressed by a hash of the object data. Transactions are the operations whereby money is combined, divided, and remitted. Blocks record the vetted transactions as valid.

2.2 Transactions

A Bitcoin transaction [2] consists of a series of inputs and outputs. Each output has only two fields: the Bitcoin value and a field that specifies who is authorized to spend that value. Each input is simply a reference to a previous transaction output, as well as its full value. Both also contain fragments of executable script, on the input side for redeeming inflows and on the output side for designating payees.

In a Bitcoin transaction, scripting language is used to construct the inputs and outputs flexibly. It is similar to a language called Forth, which is a simple, stack-based programming language. The scripting language consists of data and certain operations to verify the signature of a transaction. For a transaction to be valid, its outputs must not exceed its inputs and its owner must show his/her title to each input claimed. The title is checked by evaluating the input script fragment concatenated with the script fragment from the output (of an earlier transaction) that the input references.

2.3 Block generation

In a blockchain, a block is a structure that collects all the transactions executed during a set period into a list. In Bitcoin, mining is used for block generation, that is, the process of adding transaction records to the blockchain. The primary purpose of mining is also to allow Bitcoin nodes to reach a secure, tamper-resistant consensus. Bitcoin uses the hashcash proof of work system for mining. Let us recall the process of block generation in Bitcoin.

1) Perform proof of work to elect a block generator;
2) Collect transactions;
3) Search for longest chain;
4) Generate a new block linked to the blockchain;
5) Repeat the process for the next block.

We call the above process an accounting cycle. So that a block will be accepted by the network participants, miners must complete a proof of work that covers all the data in the block. The difficulty involved in this work is adjusted such that the rate at which new blocks can be generated by the network is limited to one every 10 min. Because of the very low probability of successful block generation, which worker computer in the network will be able to generate the next block cannot be predicted.

2.4 Related work

The construction of a scalable blockchain has been an active research issue since the Bitcoin [7] was first introduced in 2008. Karame et al. [3] discussed an approach that enables fast confirmation of transactions in the network. It is related to possible attacks on nodes that accept zero-confirmation transactions. The GHOST protocol, introduced by Sompolinsky et al. [9], increases the number of transactions processed per second and pushes more transactions to the network by accepting the main valid blockchain. Lewenburg et al. [10] replaced the blockchain structure with a directed acyclic graph. A main chain still exists, but its blocks may refer to pruned branches to include their transactions. Eyal et al. [11] proposed a blockchain protocol that improves the transaction rate by introducing two block types within an epoch: key blocks are for leader elections and microblocks contain only transactions proposed by the leader of that epoch.

Significant effort has been devoted to developing communication-efficient consensus protocols. The idea of dividing users into committees is prevalent in the existing literature. It was first introduced by Bracha [12] and used in [13–15]. For honest but crash-prone users, there exists an optimal algorithm based on the idea of universe reduction. Gilbert and Kowalski [16] extended this idea by choosing a small committee to manage the process. Further, King and Saia [17] showed the manner in which this idea can be used to go from BA to BA with a nonadaptive adversary. For malicious users, Toueg et al. [18] developed an efficient algorithm with polynomial communication complexity in key improvement.

3 System model

3.1 Design objectives

The proposed model addresses the problem of building a new consortium blockchain that allows more exact confirmation and a higher performance than a private blockchain. Our study was based on the existing public blockchain, but our objective was to introduce some new properties and tools to improve and overcome its shortcomings and defects in order to adapt it to consortium settings. Precisely, our design objective is focused on two properties, as follows.

- **Instantaneity** ensures that each transaction is handled within a certain time period in order to meet the demand of high volume business without delay.
- **Confirmability** represents that a transaction will not face the risk of becoming invalid later, once it has been appended to a new block.

In addition, the incontestability property is also necessary, because it refers to the ability to ensure that the owner cannot deny the authenticity of a signature on a transaction or that the block generator (dealer) cannot deny the prior behavior after the transaction has been validated and included in a block in the blockchain. As described above, the current blockchain cannot satisfy these properties because of problems such as the excessive mining cost of block generation and the "longest chain rule" against conflicts. Hence, we sought new methods to meet our design objectives.

3.2 Proposed framework

We propose a new system framework to meet our purpose. This framework is a new blockchain, where the permissions of block generation and management are restricted by a set of trusted institutions, called *bourses*, but read permissions may be public or restricted to an arbitrary extent in light of actual needs. In Fig. 1, a simple consortium blockchain is presented with the consensus process, which determines which blocks are added to the chain and the current state. In this figure, the consensus process is controlled by a consortium of five financial bourses, each of which operates a node with a high-speed P2P network. Each bourse has branches, which are called *traders*, deployed in different locations. A trader receives transactions generated by the clients, called the *owners* of a transaction, within its realm. The size of the consortium should not be excessively large, reflecting the idea of "small for high-speed".

In order to guarantee instant confirmation, a new concept, called the *accounting cycle*, is introduced into the system. That is, we divide the generation time per block into many smaller intervals, in which all traders submit the received transactions to the bourse that was elected to establish the next block. We call this bourse the *dealer*. In every period, there are two parallel processes: dealer election and block generation, as shown in Fig. 5. The dealer election is used to avoid the block conflict problem (the fork problem, described in Section 1, and block generation is also applied for guaranteeing incontestability. Moreover, these two processes are concurrently executed to improve the performance. In detail, they need to accomplish the following.

- **Dealer election** In this phase, all the bourses in the system participate in a decision to determine which will be the dealer in the next interval. This is an essential process in ensuring the confirmability of transactions. The reason is that it can help traders to avoid the conflict related to generating blocks. We can say a conflict arises if more than two blocks are appended to a blockchain at roughly the same time. Our approach for resolving this conflict is to specify only one dealer for generating blocks in one accounting cycle. To do so, the common method is that all the bourses cooperatively execute a voting protocol to decide the next dealer. In our system, we introduce the BA for implementing the voting protocol. In the above-mentioned example, let us imagine a consortium of five financial bourses, each of which operates a node and at least three of which must cast a valid vote for the same dealer in order for the dealer to be elected. The detailed procedure is described in the next subsection.
- **Block generation** Block generation is executed by the dealer who was elected in the previous accounting cycle. In this phase, the dealer generates a new block, which includes all the valid transactions received in this cycle. This process consists of four steps:
  - Transaction collection: the dealer picks up the transactions signed in the previous period;
  - Transaction verification: the dealer verifies the validity and integrity of contracts;
  - Block generation: the dealer combines all valid transactions into a block;
  - Block verification: the owner of a transaction confirms that his/her transaction has been appended to the block.

In these steps, the verification of the transaction and block comprises two major processes: authentication of the identity of the owner and verification of the validity of the transaction. Partial information of the dealer is attached to each transaction that has passed the verification process. It ensures the incontestability of dealers by establishing affinity between the valid transactions handled by the dealer and the identity of this dealer. We construct an IIS scheme to create such a relationship. After it has been examined by other bourses, this newly-built block is permanently appended to the blockchain.

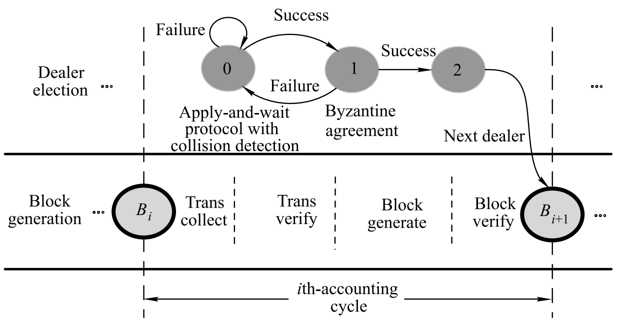


Fig. 5 Accounting cycle with two parallel processes

Our new framework must be dynamic and flexible to allow its practical application. This means that it must be possible to add or remove bourses dynamically without affecting the current setting. This requirement can be satisfied, because dealer election, built on a voting protocol, can automatically adapt and adjust the scale of the system. However, the total number of bourses must be kept small in order to ensure efficiency, because the voting protocol is based on the BA, and an increased scale will produce a traffic explosion. In this, our framework is fundamentally different in nature from the public blockchain, in which anyone in the world can generate a block.

3.3 Byzantine agreement protocol for dealer election

We now focus on the effective dealer election. This process is essentially a voting protocol in which each bourse is considered a voter and all bourses cooperatively determine which will be next dealer. The result of the dealer election should be random, so that an attacker cannot predict it. To meet our design objective, our voting protocol strives to achieve the following goals.

- **Uniqueness** The voter can vote only once and (at most) only one bourse wins the election;
- **Agreement** All correct processes determine the same result (victory) after the protocol terminates.
- **Unpredictability** No partial result is available before the final result is output.

Although voters are allowed to vote multiple times, uniqueness is ensured by counting only the most recent vote and eliminating the earlier votes linked to the same voter. Consistency guarantees that all bourses know who the next dealer is and then deliver new transactions to this dealer. A considerable amount of work has already been conducted on electronic voting protocols, but the existing protocols do not apply to our dealer election in consortium because almost all are built on the central count voting model rather than on our distributed setting.

Our dealer election method refers to the Byzantine generals' problem described by Lamport, Shostak, and Pease in their 1982 paper [19], as follows. *The Byzantine generals' problem is an agreement problem, in which a group of generals, each commanding a portion of the Byzantine army, encircles a city. These generals wish to formulate a common plan for attacking the city. In its simplest form, the generals must decide only whether to "attack" or "retreat." After observing the enemy, some generals may prefer to attack, while others prefer to retreat. The generals can communicate with one another only by messenger. The important point is that every general must agree on a common decision, because a halfhearted attack by a few generals would result in a rout and yield a worse result than a coordinated attack or a coordinated retreat.*

Fortunately, Lamport, Shostak, and Pease's work [19] showed that this problem is solvable if and only if more than two-thirds of the generals are loyal. It can be shown that if n is the total number of generals and t is the number of traitors in that n, then solutions to the problem exist only when $n > 3t$ and the communication is synchronous (bounded delay). This conclusion has been widely applied to various adversaries, which can thereby coordinate their actions most effectively. However, the protocol limits only the number of generals (less than 1/3 of generals) the adversary can control (in the Byzantine model) or stop (in the fail-stop model).

In Fig. 6, we describe an $\lfloor (n-1)/3 \rfloor$-resilient Byzantine agreement protocol proposed by Bracha [20]. This protocol provides probable security against any coalition of up to $\lfloor (n-1)/3 \rfloor$ corrupted parties, as well as the following properties.

- **Termination** With probability 1, all the uncorrupted parties eventually complete the protocol (i.e., terminate locally).
- **Correctness** The output of all the uncorrupted parties that complete the protocol is identical. Furthermore, if the transmitter is correct and sends σ to all parties, then all the uncorrupted parties output σ.

- Codeforthesender(on input m):

  Send: send the message (SE,m) toalltheparties.

- Code for party P_i:

  Spread: upon receiving a message (SE, m) , send (SP, m) to all the parties.

  Decide: upon receiving $n-t$ messages (SP,m') that agree on the value m', send (D,m') to all the parties.
  Or upon receiving $t+1$ messages (D,m') that agree on the value m', send (D,m') to all the parties.

  Agree: upon receiving $(n-t)$ messages (D,m') that agree on the value of m',send (A,m') to all the parties and accept message m' as a broadcast message.

Fig. 6 Byzantine agreement protocol for $t < n/3$

Based on [19], many BA schemes [21–23] have been presented in the last 30 years. Our dealer election method was also developed in [19]: *each voter is considered one general, and the voters must decide whether to "approve" or "oppose" a candidate dealer.* With the help of the BA, our election problems can be realized efficiently; however, the following two problems remain:

- How can a candidate dealer be recommended?
- How can an "approve" or "oppose" ballot be cast?

For the first problem, our solution is a simple protocol, called the *apply-and-wait protocol with collision detection* (AW/CD), in which a bourse randomly applies to be a candidate dealer and then detects other applications; if a collision is detected, it waits for a random time interval before attempting to reapply. So that applications are made randomly, we require that each bourse sets a working intensity λ, which denotes the number of allowed applications to be a dealer per unit of time, e.g., 15 times/hr. This value is also considered the parameter of Poisson distribution that expresses the probability of a given number of events occurring in a fixed interval of time if these events occur with a known average rate and independently of the time since the previous event. Based on this distribution, the bourse can easily determine the time at which it applies as a candidate. However, a collision will be detected if more than two candidates exist simultaneously. The following procedure is used to resolve the detected collision.

- Increment the candidate counter and continue detection until the minimum packet time is reached to ensure that all receivers detect the collision;
- Is the counter equal to 1? If so, revoke the BA protocol; otherwise, calculate a random backoff period based on the number of collisions, and then re-enter the main AW/CD procedure.

Why do we still have to execute the BA protocol even if this stage has obtained only one candidate? The reason is that the BA protocol helps us avoid nodes that failed to learn this candidate for various reasons or evade the fork problem that arises because the collision is not completely resolved.

To resolve the second problem, there are various means [24, 25] of casting a ballot for approval or opposition. The simplest approach is a random ballot that elects a representative by choosing a ballot at random. The advantages of this approach are that it is simple, stable, and easily analyzed. The more desirable approach is that each bourse may make the decision according to its own actual observations. For example, the bourses can vote according to trust modeling and evaluation of the candidate. This approach may allow untrusted bourses to be excluded, but it is difficult to build an effective trust model. Based on the above analysis, we adopted the random ballot for convenience in our study.

3.4 Validation process in block generation

We now discuss further the validation process in the block generation phase. In our system, we add a new part, called the proof module, which stores dealers' information *witness* for validating transactions, to each block. Our approach for implementing confirmation with incontestability is based on the IIS scheme. In addition, it allows a concise proof of membership by directly linking transactions to a block.

Before generating a new block, the dealer must verify every transaction collected in the current accounting cycle. The verification process consists of two steps: the dealer first authenticates whether the owner is the initiator of a certain transaction and then determines whether the transaction content is valid. If one transaction passes the validation, then the dealer and its owner jointly operate the IIS to generate a Tag, which replaces the original signature produced by the owner himself/herself. When a Tag has been attached to the transaction, it is considered that the transaction has been confirmed. Anyone can use this Tag to verify the content of the transaction publicly, and, more importantly, to trace back the identity of the dealer.

When the block has been established and broadcast to the network, other bourses must determine whether it is valid by validating all the transactions covered. This process is quite simple: the bourses examine the Tag attached to each transaction with the *witness* obtained from the proof module of this block and ensure that the referenced transaction does not constitute double spending. If a certain transaction passes the process, it is proved that this transaction is valid, as well as that it belongs to the block simultaneously.

3.5 Security model

We now consider various classes of potential adversaries against our system. The security of the entire system can be guaranteed by the trustworthiness of the consortium. The transactions stored in a blockchain are secure based on the so-called ledger principle because of the decentralization of the blockchain. Even if one third of the bourses fail or are maliciously attacked, the system can maintain normal functions according to Byzantine fault tolerance. Hence, we turn our attention to the security of transactions under the various potential adversaries.

The core security goal is to ensure the unforgeability of the interactively generated signature. In our system, there are three types of possible forgery attacks: (1) forgery of signatures by owners, (2) forgery of signatures by dealers, and (3) forgery of signatures by external attackers. Note that in our system, the dealer needs to authenticate the owner of a certain transaction before running the IIS to generate a signature. That is, that external attackers cannot forge a signature can be ensured by the process of authentication. We therefore focus our security model on the remaining two security requirements.

- **Unforgeability of signature by owner** In this case, the owner of a transaction may aim to skip the validation process to forge a signature by himself/herself.

This security type requires that the owner cannot produce any valid signature even if he/she has unlimited computational resources.

- **Incontestability of dealer** This requirement guarantees that the dealer cannot deny his/her signature once he/she has generated it interactively with owners.

We expect that our signature is provably secure for the above requirements. Moreover, this signature must have a small effect on the original blockchain structure, as well as being high-performance and easy to implement.

4 Interactive incontestable signature

4.1 Definition

Let κ represent a security parameter, $v(\cdot)$ represent a negligible function, and T indicate the set of transactions. We define the algorithms that constitute an interactive signature scheme as follows.

Definition 1 (IIS) An IIS scheme consists of a tuple of algorithms (Setup, OKeyGen, DKeyGen, WitGen, SigGen, Verify), as follows.

- **Setup** $(1^{\kappa}) \rightarrow mpk$ On the input of a security number, this algorithm outputs a parameter as the master public key mpk.
- **OKeyGen** $(mpk, u_i) \rightarrow (sk_i, pk_i)$ On the input of the master public key, for each owner u_i, this algorithm outputs an owner's key pair (sk_i, pk_i) generated by him/herself, where the owner holds the secret key sk_i, but the public key pk_i is public.
- **DKeyGen** $(mpk, dealer_d) \rightarrow (sk_d, pk_d)$ On the input of the master public key, this algorithm outputs a dealer's key pair (sk_d, pk_d) and the manager appends pk_d to mpk; that is, $mpk = \{mpk, pk_d\}$.
- **WitGen** $(mpk, sk_d) \rightarrow W$ On input of the master public key and dealer's private key, this algorithm generates a witness of a secret number and outputs this witness.
- **SigGen** $(Dealer \leftrightarrow Owner) \rightarrow \sigma_i$ This algorithm is an interactive proof protocol for yielding the signature between the dealer and the owner for a certain transaction T_i, where $\leftrightarrow$ denotes the interactive process between them, and either the dealer or owner can possess some secret information, respectively, and they interactively generate a message-signature pair (T_i, σ_i), where $T_i \in T$. It outputs the signature σ_i.
- **Verify** $(mpk, pk_i, pk_d, W, T_i, \sigma_i) \rightarrow \{0, 1\}$ Given mpk, the witness W and a message-signature pair (T_i, σ_i), this algorithm outputs 1 if it is a valid message-signature pair; otherwise, it outputs 0.

There are several difference between the above signature and a transitional signature:

1) Either the owner or the dealer can generate the public/secret key by him/herself and thus it easy to use;
2) The process of generating a signature is an interactive proof process between two parties: dealer and owner;
3) The verification of a signature requires two public keys of both dealer pk_d and owner pk_i at the same time, which means that this signature is permitted by both of them;
4) The witness is unique in each block and is shared in all transactions in this block, which sets up a strong membership between all transactions and this block.

4.2 Proposed scheme

We set up the proposed scheme using a bilinear map group system, which is obtained from general bilinear pairings. A bilinear map group is a tuple $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, p, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of the same order p. We say that the function e is a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ if, for any $g, h \in \mathbb{G}, a, b \in \mathbb{Z}_q^*$, $e(g^a, h^b) = e(g, h)^{ab}$.

We now describe the construction of the IIS scheme based on a bilinear map system $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, p, e)$, as follows.

- **Setup** $(1^{\kappa}) \rightarrow mpk$ This algorithm first generates the bilinear group $\mathbb{G}, \mathbb{G}_T$ of prime order p. Let g be the generator of $\mathbb{G}$. It chooses a random $h \in \mathbb{G}$ and outputs $mpk = (g, h)$.
- **OKeyGen** $(mpk, u_i) \rightarrow (sk_i, pk_i)$ The algorithm selects a random element $x_i \in \mathbb{Z}_p^*$ as sk_i and calculates $pk_i = h^{x_i}$.
- **DKeyGen** $(mpk, dealer_d) \rightarrow (sk_i, pk_i)$ For a certain dealer, the algorithm selects a random element $d \in \mathbb{Z}_p^*$ as sk_d and calculates $pk_d = g^d$.
- **WitGen** $(mpk, sk_d) \rightarrow W$ For a certain dealer, the algorithm randomly selects a secret element $a \in \mathbb{Z}_p^*$. Then, it calculates $wit_1 = g^a, wit_2 = (h^a)^{sk_d} = h^{ad} \in \mathbb{G}$ and outputs $W = (wit_1, wit_2)$.
- **SigGen**$(D(a, sk_d) \leftrightarrow O(sk_i))(mpk, W) \rightarrow \sigma_i$ The algorithm takes as input the master public key mpk and witness W, secret a, sk_d possessed by the dealer, and

sk_i possessed by the owner. To sign a transaction T_i, the dealer and this owner conduct a two-party protocol to calculate the signature, as follows.

1) The dealer calculates the hash value $T = H(T_i)^a$ of transaction T_i together with his/her secret $a \in \mathbb{Z}_p^*$, and then transmits T to the owner.
2) The dealer transmits T to the owner.
3) The owner randomly selects a number $r \in \mathbb{Z}_p^*$.
4) The owner transmits the value

$$\sigma_i' = (g^{x_i H(ID_i)} \cdot T)^r = (g^{x_i H(ID_i)} \cdot H(T_i)^a)^r, \quad (1)$$

to the dealer, where ID_i is the ID of T_i.

5) The dealer calculates $\sigma_i'' = (\sigma_i')^d$ with his/her private key sk_d and transmits the value

$$\sigma_i'' = (g^{x_i H(ID_i)} \cdot H(T_i)^a)^{rd}, \quad (2)$$

to the owner.

6) Finally, the owner removes the random r and obtains the signature

$$\sigma_i = (\sigma_i'')^{1/r} = g^{x_i H(ID_i) d} \cdot H(T_i)^{ad}. \quad (3)$$

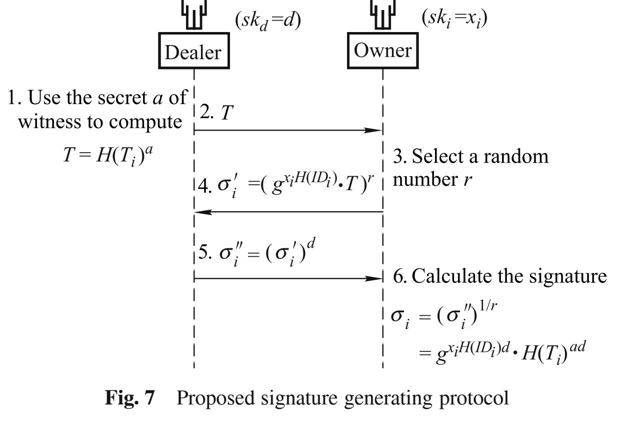
It outputs a signature σ_i for T_i.

- **Verify**$(mpk, pk_i, pk_d, W, (T_i, \sigma_i)) \rightarrow \{0, 1\}$ Given mpk, the witness W, and a message-signature pair (T_i, σ_i), this algorithm examines the equation

$$e(\sigma_i, h) = e((pk_d)^{H(ID_i)}, pk_i) \cdot e(H(T_i), wit_2). \quad (4)$$

If it is holds, it outputs 1; otherwise, it outputs 0.

In this scheme, the SigGen algorithm, which comprises five steps, is a three-move interactive proof system between the dealer and owner. For ease of comprehension of this protocol, the schematics for this protocol is provided in Fig. 7.



$(sk_d = d)$
$(sk_i = x_i)$
Dealer
Owner
1. Use the secret a of witness to compute $T = H(T_i)^a$
2. T
3. Select a random number r
4. $\sigma_i' = (g^{x_i H(ID_i)} \cdot T)^r$
5. $\sigma_i'' = (\sigma_i')^d$
6. Calculate the signature
$\sigma_i = (\sigma_i'')^{1/r} = g^{x_i H(ID_i) d} \cdot H(T_i)^{ad}$

Fig. 7 Proposed signature generating protocol

Finally, by using Eq. (4) of the Verify algorithm, the correctness of signature σ_i generated interactively can be proved based on the bilinear map as

$$\begin{aligned} & e((pk_d)^{H(ID_i)}, pk_i) \cdot e(H(T_i), wit_2) \\ & = e((g^d)^{H(ID_i)}, g^{x_i}) \cdot e(H(T_i), h^{ad}) \\ & = e(g^{x_i d H(ID_i)}, h) \cdot e(H(T_i)^{ad}, h) \\ & = e(g^{x_i d H(ID_i)} \cdot H(T_i)^{ad}, h) \\ & = e(\sigma_i, h). \end{aligned} \quad (5)$$

4.3 Integration with blockchain

While the system model described in the previous section provided an overview of instant confirmation, we have not yet described the integration process of the IIS scheme with a consortium blockchain for instant confirmation. In fact, it is not difficult to understand the integration process. The reason is that we are not attempting to change the original blockchain structure, but only indirectly to replace the existing elliptic curve digital signature algorithm (ECDSA) signature, which is used in the current blockchain, with our new IIS signature.

We first discuss the preprocessing of transactions before block generation, which is fairly straightforward. Assume there exists a trade between Alice (the sender) and Bob (the recipient). Note that Alice and Bob hold the public-private key pair (sk_A, pk_A) and (sk_B, pk_B), respectively, which is generated by $OKeyGen(mpk, u_A)$ and $OKeyGen(mpk, u_B)$. Alice first generates a transaction T_A in terms of the trade, in which the transaction T_A contains the recipient's public-key pk_B, transaction content, and a simple ECDSA signature σ. By using such an ECDSA signature, the dealer authenticates that Alice is the sender of the transaction T_A. Then, Alice broadcasts T_A to the entire network.

Next, let us show how the dealer generates a new block. The dealer first produces a blockhead and appends a new field, which stores the witness W for validating the dealer's identity. The witness can be computed by $WitGen(mpk, sk_d) \rightarrow W$. Second, the dealer searches for all transactions collected in the previous accounting cycle. For a certain transaction T_A, the dealer first authenticates Alice's identity through its original signature σ by using Alice's public key pk_A and verifies the correctness of the transaction' content according to the regulation that the value redeemed is greater than the value transferred. If these two conditions hold and the referenced transaction is not claimed as an input into a different transaction (to avoid double-spending), the dealer invites Alice to execute the $SigGen$ protocol interactively. The final result of the protocol $(D(a, sk_d) \leftrightarrow O(sk_A))(mpk, W)$ is a new signature σ_A, called Tag, with which the original signature σ is replaced. In Fig. 8, we show the new block structure after the witness W and the tag σ_A are integrated into a blockchain, where the new components are marked by the color blue.

After this block has been built and broadcast, other bourses in the consortium must examine whether all the transactions in this block are valid by running the $Verify$ algorithm. In Fig. 8, we illustrate this verification process in detail. For a certain transaction T_A, the bourse first obtains Tag (σ_A) attached to T_A, the witness W stored in the block, and Alice's pk_A in the referenced transaction, which can be found through the hash chain. T_A is regarded as valid if two conditions, $Verify(mpk, pk_A, pk_d, W, (T_A, \sigma_A)) = 1$ and the referenced transaction is not claimed as an input into a different transaction, are satisfied. The bourse continues to examine other transactions in this block; otherwise, if the *Verify* algorithm returns 0 or the referenced transaction has been claimed, T_A is regarded invalid and this block is discarded. Finally, if all bourses pass this validation process, the consortium accepts this block as valid and appends it to the blockchain.

4.4 Security analysis

We now analyze the security of our construction. Considering that the unforgeability of a signature by external attackers [26, 27] is ensured by the authentication process, as discussed above, we focus on two types of security properties (unforgeability of a signature by an owner and the incontestability of the dealer). We analyze these two properties as follows.

4.4.1 Unforgeability of signature by an owner

First, we define the unforgeability of a signature by an owner based on the general security definition of the signature, as follows.

Definition 2 A signature scheme is $(t, q_O, q_D, q_H, \epsilon)$-secure against the unforgeability of a signature by an owner if any adversary $\mathcal{A}_1$ breaks the scheme with a negligible probability ϵ. The advantage is

$$Adv_{\mathcal{A}_1} = \Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ mpk \leftarrow Setup(1^\kappa), \\ \{pk_i\} \leftarrow \mathcal{A}_1^{OKeyGen(mpk,\cdot)=(sk_i,pk_i)}, \\ pk_d \leftarrow \mathcal{A}_1^{DKeyGen(mpk,\cdot)=(sk_d,pk_d)}, \\ W \leftarrow WitGen(mpk, sk_d), \\ (T_i^*, \sigma_i^*) \leftarrow \mathcal{A}_1(\{pk_i\}, pk_d, W) \end{array}\right] \leqslant \epsilon,$$

for t time, q_O and q_D times queries for owner's and dealer's keys, and q_H times queries for the hash oracle, where $\mathcal{A}_1^{OKeyGen(mpk,\cdot)=(sk_i,pk_i)}$ and $\mathcal{A}_1^{DKeyGen(mpk,\cdot)=(sk_d,pk_d)}$ denote the adversary's oracle queries for owner's and dealer's public keys, respectively.

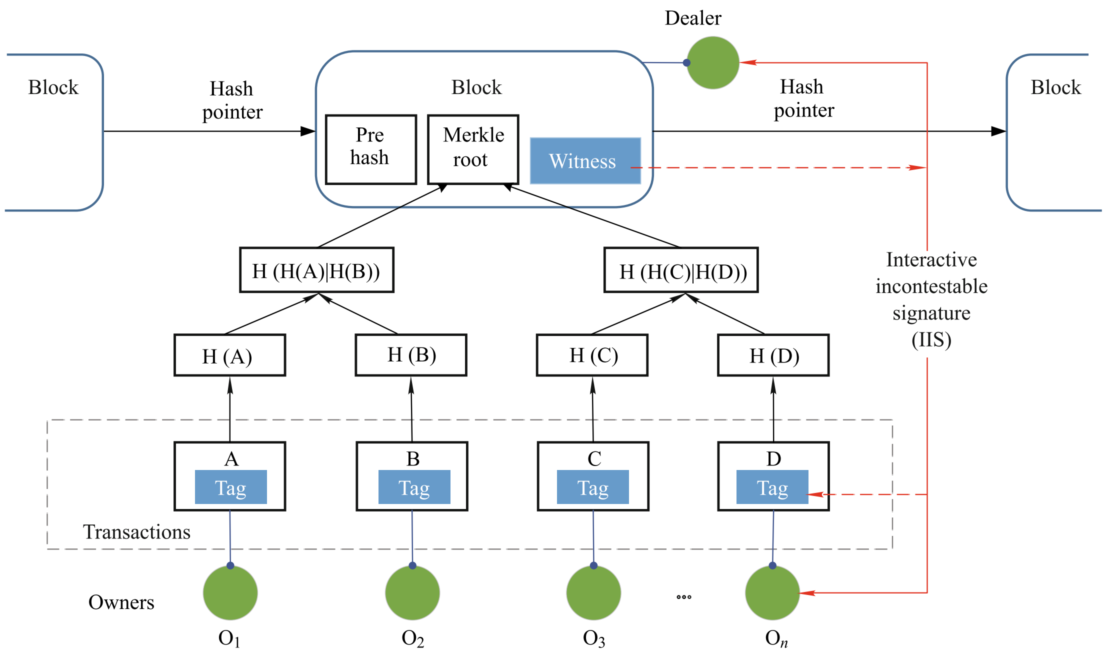


Fig. 8 New blockchain structure and verification process of proposed interactive incontestable signature scheme

The unforgeability of a signature by an owner in our proposed scheme is based on the eCBDH$_1$ assumption, which is defined as follows.

Definition 3 (eCBDH$_1$ assumption) Given $G, H, G^a, H^a, H^b \in \mathbb{G}$ for unknown $a, b \in \mathbb{Z}_q^*$, the eCBDH$_1$ assumption states that it is computationally intractable to compute the value of G^{ab}.

Next, we prove that our scheme provides unforgeability of a signature by an owner according to the following theorem.

Theorem 1 Let $\mathbb{G}$ be a (t', ϵ') group for Diffie-Hellman of order p. Then, the signature scheme on $\mathbb{G}$ is $(t, q_O, q_D, q_H, \epsilon)$-secure against the unforgeability of a signature by an owner, where

$$t \leqslant t' - 2(\log p)(q_O + q_D + q_H), \quad \epsilon \geqslant \epsilon'.$$

We assume an adversary $\mathcal{A}$ breaks our interactive signature scheme. We use $\mathcal{A}$ to construct a simulator $\mathcal{B}$ that breaks two types of extended-Computational Bilinear Diffie-Hellman problems, defined later in this section. The proof process is shown in Fig. 9.

Proof Suppose there exists a PPT adversary $\mathcal{A}_1$ that outputs a forged signature for the interactive signature scheme with a non-negligible advantage ϵ. We can use the algorithm $\mathcal{A}_1$ to construct a PPT algorithm $\mathcal{B}_1$ that can break the eCBDH$_1$ problem: for $x, y \in \mathbb{Z}_q^*$, given $G, H, G^x, H^x, H^y \in \mathbb{G}$, compute $G^{xy} \in \mathbb{G}$. Algorithm $\mathcal{B}_1$ is described as follows.

- **Setup** Given an eCBDH$_1$ problem $(G, H, G^x, H^x, H^y \in \mathbb{G})$, the algorithm $\mathcal{B}_1$ runs the $Setup$ to generate the msk and calculates $mpk = (g = G, h = H)$.
- **Learning** $\mathcal{A}_1$ can issue up to q_O owner-key queries and q_D dealer-key queries. In response, $\mathcal{B}_1$ runs as follows.
  - Owner-key queries. Given an owner's index i, $\mathcal{B}_1$ chooses a random number $\lambda_i \in \mathbb{Z}_p^*$. Assuming that $sk_i = y\lambda_i$, the public key is computed as $pk_i = (H^y)^{\lambda_i}$. $\mathcal{B}_1$ sends pk_i to $\mathcal{A}_1$.
  - Dealer-key queries. Given a dealer's index j, $\mathcal{B}_1$ chooses a random number $\zeta_j \in \mathbb{Z}_p^*$. Assuming that $sk_d = x\zeta_j$, the public key is computed as $pk_d = (G^x)^{\zeta_j}$. $\mathcal{B}_1$ sends pk_d to $\mathcal{A}_1$.
- **Hash Query** $\mathcal{A}_1$ can query a hash oracle up to q_H times as $H(T_i) = G^{h(T_i)}$, where there is a map: $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $h(T_i) \in \mathbb{Z}_p^*$.
- **Challenges** $\mathcal{B}_1$ runs the $WitGen$ to generate $W = (wit_1, wit_2)$. It computes $wit_1 = g^a = G^a, wit_2 = h^{ad} = (H^x)^a$, and sends W to $\mathcal{A}_1$ as a challenge.
- **Response** The adversary $\mathcal{A}_1$ calculates a message-signature pair (T_i^*, σ_i^*) in polynomial time, and then sends it to $\mathcal{B}_1$.
- **Output** $\mathcal{B}_1$ determines whether the pair is valid by examining the relation

  $$e(\sigma_i^*, h) \overset{?}{=} e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i^{1/\lambda_i}) \cdot e(H(T_i^*), wit_2).$$

  If this equation holds, this means that σ_i^* is a valid signature. Then, $\mathcal{B}_1$ computes $G^{xy} = (\sigma_i^*/(G^x)^{h(T_i^*)a})^{1/H(ID_i^*)}$ and returns this value as the final result.

We now analyze the validity of the above construction as follows. The final equation is used to check the validity of a forged signature according to

$$\begin{aligned} & e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i^{1/\lambda_i}) \cdot e(H(T_i^*), wit_2) \\ & = e((G^x)^{\zeta_j H(ID_i^*)/\zeta_j}, (H^y)^{\lambda_i/\lambda_i}) \cdot e(G^{h(T_i^*)}, (H^x)^a) \\ & = e(G^{xyH(ID_i^*)+h(T_i^*)xa}, H). \end{aligned}$$

In addition, the above algorithm $\mathcal{B}_1$ is a probabilistic polynomial time (PPT) algorithm only if the adversary $\mathcal{A}_1$ can return the result within polynomial time. This also means the PPT algorithm $\mathcal{B}_1$ can solve the eCBDH$_1$ problem with a non-negligible probability. This contradicts the hypothesis that the eCBDH$_1$ problem is hard for any PPT algorithm. That

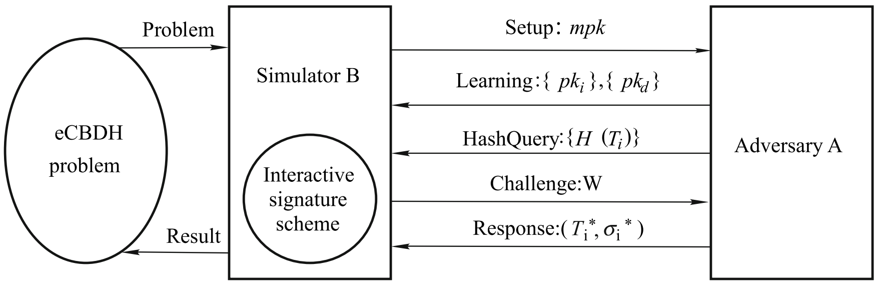


Fig. 9 Diagram of proof process in Theorem 1

is, the advantage of any probabilistic polynomial time algorithm $\mathcal{B}_1$ in solving the eCBDH$_1$ is negligibly small.

$$\begin{aligned} Adv_{\mathcal{B}_1}^{eCBDH_1} &= \Pr[\mathcal{B}_1(G, H, G^x, H^x, H^y) = G^{xy}] \\ &= \Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ (T_i^*, \sigma_i^*) \leftarrow \mathcal{A}_1(\{pk_i\}, pk_d, W) \end{array}\right] \leqslant \epsilon. \end{aligned}$$

The algorithm $\mathcal{B}_1$'s running time includes the running time of forgery. The additional overhead imposed by $\mathcal{B}_1$ is dominated by the need to evaluate group exponentiation for each signature, key request, and hash request. Any one such exponentiation may be computed by using at most $2\log p$ group actions [28], and thus, at most $2\log p$ time units on $\mathbb{G}$. $\mathcal{B}_1$ may need to answer as many as $q_O + q_D + q_H$ such requests, and therefore, its overall running time is $t' \leqslant t + 2(\log p)(q_O + q_D + q_H)$.

4.4.2 Incontestability of dealer

Similarly to the above definition, we also define the incontestability of the dealer as follows.

Definition 4 A signature scheme is $(t, q_O, q_D, q_H, \epsilon)$-secure against the incontestability of a dealer if any adversary $\mathcal{A}_2$ breaks the scheme with a negligible probability ϵ, the advantage is

$$Adv_{\mathcal{A}_2} = \Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ mpk \leftarrow Setup(1^\kappa), \\ \{sk_i, pk_i\} \leftarrow \mathcal{A}_2^{OKeyGen(mpk,\cdot)=(sk_i,pk_i)}, \\ pk_d \leftarrow \mathcal{A}_2^{DKeyGen(mpk,\cdot)=(sk_d,pk_d)}, \\ W \leftarrow WitGen(mpk, sk_d), \\ (T_i^*, \sigma_i^*) \leftarrow \mathcal{A}_2(\{sk_i, pk_i\}, pk_d, W) \end{array}\right] \leqslant \epsilon,$$

for t time, q_O and q_D times queries for owner's and dealer's keys, and q_H times queries for the hash oracle, where $\mathcal{A}_2^{OKeyGen(mpk,\cdot)=(sk_i,pk_i)}$ and $\mathcal{A}_2^{DKeyGen(mpk,\cdot)=(sk_d,pk_d)}$ denote the adversary's oracle queries for the owner's public/private keys and dealer's public keys, respectively.

By comparing this definition with Definition 2, it is easy to find the difference that the adversary $\mathcal{A}_2$ holds an owner's secret keys sk_i. This means that no owner can forge the signature of a dealer. The incontestability of a dealer in our proposed scheme is based on the eCBDH$_2$ assumption, which is defined as follows.

Definition 5 (eCBDH$_2$ Assumption) Given $G, H, G^a, G^b, H^{ab} \in \mathbb{G}$ for unknown $a, b \in \mathbb{Z}_q^*$, the eCBDH$_2$ assumption states that it is computationally intractable to compute the value of G^{ab}.

Based on this assumption, we prove our scheme provides incontestability of a dealer, as follows.

Theorem 2 Let $\mathbb{G}$ be a (t', ϵ') group for Diffie-Hellman of order p. Then, the signature scheme on $\mathbb{G}$ is $(t, q_O, q_D, q_H, \epsilon)$-secure against the unforgeability of a signature by a dealer, where

$$t \leqslant t' - 2(\log p)(q_O + q_D + q_H), \quad \epsilon \geqslant \epsilon'.$$

Proof Suppose there exists a PPT adversary $\mathcal{A}_2$ that outputs a forged signature for the above scheme with a non-negligible advantage ϵ. We can use the algorithm $\mathcal{A}_2$ to construct a PPT algorithm $\mathcal{B}_2$ that can break the CBDH$_2$ problem: for $x, y \in \mathbb{Z}_q^*$, given $G, H, G^x, G^y, H^{xy} \in \mathbb{G}$, compute $G^{xy} \in \mathbb{G}$. Algorithm $\mathcal{B}_2$ is described as follows.

- **Setup** Given an eCBDH$_2$ problem $(G, H, G^x, G^y, H^{xy} \in \mathbb{G})$, the algorithm $\mathcal{B}_2$ runs the $Setup$ to generate the msk and calculates $mpk = (g = G, h = H)$.
- **Learning** $\mathcal{A}_2$ can issue up to q_O owner-key queries and q_D dealer-key queries. In response, $\mathcal{B}_2$ runs as follows.
  - Owner-key queries. Given an owner's index i, $\mathcal{B}_2$ runs $OKeyGen$ to generate $(sk_i = x_i, pk_i = H^{x_i})$ for any x_i and sends (sk_i, pk_i) to $\mathcal{A}_2$.
  - Dealer-key queries. Given a dealer's index j, $\mathcal{B}_2$ chooses a random number $\zeta_j \in \mathbb{Z}_p^*$. Assuming that $sk_d = y\zeta_j$, the public key is computed as $pk_d = (G^y)^{\zeta_j}$. $\mathcal{B}_2$ sends pk_d to $\mathcal{A}_2$.
- **Hash Query** $\mathcal{A}_2$ can query a hash oracle up to q_H times as $H(T_i) = G^{h(T_i)}$, where there is a map: $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $h(T_i) \in \mathbb{Z}_p^*$.
- **Challenges** Assuming that $a = x$, $\mathcal{B}_2$ computes $wit_1 = g^a = G^x, wit_2 = h^{ad} = H^{xy}$. Let $W = (wit_1, wit_2)$ and send W to $\mathcal{A}_2$ as a challenge.
- **Response** The adversary $\mathcal{A}_2$ calculates a message-signature pair (T_i^*, σ_i^*) in polynomial time and then sends it to $\mathcal{B}_2$.
- **Output** $\mathcal{B}_2$ determines whether the pair is valid by examining the relation

  $$e(\sigma_i^*, h) \stackrel{?}{=} e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i) \cdot e(H(T_i^*), wit_2).$$

  This equation used to determine the validity of a forged signature holds as:

  $$\begin{aligned} &e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i) \cdot e(H(T_i^*), wit_2) \\ &= e((G^y)^{\zeta_j H(ID_i^*)/\zeta_j}, H^{x_i}) \cdot e(G^{h(T_i^*)}, H^{xy}) \\ &= e(G^{yH(ID_i^*)x_i + h(T_i^*)xy}, H). \end{aligned}$$

If σ_i^* is a valid signature, $\mathcal{B}_2$ computes $G^{xy} = (\sigma_i^*/(G^y)^{H(ID_i^*)x_i})^{1/h(T_i^*)}$, which means the PPT algorithm $\mathcal{B}_2$ can solve the eCBDH$_2$ problem with a non-negligible probability. This contradicts the hypothesis that the eCBDH$_2$ problem is hard for any PPT algorithm.

That is, the advantage of any probabilistic polynomial time algorithm $\mathcal{B}_2$ in solving the eCBDH$_2$ is negligibly small.

$$\begin{aligned} Adv_{\mathcal{B}_2}^{eCBDH_2} &= \Pr[\mathcal{B}_2(G, H, G^x, G^y, H^{xy}) = G^{xy}] \\ &= \Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ (T_i^*, \sigma_i^*) \leftarrow \mathcal{A}_2(\{sk_i, pk_i\}, pk_d, W) \end{array}\right] \leqslant \epsilon. \end{aligned}$$

The algorithm $\mathcal{B}_2$'s running time is similar to the above analysis of unforgeability of a signature by an owner and its overall running time is the same as $t' \leqslant t + 2(\log p)(q_O + q_D + q_H)$.

5 Performance evaluation

5.1 Performance analysis

Our interactive signature scheme is constructed on a bilinear map system from elliptic curve pairings. For simplification, we give several notations to denote the time required for various operations in our signature scheme. $E(\mathbb{G})$ is used to denote the exponentiation in $\mathbb{G}$ and B to denote the pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. We neglect the operations on $\mathbb{Z}_p^*$, the hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$, and the multiplication in $\mathbb{G}$ and $\mathbb{G}_T$, because they are considerably more efficient than the exponentiation and pairing operation. We analyze the computation and communication complexity for each phase, where $l_{\mathbb{Z}_p^*}$ and $l_{\mathbb{G}}$ denote the length of elements in $\mathbb{Z}_p^*$ and $\mathbb{G}$, respectively.

In Tables 1 and 2, the analysis of the performance of our interactive signature scheme is shown from two aspects: computation and communication/storage costs. Note that in Table 1 there is no exponentiation and pairing operation in the *setup* phase; thus, we do not list this algorithm here. In Table 2, we use sk to denote sk_i and sk_d since they both have the same element length. Similarly, we use pk to denote pk_i and pk_d.

Table 1 Complexity analysis of our scheme

Function	Computation complexity
OKeyGen	$1 \cdot E(\mathbb{G})$
DKeyGen	$1 \cdot E(\mathbb{G})$
WitGen	$3 \cdot E(\mathbb{G})$
SigGen	Dealer: $2 \cdot E(\mathbb{G})$ | Owner: $4 \cdot E(\mathbb{G})$
Verify	$1 \cdot E(\mathbb{G}) + 3 \cdot B$

Table 2 Communication/storage analysis of our scheme

Object	Communication/Storage complexity
Master public key (*msk*)	$2 \cdot l_{\mathbb{G}}$
Private key (*sk*)	$1 \cdot l_{\mathbb{Z}_p^*}$
Public key (*pk*)	$1 \cdot l_{\mathbb{G}}$
Witness (*W*)	$2 \cdot l_{\mathbb{G}}$
Signature (σ)	$1 \cdot l_{\mathbb{G}}$

5.2 Performance evaluation

We report our experimental results to demonstrate the performance of the interactive signature scheme. We built a simple demonstration program simulating the interactive process between two participants. This program was implemented in Java and built on the Java Pairing Based Cryptography Library (JPBC) library. In Table 3, the detailed data for the experiments are listed. We used a type A elliptic curve parameter to generate bilinear pairing. In type A pairing, let l_q be the length of a prime q and l_r be the length of the order r, where r is a prime factor of q + 1. We applied five elliptic curves parameters in the experiments, in which each of l_q, l_r has a different value: a_160 (l_q=512, l_r=160), a_200 (l_q=640, l_r=200), a_240 (l_q=768, l_r=240), a_280 (l_q=896, l_r=280), and a_320 (l_q=1024, l_r=320).

Table 3 Computational costs for different elliptic curve types

Type	Setup	OKeyGen	WitGen	SigGen	Verify
a_160	0.01324	0.02538	0.07569	0.15141	0.11066
a_200	0.02335	0.04598	0.13500	0.27503	0.20850
a_240	0.03725	0.07072	0.28572	0.47010	0.32450
a_280	0.06605	0.10510	0.32277	0.64518	0.48799
a_320	0.07756	0.14998	0.44662	0.91350	0.71197

5.3 Simulation results

Finally, we also present the simulation results that validate the instant confirmation model and show how our methods proposed in Section 3.3 and 4.2 improve on the Bitcoin?s consensus mechanism. Our simulation tests were built on the *Ethereum consortium blockchain* (see the Ethereum's website), which is an open-source blockchain network. This blockchain network runs on a small cloud based on the open-source *OpenStack* cloud platform [29, 30] with six enterprise level servers. Based on this cloud platform, our consortium blockchain consists of a set of shared bourses (the number of bourses is changed from 4 to 10), as seen in Fig. 1, and each bourse supervises virtual clients within the same virtual subnet. The clients are used to simulate the generation of transactions and the bourses are responsible for submitting transactions and governing the blockchain.

One interesting aspect of our consortium blockchain is the performance of the dealer election using the BA protocol. The simulation results show that the new dealer election protocol is valid for choosing one single next block generator for our blockchain environment, such that no fork of a blockchain was detected for two weeks in our test. However, we also observed that our proposed protocol incurs a large communication overhead. Moreover this overhead grows with the square of the number of bourses. Therefore, when the number of consortium members is smaller, the experimental results are better; however, with an increase in the number, the communication overhead grows larger and the period of block generation becomes longer.

Another interesting aspect is the effect of replacing the ECDSA signature with the proposed IIS. In the results of our simulation experiments, we observed that the influence of the runtime of the IIS was not significant, even if the number of consortium members increased. Although it increases the interactive process, the new signature scheme has little effect on the communication and computation overhead of the entire network, given that the blockchain structure is not significantly altered. This indicates that our IIS design achieves the desired results.

6 Conclusions

In this paper, we presented a new system framework in a consortium blockchain for implementing *Instant Confirmation with Incontestability*. To construct this system, we used the BA to provide a voting protocol for the next block generator. This voting protocol can be considered a new consensus mechanism to avoid block conflict and double spends. More importantly, as the core of our system, a new signature scheme, the IIS, to ensure that the transaction's sender can obtain the dealer's instant confirmation with incontestability, was also proposed. Our simulation results show that the voting protocol and the signature scheme were effective and efficient for a consortium blockchain.

Acknowledgements The authors are indebted to anonymous reviewers for their valuable suggestions. This work was supported by the National Basic Research Program of China (2013CB329601) and the National Natural Science Foundation of China (Grant Nos. 61370187 and 61472032), NSFC-Genertec Joint Fund For Basic Research (U1636104), and Joint Research Fund for Overseas Chinese Scholars and Scholars in Hong Kong and Macao (61628201).

References

1. Bogart S, Rice K. The blockchain report: welcome to the internet of value. Needham Insight, 2015
2. Barber S, Boyen X, Shi E, Uzun E. Bitter to better – how to make bitcoin a better currency. In: Proceedings of International Conference on Financial Cryptography and Data Security. 2012, 399–414
3. Karame G O, Androulaki E, Capkun S. Double-spending fast payments in bitcoin. In: Proceedings of ACM Conference on Computer and Communications Security. 2012, 906–917
4. Eyal I, Sirer E G. Majority is not enough: bitcoin mining is vulnerable. In: Proceedings of International Conference on Financial Cryptography and Data Security. 2014, 436–454
5. Chaudhary K, Fehnker A, van de Pol J, Stoelinga M. Modeling and verification of the bitcoin protocol. 2015, arXiv preprint arXiv:1511.04173
6. Zhu Y, Guo R, Gan G, Tsai W T. Interactive incontestable signature for transactions confirmation in bitcoin blockchain. In: Proceedings of the 40th IEEE Annual Computer Software and Applications Conference. 2016, 443–448
7. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. Consulted, 2008
8. Pilkington M. Blockchain technology: principles and applications. In: Olleros F X, Zhegu M, eds. Research Handbook on Digital Transformations. Cheltenham, UK: Edward Elgar, 2016
9. Sompolinsky Y, Zohar A. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. IACR Cryptology ePrint Archive. 2013
10. Lewenberg Y, Sompolinsky Y, Zohar A. Inclusive block chain protocols. In: Proceedings of International Conference on Financial Cryptography and Data Security. 2015, 528–547
11. Eyal I, Gencer A E, Sirer E G, van Renesse R. Bitcoin-NG: a scalable blockchain protocol. In: Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation. 2016, 45–59
12. Bracha G. An $O(\log n)$ expected rounds randomized byzantine generals protocol. Journal of the ACM, 1987, 34(4): 910–920
13. Cooper J, Linial N. Fast perfection-information leader-election protocol with linear immunity, In: Proceedings of the 25th Annual ACM Symposium on Theory of Computing. 1993, 662–671
14. Ostrovsky R, Rajagopalan S, Vazirani U. Simple and efficient leader election in the full information model. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing. 1994, 234–242
15. Russell A, Zuckerman D. Perfect information leader election in $\log^* n + O(1)$ rounds. In: Proceedings of the 39th IEEE Annual Symposium on Foundations of Computer Science. 1998, 576–583
16. Gilbert S, Kowalski D R. Distributed agreement with optimal communication complexity. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms. 2010, 965–977
17. King V, Saia J. From almost everywhere to everywhere: Byzantine agreement with $\tilde{O}(n^{3/2})$ bits. In: Proceedings of International Symposium on Distributed Computing. 2009, 464–478
18. Toueg S, Perry K J, Srikanth T. Fast distributed agreement. SIAM Journal on Computing, 1987, 16(3): 445–457
19. Lamport L, Shostak R, Pease M. The byzantine generals problem. ACM Transactions on Programming Languages and Systems, 1982, 4(3): 382–401
20. Bracha G. An asynchronous [(n-1)/3]-resilient consensus protocol. In: Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing. 1984, 154–162
21. Dolev D, Reischuk R, Strong H R. Early stopping in byzantine agree-

ment. Journal of the ACM, 1990, 37(4): 720–741

22. Cachin C, Kursawe K, Shoup V. Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography, In: Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing. 2000, 123–132
23. Braud-Santoni N, Guerraoui R, Huc F. Fast byzantine agreement. In: Proceedings of ACM Symposium on Principles of Distributed Computing. 2013, 57–64
24. Zhu Y, Ahn G J, Hu H, Ma D, Wang S. Role-based cryptosystem: a new cryptographic rbac system based on role-key hierarchy. IEEE Transactions on Information Forensics and Security, 2013, 8(12): 2138–2153
25. Zhu Y, Huang D, Hu C J, Wang X. From RBAC to ABAC: constructing flexible data access control for cloud storage services. IEEE Transactions on Services Computing, 2015, 8(4): 601–616
26. Su D, Lv K. A new hard-core predicate of paillier's trapdoor function. In: Proceedings of International Conference on Cryptology in India. 2009, 263–271
27. Su D, Lv K. Paillier's trapdoor function hides θ (n) bits. Science China Information Sciences, 2011, 54(9): 1827–1836
28. Boneh D, Lynn B, Shacham H. Short signatures from the weil pairing. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. 2001, 514–532
29. Zhu Y, Hu H, Ahn G J, Yu M. Cooperative provable data possession for integrity verification in multicloud storage. IEEE Transactions on Parallel and Distributed Systems, 2012, 23 (12): 2231–2244
30. Zhu Y, Ahn G J, Hu H, Yau S S, An H G, Hu C J. Dynamic audit services for outsourced storages in clouds. IEEE Transactions on Services Computing, 2013, 6(2): 227–238

Yan Zhu is currently a professor in the School of Computer and Communication Engineering at the University of Science and Technology Beijing, China. He was an associate professor at Peking University, China from 2007 to 2012. He was a visiting associate professor in the Arizona State University, USA from 2008 to 2009, and a visiting research investigator of the University of Michigan-Dearborn, USA in 2012. His research interests include cryptography, secure group computation, secure multi-party computing, and network security.

Khaled Riad is a lecturer at Mathematics Department, Faculty of Science, Zagazig University, Egypt. He has received his MS degree in computer science from Zagazig University in January 2011. He received his PhD degree from the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His research interests include cloud security, cryptography, dynamic authorization, and access control.

Ruiqi Guo received the BS degree from the Taiyuan University of Technology, China in 2014. She is a master student in the School of Computer and Communication Engineering at University of Science and Technology Beijing, China from 2014. Her research interests include cryptography, network security, and software engineering.

Guohua Gan received the MS degree in computer science from Harbin Engineering University, China in 2005. He is currently a PhD Student in the School of Computer and Communication Engineering at the University of Science and Technology Beijing, China. His research interests include cryptography, secure computation, and network security.

Rongquan Feng received the PhD degree in mathematics from the Institute of Systems Science, Chinese Academy of Sciences, China in 1994. He is currently a professor in Peking University, China. He was a postdoctorate fellow in Pohang University of Science and Technology (POSTECH), Korea from October 1995 to August 1997, and a visiting professor there from July 2002 to August 2003. His research interests are in the areas of algebraic combinatorics, cryptology and information security. He has published more than 90 papers on these fields. He is now an administrative committee member of Chinese Association for Cryptologic Research. He served as the secretary-general of Beijing Mathematical Society from 2005. He is an associate editor of the journal Mathematics in Practice and Theory and in the Editorial Board of Journal of Cryptologic Research.