**A Stable Consensus Protocol in Wireless Blockchain System**

**Abstract-** Recently, virous applications are constructed on blockchain, which has attracted attention because of its properties such as decentralization, security, immunity and traceability. Blockchain can solve the security and reliability challenges of wireless networks. However, most previous studies of blockchain consensus protocols in wireless network rely on efficient and stable transmissions. Nevertheless, nodes in wireless blockchain have limited physical resources, unreliable channels, and variable bandwidths influenced jamming attacks or environments. In this paper, we propose a novel Byzantine fault-tolerant consensus protocol for wireless blockchain network that does not rely on reliable leader-driven communication. Our protocol selects a block proposer randomly and uses threshold signature scheme as block proposal voting mechanism. Because only one block will be confirmed per round, our protocol can prevent the occurrence of conflicting blocks and blockchain forks. Besides, the protocol can guarantee security while tolerating peak up to faulty nodes among consensus nodes. Our protocol can also resistant to jamming attacks, double-spending attacks, and Sybil attacks, which are demonstrated by massive simulations.

# Introduction

With the rapid development of wireless communication technology and blockchain technology, many researches focus on the applications of blockchain technology in wireless field, such as mobile edge computing [1], intelligent 5G technology [2], Internet of vehicles [3], and others. Reliable and secure resource sharing services can be provided in distributed environment by using blockchain, which has received great attention from both academia and industry. Applications that built on wireless network face with significantly challenge of security and trust. Blockchain that is decentralization, persistence and traceability provide a new way to solve these problems. In this case, secure, trust and efficient services of data interaction, secure access control, data traceability, identity authentication in wireless field can be supported by integrating blockchain technology into wireless networks.

Currently, many studies on wireless blockchain system are directly enabling popular blockchain protocols that are deployed in the Internet to wireless network environment. Such blockchain protocols make use of consensus algorithm that always rely on massive resources consumption (e.g. Proof of Work [4]), complicated design (e.g. Proof of Stake [5]), and reliable communication (e.g. Practical Byzantine Fault Tolerant [6]). Although theses consensus algorithms work well in the Internet, they are not suitable for wireless networks with limited resources and unstable channel. The open communication of wireless networks is heavily impacted by environment. Both unstable channel bandwidth and vulnerable to Jamming attacks are the bottlenecks of wireless communication network. These barriers limit the application of combining traditional blockchain consensus algorithms and wireless networks, which is the motivation of researching blockchain protocol over wireless networks.

Recently, some researches on wireless blockchain systems leverage the natures of wireless networks to design efficient wireless blockchain consensus protocols. In order to adapt to the high dynamics of the mobile ad-hoc network, Z. Jiao et al. [7] design a PoW-based stability-aware consensus protocol, whose leader election is based on node information and proof of work. This novel design can make sure wireless blockchain system work efficiently and steadily. According to the characteristics of wireless communication network, Z. Jiang et al. [8] propose a Sybil-proof-based Byzantine fault-tolerant consensus protocol, which can realize real-time consensus in wireless networks. Considering the low-powered wireless devices and instability wireless transmission, Q. Xu et al. [9] propose an efficient and fair Proof-of-Communication consensus protocol in wireless blockchain system. And Y. Zou et al. [10] propose a fast consensus protocol for permissioned wireless blockchain system. This protocol can achieve k-times consensus in unreliable and multi-access wireless environment. Besides, to overcome the interference of wireless broadcast communication, M. Xu et al. propose a single-hop wireless blockchain consensus protocol under an adversarial SINR model BLOWN, which is based on a Proof-of-Channel consensus algorithm [11]. To solve the challenges of multi-hop wireless communication networks, M. Xu et al. design a fast fault-tolerant for wireless blockchain network wChain [12]. To accelerate data aggregation, this protocol constructs communication spanner by the maximum independent set. These blockchain protocols achieve consensus by either consuming massive resources or reliable interaction. In this way, the security of these protocols relies on the correctness of leader, which means malicious leader can interrupt consensus process arbitrarily.

To overcome the mentioned challenge of wireless blockchain system, we propose a blockchain consensus protocol that can reach consensus in instability wireless environment. Stable wireless blockchain consensus protocol is analogy with Proof-of-Stake consensus algorithm, which means nodes can achieve consensus without consuming massive resources. Our protocol combines verified random selection scheme and threshold signature scheme to make sure all nodes in wireless blockchain system can reach consensus randomly and steadily. Stable wireless blockchain consensus protocol operates round by round. In each round, a single block proposer is randomly and non-interactively selected according to nodes' probability, which is depended upon nodes' stability that defined by the lifetime and the number of recent generated blocks of nodes. In block proposer election phase, all nodes only know whether they become block proposer, but do not know who actually is elected as the block proposer. However, each node can verify the legitimacy of real block proposer independently. Such design can greatly reduce resource cost of block proposer election phase and the corruption risk of adversary. What's more, we adopt threshold BLS(Boneh-Lynn-Shacham) signature scheme to decouple block proposer and block verification and finalization phase. In this way, block finalization can be achieved by any node that obtain sufficient votes, not only rely on correct block proposer transmission. Such design can improve the stability and efficiency of consensus process, and reduce the risk of failure consensus due to fault node or instable wireless channel.

We make the following main contributions:

* We propose a new blockchain consensus protocol SWIB that combines verifiable random election scheme with threshold BLS signature scheme. Our protocol can ensure blockchain system stably generate block and reach consensus in unreliable and unstable wireless networks.
* We define node stability through the lifetime and the number of recent generated blocks of node. According to the stability of consensus nodes, our protocol can elect a quality node as block proposer randomly and verifiably. This way can reduce the corruption risk of adversary and improve the chance of generating valid block.
* To improve the robustness of SWIB, we use threshold BLS signature scheme to decouple block proposer with consensus process. In this way, even block proposer fails after broadcasting a new block, block finalization can be completed through a round of partial signatures exchanges. What's more, our protocol satisfies strong consistency that can efficiently prevent blockchain forks.
* We prove that when adversary controls less than 50% of voting power, SWIB guarantees persistence and liveness to wireless blockchain system.
* Finally, massive simulation studies are supported our theoretical analysis.

The rest of this paper is composed as follows. Section 2 introduces the most related works on state-of-art blockchain protocols, wireless consensus algorithms and threshold BLS signature scheme. Models and assumption of this paper is presented in Section 3. In section 4, we discuss the details of the stable wireless blockchain consensus protocol. Security analysis and performance analysis of our protocol is discussed in section 5. We report the result of our simulation in section 6 and give the conclusion of this paper in section 7.

# Related Work

## 2.1 Blockchain Consensus Protocols

We divide current popular blockchain consensus protocols into resource-proof-based consensus protocols and communication-based consensus protocols. We will briefly introduce blockchain consensus protocols category in this section, more detailed and comprehensive overview of blockchain consensus protocols has been introduced in [13].

Resource-proof-based consensus protocols require participants compete for block proposal right in each round through physical resources (e.g. computational power, memory, etc.) or virtual resources (e.g. shares, reputation, wealth, etc.). The most classical proof-of-physical-resources consensus algorithm is Proof-of-Work (PoW) [4], which is adopted by Bitcoin and Ethereum. In PoW consensus algorithm, nodes win the block proposal chance by solving a computational puzzle. However, this algorithm cannot provide instant consensus finality of blockchain protocol [14]. Actually, while adversary controls computing power is less than 50% of total power, multi-blocks confirmation can only guarantee probabilistic consistency of PoW-based blockchain consensus protocol. Due to the large time of generate a block, the block-confirmation latency of PoW-based blockchain protocol is large, and transaction throughput is limitation. In addition, there are some other physical-resource-proof-based consensus protocols include Proof of Space [15], in which consensus nodes compete for block proposal right through occupied memory or disk space; and Proof of Burn [16], in which consensus nodes obtain block proposal chance by burning another “coin”, such as Bitcoin. Physical-resource-proof-based consensus protocols require consensus nodes win block proposal chance by consuming huge physical resources, which lead to the massive waste of resources. As an alternative to physical-resource-proof-based consensus protocol, virtual-resource-proof-based consensus protocols can avoid large resources overhead. Proof-of-Stake (PoS) [5] is a typical consensus algorithm for virtual-resource-proof-based blockchain consensus protocols. Consensus nodes is elected as block proposer according to their holding stakes. The more stakes of nodes, the higher probability to be block proposer. The first version of Casper [17] is a hybrid consensus of PoW and PoS, aiming to replace the PoW consensus algorithm with PoS consensus algorithm in Ethereum. Proof-of-Reputation [18] is also a virtual-resource-proof-based consensus protocol, in which consensus nodes with enough reputation can obtain the right to generate a new block. Virtual-resource-proof-based consensus protocols do not consume physical resources, and is friendly to environment. Therefore, more and more blockchain consensus protocols would like to use proof-of-virtual-resource as consensus algorithm.

In communication-based blockchain consensus protocols, all consensus nodes reach consensus on block proposal by exchanging messages and performing local computation. Most of them can tolerate Byzantine fault and provide strong consistency. The very classical communication-based consensus protocol is practical Byzantine fault tolerant consensus protocol (PBFT) [19]. A block proposer is elected from all consensus nodes to propose a new block. And then the block proposer is responsible for communicating with other nodes to reach agreement on the block proposal. The cost of adversary being malicious in communication-based consensus protocol can be small, because consensus nodes do not equity mortgage or resource consumption. However, the cooperation mechanism of this protocol can eliminate the influence of malicious behavior to ensure blockchain system security. Some protocols are proposed to improve consensus performance to overcome the low scalability of communication-based consensus protocol. In NEO [20], partial nodes of delegated Byzantine fault tolerant consensus protocol are delegated to participant consensus process to reach agreement by voting on generated block. In this case, small number of consensus nodes can greatly increase consensus performance. In Zyzzyva [21], the modified BFT consensus algorithm SBFT employs threshold signatures to reduce communication overhead of consensus process. However, it still requires two-rounds communications to aggregate signatures and terminate block confirmation. In addition, Tendermint consensus algorithm of Cosmos [22] adopts leader rotation mechanism to avoid adversary corruption and use gossip protocol to improve the scalability of transaction propagation. Algorand consensus protocol [23] combines Byzantine agreement protocol with VRF committee election scheme to ensure the security and scalability of blockchain consensus process. The consensus security of most communication-based consensus protocols depends on the correctness of leader. All consensus nodes should change view when the leader fails and consensus process is interrupted. Besides, communication-based blockchain consensus protocols usually require reliable message transmission model, as well as make use of all-to-all broadcast communications. Therefore, this blockchain consensus protocol is more suitable for small network size and reliable communication scenarios.

## 2.2 Consensus Protocols for Wireless Networks

We briefly introduce the exist studies of wireless consensus protocols in this subsection.

Many consensus algorithms are proposed to improve consensus efficiency in wireless networks. Leverage the natural superposition property of wireless multiple-access channels, M. Zheng et al. [24] propose fast average consensus in clustered wireless sensor networks to achieve consensus within low times. In order to efficiently achieve a global consensus among nodes in clustered wireless network with respect to arbitrary initial states, M. Goldenbaum et al. [25] present an iterative gossip algorithm that based on the superposition property of wireless channel. C. Newport and P. Robinson [26] propose fault-tolerant distributed consensus algorithms to solve consensus problem of wireless systems through abstract MAC layer model. These consensus algorithms can guarantee consensus termination with high probability even there are any number of failures and no advanced information of network. To achieve finite-time max-consensus in a multi-agent system, F. Molinari et al. [27] present a switching consensus protocol according to the superposition property of fading wireless channel. Max-consensus can be achieved under this protocol within finite number of iterations. Besides, H. Moniz et al. [28] propose an asynchronous Byzantine consensus protocol for resource-constrained wireless ad hoc networks. Even some messages are lost dynamically, the protocol can efficiently achieve consensus.

In recent years, some studies combine wireless consensus algorithms with blockchain to design blockchain consensus protocols that are more suitable for wireless networks. According to the nature property of wireless broadcast communication, Z. Jiang et al. [8] propose a Sybil-proof-based Byzantine fault-tolerant consensus protocol, which can realize real-time consensus in wireless networks. Some studies increase consensus efficiency by making full use of wireless broadcast operation. Q. Xu et al. [9] propose an efficient and fair Proof-of-Communication consensus protocol in wireless blockchain system. And Y. Zou et al. [10] propose a fast consensus protocol that can achieve k-times consensus in unreliable and multi-access wireless environment. Besides, to overcome the interference of wireless broadcast communication, M. Xu et al. propose BLOWN that based on a Proof-of-Channel consensus algorithm under adversarial SINR model [11]. In addition, M. Xu et al. design a fast fault-tolerant wireless blockchain protocol wChain [12], which can quickly aggregate data and reach consensus in multi-hop wireless communication networks.

## 2.3 Threshold Signature Scheme

Blockchain consensus protocols that adopt threshold signature scheme [29] can quickly and steadily achieve consensus in wireless networks with unreliable and unstable communication channels. Threshold signature scheme allows a group of parties to constructing a signature without learning information about private keys. In a -threshold signature scheme, parties hold distinct key shares. And any subset of distinct parties can issue a valid signature, where as any subset or fewer parties can't. In blockchain consensus protocols, all consensus nodes generate partial signature shares to vote for block proposal through their private keys, and broadcast it to other nodes. Unique full signature will be recovered by any node who collects enough distinct partial signature shares of block proposal. In this way, even some messages loss or some nodes failing, block consensus can be achieved in blockchain system efficiently.

BLS signature scheme [30] utilizes cyclic group and bilinear mapping to construct aggregate signature, which used in multi-party signature and verification. The BLS signature scheme consists of signature generation algorithm and signature verification algorithm. Let be a cyclic group with prime order and generator . And let be a secure hash function. Tuple is considered as global information. Each party has a key pair , where is private key that can be used to compute public key. Signature of message can be computed as, where can ensure the integrity of messageVerifiers verify the validity of signature by checking whether is valid.

Threshold BLS signature (TSS) scheme [29] is derived from BLS signature scheme, and work in a non-interactive way. The partial signature generation of TSS is similar to BLS signature scheme, and the recovery of complete signature can be finished without interaction. Threshold BLS signature scheme includes key generation algorithm, signature generation algorithm and verification algorithm. The key generation algorithm adopts distributed key generation protocol [31] to distribute key pair and the aggregated main public key to participants. Participant can obtain other public key shares by exchange messages. The aggregated main public key is used to verify the validation of complete signature. Discrete log-based distributed key generation protocol is a common key generation algorithm of threshold BLS signature scheme. Let and be the order and generator of cyclic group, respectively. Tuple is the global information of threshold BLS signature scheme. The key generate protocol will randomly select a special value of -degree polynomial to generate main complete public key , which is usually used to verify complete signature. And then, each node will use a random value of the polynomial to generate a private-public key pair . Signature generation algorithm contains a partial signature generation protocol and a full signature recovery protocol. Among them, the partial signature generation protocol will generate partial signature of each node; the full signature recovery protocol will recover unique full signature, i.e. a Lagrange interpolation polynomial of partial signatures. A node can recover the unique value if it collects enough secret shares . Besides, the verification algorithm of BLS threshold signature scheme uses the main public key to verify the validation of the complete signature.

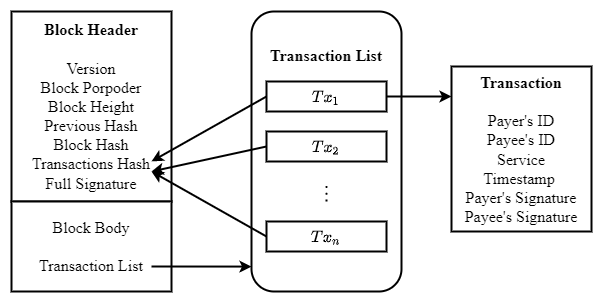
Stable wireless blockchain consensus protocol can use -BLS threshold signature scheme to guarantee the stability, security and efficiency of consensus process in wireless blockchain system. Since signature aggregation can be executed by all consensus nodes, full signature will be recovered even if some nodes fail to aggregate enough partial signature shares. Our protocol decreases the risk of single point failure by decoupling block proposer with consensus process. This design greatly improves the stability of wireless blockchain consensus protocol. In addition, our consensus protocol can significantly reduce communication overhead, because consensus process can be finished after one round communication.

# Models And Assumptions

In this section, we introduce the models and assumptions in this paper.

## 3.1 Blockchain Basics

In wireless blockchain system, we assume that each node locally maintains a blockchain, which is a hash-chain of blocks. Each block contains a set of transactions, which consists of some inputs and outputs that reference other transactions. We denote as blockchain, block and transaction, respectively. The data structure of block includes block header and block body. The body usually stores transaction meta data. And block header records version, block proposer, block height, previous hash, block hash, block full signature, and transaction hash. The format of a transaction contains payer' ID, payee's ID, services information, timestamp, payer' signature, payee's signature.



**Fig 1. Data structure of block and transaction**

## 3.2 Network Model

In this paper, we consider a wireless network consisting of distributed nodes that deployed in 2-dimentional Euclidean space. Let be the set of nodes, in which any pair of nodes can communicate with each other directly in the network. In practice, such network can be built on a group of Unmanned Aerial Vehicles or intelligent vehicles. Each node has a half-duplex transceiver, which used to transmit message or listen to wireless channel, but cannot do both. We further assume that any node can join the wireless blockchain network freely. After running a distributed key generation protocol, node will obtain its private-public key pair and main public key. Node can get other nodes' public keys and identities by exchanging messages. Due to SWIB relies on secure threshold BLS signature scheme, we assume that the number of honest nodes should satisfy the requirement of the secure threshold.

## 3.3 Communication Model

We assume that messages are transmitted in Rayleigh channel, which means that message transmission between nodes will be influenced by environment and wireless network interference. Through the characteristic of small-scale fading in wireless communication, the signal-to-noise-ratio of receive node should be

where is the uniform transmit power; is a random variable that represents the positive power gain of Rayleigh fading and follows the negative exponential distribution with parameter ; is the distance between two nodes, is the path-loss exponent; is the composite noise generated by the environment and adversaries. Let be wireless network signal-to-noise-ratio threshold that is determined by hardware. In a network area with radius , the probability density function of distance from transmit node to receive node is , the average probability of successful message transmission is

## 3.4 Adversary

Assuming that adversary can freely join or leave wireless network, and controls no more than of the total voting power. The malicious behaviors of adversary are as follows:

* Adversary can issue Sybil attack, that is, create pseudo identities. These malicious nodes will not transmit any valid messages or generate valid block in consensus process, or even transmit faulty messages to interrupt consensus process;
* Adversary can launch jamming attack to interfere with the message transmission of other honest nodes at any time. To leave chance for honest nodes to communicate, we assume that the capability of adversary is limited. In any interval of length rounds, adversary can jam no more than rounds, where and 0< ϵ ≤ 1. Each node in wireless network maintains an estimate of .

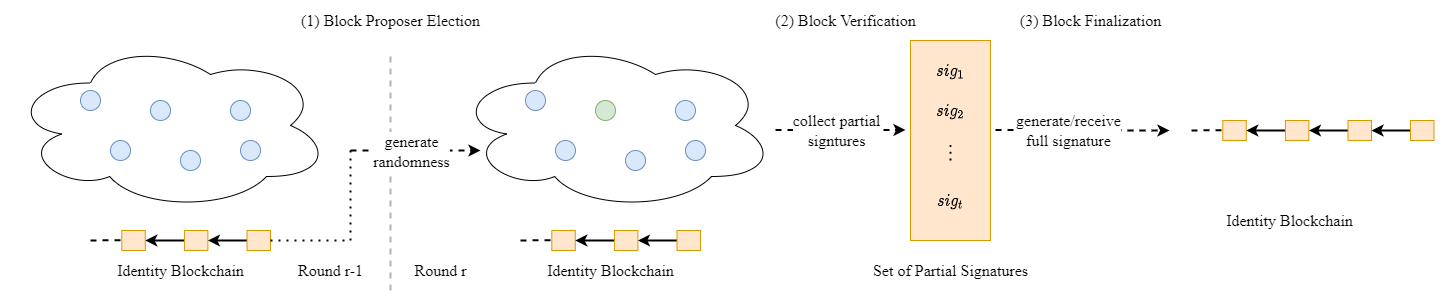
In this paper, if for any , event happens with probability at least then we can say event happens with high probability(w.h.p.).

# The Stable Wireless Blockchain Consensus Protocol

In this section, we present stable wireless blockchain consensus protocol, named SWIB. We first introduce the overview architecture of our protocol, and then discuss the details of the protocol.

## 4.1 Overview Architecture

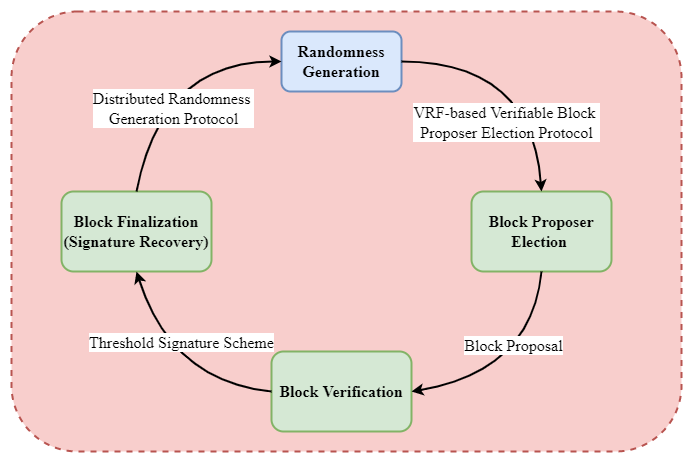
This subsection introduces SWIB protocol's design. Below we describe one round of SWIB protocol and show how it smoothly transition from round to round .



**Fig 2. Architecture Overview:** At the beginning of a round , all consensus nodes maintain identity blockchain. (1) run Block Proposer Election Protocol to randomly determine current round block proposer;(2) verify the legality of new block, and run signature aggregation algorithm to gather partial signatures; (3) finalize block through full signature which is recovered by running signature recovery algorithm.

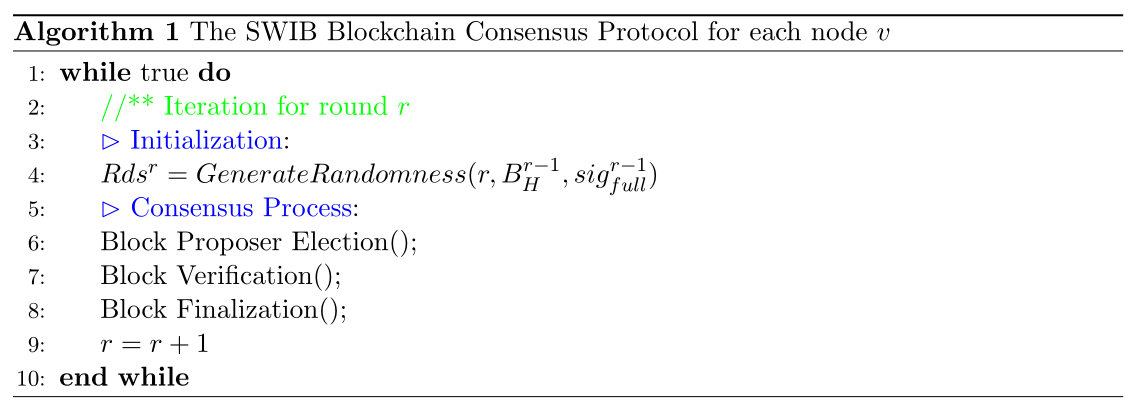
Our consensus protocol executes in disjoint and consecutive rounds sequentially. As shown in Fig 2, we star with a secure randomness generation. Permitting each node predicts the next round block proposer is insecure. Thus, we need a source of randomness to ensure that nodes will not know the information of block proposer preciously. In SWIB protocol, consensus nodes can create their identities to join blockchain system by submitting Sybil-resistant-proof. Thus, SWIB protocol uses a distributed randomness generation scheme to make sure that all nodes can generate a same randomness per round independently. According to the round randomness, a block proposer will be elected at the beginning of round through block propose election protocol. Then, the block proposer will generate a block and broadcast it to all other nodes. Each node will vote on the validity of the block by generating partial signature through a partial signature generation protocol. Once aggregating a threshold of partial signature shares, any consensus node can recover the full signature to finalize block through a signature recovery protocol.

We aim to build a protocol that hundreds of nodes can achieve consensus in wireless network with unreliable and unstable channels. As shown as Fig 3, the main components of our protocol are:



**Fig 3. Main components of SWIB**

* Block proposer election: A block proposer will be elected from all consensus nodes. Each node independently generates randomness in interval through a distributed randomness generation scheme. Nodes use their private key and the round randomness to check whether to be current round block proposer through verifiable block proposer election protocol. Besides, each node can verify the legality of block proposer according to its public key. The elected block proposer packs transactions from transaction pool to generate a new block, and disseminate it to all other consensus nodes in wireless network.
* Block validation: After receiving the proposed block, other nodes will verify the validity of block. Each node will generate a partial signature of block hash if the result of the verification is true. And then, each node will broadcast their partial signature to other nodes.
* Block finalization: A full signature will be recovered if a node aggregates enough partial signature shares. This full signature can be seen as the proof of block finalization, and will be broadcast to other consensus nodes. the conditions of block finalization are: 1) collect enough partial signature shares and recover the full signature; 2) receive the valid full signature of block hash. When nodes receive or generate full signature, they will append block into their local blockchain and compute a new randomness for next round through block hash and the full signature.



Algorithm 1 shows the process of a complete round in SWIB protocol. The precise details of every stage are introduced in the following subsections. We solve the challenges of block proposer election and block verification as well as finalization in consensus protocol to ensure the security and stability of wireless blockchain system. Even adversary occupies some nodes, our protocol can work smoothly in a wireless blockchain system.

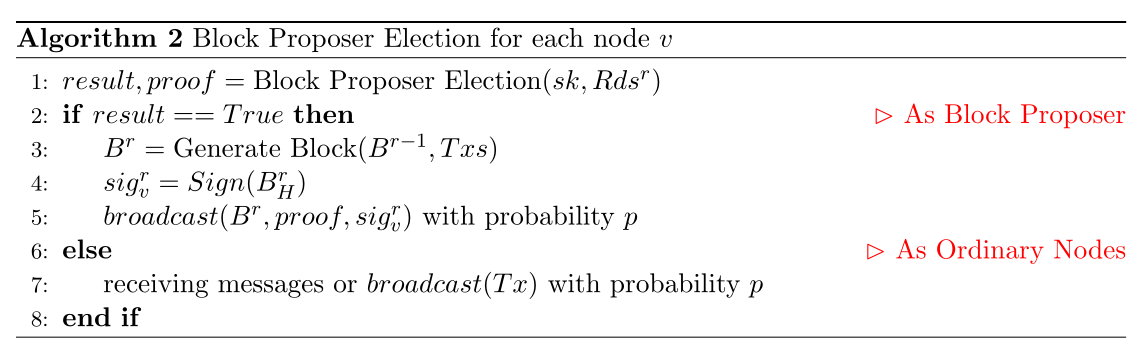
## 4.2 The SWIB Protocol

In this subsection, we discuss the details of the SWIB protocol.

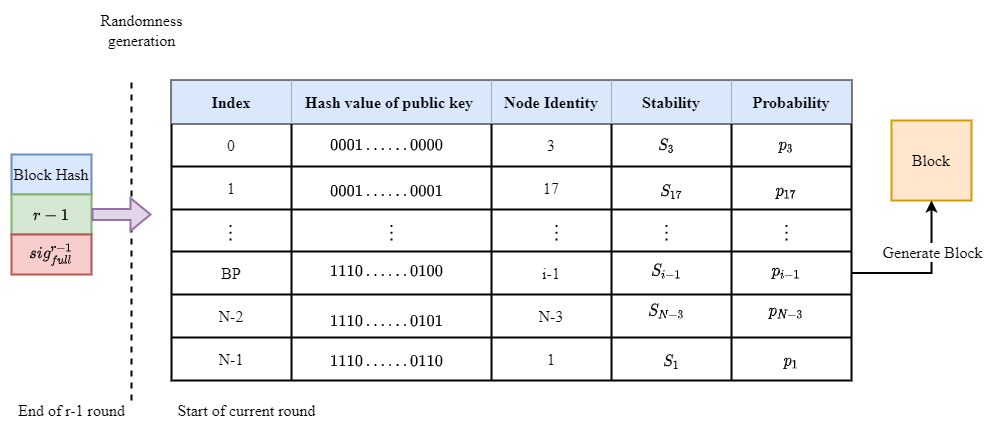
Before participating consensus, each node requires to obtain some information of blockchain system. Such as the identities and public keys of other consensus nodes as well as necessary blockchain history. Besides, each node needs to generate key pairs and the main public key through a distributed key generation algorithm. Our protocol uses three important functions of threshold BLS signature scheme: a signature generation function to generate partial signature; a signature recovery algorithm to reconstruct the full signature from a threshold of partial signature shares, as well as a signature verify function to check both partial signature and full signature.

### 4.2.1 Block Proposer Election and Proposal

We present a verifiable block proposer election protocol, which is based on verifiable random function, BLS threshold signature scheme and distributed randomness generation protocol. Algorithm 2 shows the process of block proposer election.

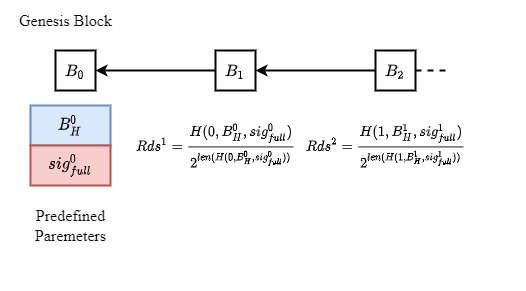


As shown in Fig 4, block proposer will be elected according nodes' election probability and current round randomness. Each node executes distributed key generation protocol to obtain a private-public key pair and main public key . All nodes will exchange the public key before entering consensus process. And each node will maintain a public keys list , which can be seen as the identities of other nodes. To ensure same view of nodes list, all node will sort the list according to the hash value of public keys. Consensus nodes will independently check whether they are the current round block proposer.



**Fig 4. The block proposer election for round .**

The election probability of each node is mainly determined by the stability of node in stable wireless blockchain consensus protocol. We first give the definition of node stability. Let be the remaining lifetime of node , then the sum lifetime of all consensus nodes should be . The lifetime ratio of can be denoted as . The consensus ratio of node is denoted as , where is the number of blocks generated by in the latest blocks of blockchain. When the length of blockchain is smaller than , the consensus ratio of node should zero. We represent the stability of as , where and are respectively weight coefficients of lifetime ration and consensus ratio. According nodes' stability, we can define the elected probability of node as . Our election protocol is designed to elect current block proposer through the probability.



**Fig 5. Randomness generation of per round.**

Block proposer election protocol adopts randomness seed to guarantee the security of election process. The distributed randomness generation protocol can enable all consensus nodes to jointly generate a round randomness with unbiased and unpredictable manner. To ensure the uniqueness and immediacy of the input of the protocol, we use the full signature of current round as a randomness source, and combine it with block hash and round number . As shown in Fig 5, the randomness is performed as the normalized hash value of the combination:

For simplicity, the very first randomness is set to be the normalized hash value of the genesis block of blockchain.

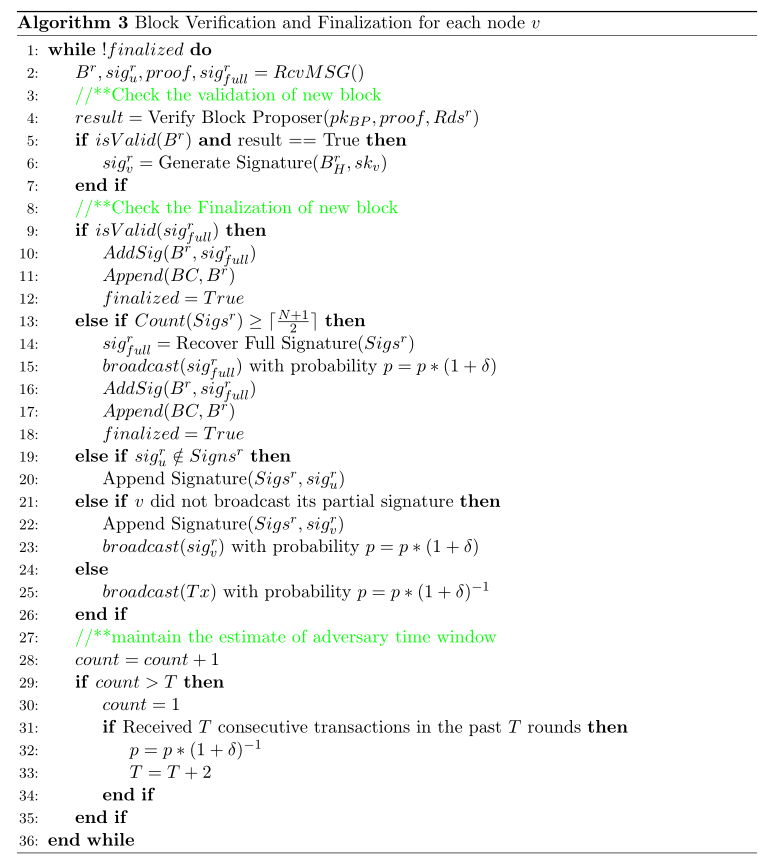
The output of the distributed randomness generation protocol is unpredictable and uniqueness. The recovery process of full signature for each round is unpreventable, due to the assumptions that majority nodes are correct. Thus, the full signature will be generated by correct node or received by other consensus nodes with high probability after finite times. Although block hash and round number are advanced known information, the full signature can only be revealed at the end of round. If a node first recovers full signature before others, it cannot tamper the result of recovery. Thus, anyone cannot predict the output, which is always consistent because the randomness is calculated based on common inputs. Another benefit of distributed randomness generation protocol is that nodes can enter following round non-interactively. Upon receiving or generating the valid full signature, consensus nodes will append block into their local blockchain and start next consensus process by computing the following round randomness.

The verifiable block proposer election protocol consists of a block proposer election algorithm and a block proposer verification algorithm. Block proposer election algorithm ensures that the election process is random, verifiable, and secure. To determine the index of current round block proposer, the algorithm divides interval into consecutive intervals as where is the elected probability of node . If the round randomness , then the node whose public key with index will be the block proposer of round . After executing block proposer election algorithm, block proposer will obtain a proof and the election result while others will only get the election result and a common string. Therefore, each node can only know whether they are elected as block proposer by their private key and round randomness. Meanwhile, block proposer verification algorithm guarantee that consensus nodes can verify the legality of block proposer by the public key and proof of block proposer as well as the current round randomness.

The verifiable block proposer election protocol is built upon secure and robust threshold BLS signature scheme, which can tolerate any malicious nodes among . The secure threshold of threshold BLS signature scheme can be satisfied when majority consensus nodes are correct. Therefore, a block proposer can be elected sequentially while the full signature is always reconstructed in each round.

The node who elected as current round block proposer will pack transactions to generate a new block. The block is divided into block header and block body. The block header is represented by tuple , where is the round number, is the hash of previous block, is the block hash, and is the identities of block proposer. The block body usually store transaction metadata. The block proposer will broadcast the block and signature of the block hash to other nodes. The full signature will be append in this block when a node gathers enough partial signature shares and reconstructs the signature.

### 4.2.2 Block Verification and Finalization



As, shown in Algorithm 3, the block verification and finalization depend on secure threshold signature scheme. We propagate blocks and the signatures to consensus nodes through wireless network broadcasting. When receiving a new block from block proposer, a node evaluates the validity of the block through the following components:

* Block proposer: The result of block proposer verification algorithm that uses the public key and proof of the block proposer as well as current round randomness as inputs should be true.
* Previous hash : The previous hash has to be equal to the hash of the block, which was confirmed during the previous round.
* Transactions: All transactions included in the proposed block should not conflict with previous confirmed transactions.

As depicted in Fig 6, if all mentioned conditions are satisfied, a node will then generate partial signature of the block hash and broadcast to other nodes. And when a node aggregating valid partial signature shares of the block hash to recover the full signature, which is the proof of block finalization. The reconstruction of valid full signature proves that a threshold of nodes signed block hash, which means that a sufficient number of nodes vote for block validity. Therefore, it is feasible to use the full signature as the signal of block finalization. Moreover, since correct nodes can only sign a block once in a round, only one block will complete the verification and finalization process. In this case, SWIB protocol can make sure that only one block will be confirmed per round, which preventing the occurrence of chain fork.



**Fig 6. Block verification and finalization**

As we discussed, full signature recovery can be done independently by any correct node, which increasing the stability of consensus process. In this way, any node who aggerates enough partial signature shares can recover the full signature. Even if malicious nodes reject to recover full signature or broadcast valid full signature to other nodes, some honest nodes will reveal block finalization too. This design enables that block finalization will be stably achieved in wireless network with faulty nodes or unreliable communication channels.

### 4.2.3 Incentive and Punishment Mechanism

We design an incentive mechanism to improve the enthusiasm of consensus nodes generate signature. Both verifying block and generating signature will consume the computational power of consensus nodes. Rational nodes who are not block proposer would be more willing to wait for full signature to enter new consensus process round, rather than wasting computational power to verify block and generate signature. Therefore, an incentive mechanism is required to motivate consensus nodes verify and finalize block. The reward in blockchain system are only the submitted transaction fees of valid blocks. In our incentive mechanism, a part of transaction fees is rewarded to block proposer, and the rest of transaction fees will be averagely distributed to the nodes whose partial signature shares are used to recover the full signature.

Transaction fees will be shared by these nodes whose average timestamp of partial signature is the smallest. Because the uniqueness of threshold BLS signature scheme, any different partial signature shares can recover the same full signature. We only reward a set of consensus nodes that have signed block hash quickly. Let and be the set of partial signature shares and the set of corresponding timestamps, respectively. Assuming there are two different partial signature shares sets and , and the corresponding timestamps sets are and . The average timestamps of the two sets are . If , then consensus nodes whose partial signatures in set will share the reward. This incentive mechanism not only encourages nodes to verify block and generate signature, but also motivates nodes to broadcast partial signature and full signature as soon as possible. The earlier broadcasting signature, the higher chance to get reward. In this way, the performance of blockchain system will be improved.

In addition, we use a punishment mechanism to reduce the opportunity of rational nodes become malicious. The malicious behaviors of nodes contain: 1) when becoming the legal block proposer, node usually generates invalid block or not generate any block within timeout; 2) node usually send invalid signature or garbage messages in block verification and finalization phase. Once nodes are judged malicious, we will punish to reduce their lifetime. This measure will decrease the stability of nodes, which will reduce the probability that node is elected as block proposer to obtain reward. As result, the security and efficiency of blockchain system will be improved, because rational nodes would prefer to follow the consensus protocol to obtain reward than become malicious.

### 4.2.4 Synchronization Mechanism

In our protocol, we consider a more efficient and secure self-initiate mechanism. When a new node joins blockchain system, it is necessary to get information of other consensus nodes and necessary blockchain history before participating in consensus process. Node will request some blocks of necessary blockchain history from nodes with high stability. Nodes with higher stability are more likely to maintain the latest blockchain, and have lower probability of transmitting fault blocks.

When the total number of blocks of blockchain history is , a node can request different blocks from different neighbors. Suppose that a node can successfully receive a block from another node with consistent probability within , which is the time of a slot. The probability that a node success in receiving a block after is . In this way, a node can successfully receive blocks with high probability after slots. Such design avoids that single node absences consensus process long time due to transmit whole blockchain history. Meanwhile, it can also reduce the risk that single node transmits stale blockchain history to new node. Once a node transmits wrong blockchain history, the receive node can immediately detect through transmitted blocks from other nodes.

# Protocol Analysis

In this section, we discuss the security and performance of SWIB protocol.

## 5.1 Security Analysis

We conduct a security analysis to show that our protocol provides persistence and liveness to wireless blockchain system. With majority consensus nodes being honest, our protocol satisfies:

* Persistence: If an honest node proclaims block is valid, then other honest nodes also agree on that the validity of block .
* Liveness: All honest nodes eventually commit a block, and generate the following round randomness.

SWIB protocol can guarantee the security of consensus even if failures happen in blockchain system. Nodes failure can be roughly classified into crash failure and Byzantine failure. When crash failure happens, nodes will not send messages or respond to any request. Besides, nodes may send some error messages to mislead other nodes while Byzantine failure occurring. The security of our protocol relies on the secure threshold BLS signature scheme, but not the correctness of block proposer. In our protocol, block proposer is only responsible for generating block. If block proposer generates an invalid block or not generates block, all consensus nodes also can securely proceed to the following round. There are two possible outputs at the end of a round: a finalized valid block or a finalized empty block. A valid block is confirmed if the following two condition are satisfied: 1) the block proposer honestly generate a valid block and 2) majority correct consensus nodes vote for it. If one of the conditions is not fulfilled, an empty block will be finalized eventually. However, block finalization requires a threshold of partial signature shares to recover the full signature. Therefore, the number of honest nodes should satisfy the requirement of threshold BLS signature scheme. Threshold signature scheme is secure, because its output is unforgeable and robust. The most important properties of the threshold signature scheme are:

* Uniqueness: Any set of more than partial signature shares can recover the full signature. That is, the recovery result of any enough partial signature shares is always the same.
* Verifiability: The full signature can be verified by anyone who using the unique main public key. The public key is generated and distributed by the distributed key generation protocol when node apply to enter consensus process.

### 5.1.1 Persistence Analysis

If there is a quorum of nodes to fulfill the requirement of threshold BLS signature scheme, out protocol can guarantee persistence:

**Theorem 1. (Persistence)** If an honest node proclaims block is the th block of blockchain, then other queried honest nodes should report the same result.

**Proof.** In order to prove the persistence property, we need to show that for any two blockchains and of honest nodes and should have same block in the same position. To prove by contradiction, we assume that and are two different blocks that at the position of and that maintained by nodes and , respectively. There are two cases when holding the assumption:

Case 1. and are respectively appended to the position of blockchains and at the same round. Block finalization depends on the unique full signature, which is recovered by at least valid partial signature shares. If two different blocks and are finalized in the same round, it indicates that the threshold signature scheme recover two different full signatures in a round. This conflicts with the uniqueness property of threshold signature scheme, which means contradicting our assumption.

Case 2. and are appended to blockchains and respectively in two distinct rounds. Assuming that and respectively are append to the position of blockchains and at round and . According to our protocol, block is finalized in round if the number of nodes voting for the block should be more than . Since two different blocks cannot be finalized in the same round, at least nodes will agree that is the th block of their blockchains in round . Using contradiction, we assume that Since , node have crash before round and recover in round so that block is appended into when node updates its blockchain. In this case, at least nodes have same view on th block is at the end of round . Since there are also at least nodes agree on the th block in round is and the network size is , we have contradiction that . Hence , which contradicts the mentioned assumption that

In summary, all honest nodes are queried for a special position block should respond the same result. In this case, if an honest node says block is valid, then other honest nodes also agree on that the block is valid.

### 5.1.2 Liveness Analysis

For liveness, we need correct nodes that have same view to prevent the protocol from stalling.

**Theorem 1. (Liveness)** Even if there are faulty nodes present in blockchain system, honest nodes can terminate a block and obtain the following round randomness seed.

**Proof.** The best case is that no failures occur in blockchain system. Our protocol can always finalize a valid block by recovering full signature, which can be the round randomness seed of the following round. We require to prove that our protocol can guarantee liveness under the influence of adversary. The behaviors of malicious nodes that controlled by adversary include: 1) intentionally not generating a valid block and 2) refusing to vote for the valid block. There are two cases for node failures:

Case 1. Block proposer is malicious. If a malicious block proposer not generates any block, all honest nodes will timeout and commit on an empty block. If a malicious block proposer generates an invalid block, the block will fail to pass the block validation process. Thus, all honest nodes will refuse to vote for the invalid block and commit an empty block finally. Each honest node is willing to generate partial signature of empty block hash. Once the full signature of the block hash is reconstructed from enough partial signature shares, the round randomness will be generated successfully.

Case 2. malicious nodes refuse to vote for a block. In this case, the rest of honest nodes would always generate partial signature shares of the block hash, and broadcast to other nodes. Once collecting a threshold of partial signature shares, any honest node can recover the full signature and broadcast to other nodes. Nevertheless, the rest honest nodes would also finalize block and generate the following round randomness seed-full signature when nodes failures happen.

In summary, if the number of faulty nodes is at most , all honest nodes of our protocol finally can always commit on a block and generate the following round randomness.

### 5.1.3 Stability Analysis

Our verifiable block proposer election protocol is based on a robust threshold signature scheme and verifiable random function. As long as honest nodes take part in the signature aggregation process, small part of malicious nodes cannot hinder the generation of a valid full signature. With the robust randomness, adversary cannot corrupt the process of block proposer election. Therefore, verifiable block proposer election protocol can guarantee the block proposer is elected unpredictably and unbiasedly. However, it does not ensure that the elected block proposer is still work honestly. If an adversary has less than 50% opportunity to be elected as block proposer per round, the probability that the adversary controls consecutive block proposer election is . Let be the upper bound of the probability. Given , the adversary can at most control consecutive rounds to become block proposer. This indicates that the probability of adversary controlling more than consecutive rounds is less than , which can be neglected. Therefore, our verifiable block proposer election protocol can make sure that adversary cannot always control the election of block proposer.

The number of blocks that used to calculate consensus ratio should far greater than . If the value of is too small, the consensus ratio of adversary will be high when it consecutively generates blocks. In this case, the elected probability of adversary might be very high, which means blockchain system may always commit on empty blocks. Therefore, the performance of system will be reduced. If we set large value of , the elected probability of adversary will not increase too much due to generating consecutive blocks.

### 5.1.4 Sybil Attack Analysis

Our protocol can efficiently prevent rational nodes issue Sybil attack to affect the block proposer election process of SWIB protocol. Adversary usually launches Sybil attacks by generating pseudonyms. We assume that the total probability of an adversary being block proposer is . If it generates pseudonyms, whose probabilities respectively are . The expected probability of adversary that creates multiple pseudonyms is . This indicates that generating multiple pseudonyms will decrease the elected probability. As result, rational nodes are motivated to not issue Sybil attacks to create too many pseudonyms.

Our protocol can ensure the security of block verification and finalization processes when adversary launching Sybil attack. Since adversary can control nodes with pseudonyms not send messages or send some error messages, the process of reaching consensus may be affected. Once a Sybil node becomes block proposer, it may generate an invalid block or not generate any block. All honest nodes will commit on an empty block and finalized the empty block eventually. Sybil nodes will refuse to vote for a valid block if they are not current block proposer. Due to the number of Sybil pseudonyms is bounded by , the remaining honest nodes can terminate block finalization and recover the full signature. Therefore, our protocol is resistant to Sybil attack as long as adversary controls less than the threshold of voting power.

### 5.1.5 Jamming Attack Analysis

Our protocol can smoothly operate when adversary issuing jamming attacks. Adversary can influence consensus process by jamming the message propagation of honest nodes. If honest nodes always cannot aggregate enough partial signature shares to recover full signature in time, the liveness of system cannot be guaranteed. To solve the problem, we make each node to maintain an estimate of adversary time window through sensing wireless channels. According to the estimate, each node can dynamically adjust timeout to ensure collection of partial signature shares. As result, SWIB protocol can terminate consensus process in time even adversary issues jamming attack. Such design can ensure the liveness of blockchain system under adversary issuing jamming attack, further ensuring the security of wireless blockchain system.

**Theorem 3**. If adversary is -bounded at any time interval of length slots, our protocol can successfully finalize a block by recovering full signature within slots w.h.p., where is the threshold, and is the successful probability of nodes transmit message in a slot.

**Proof**. Assuming that adversary can jam within slots, and is the successful probability of propagate message between nodes within a slot. In normal operation, full signature can be recovered within slots w.h.p.. A consensus node can successful transmit partial signature to another node in slots w.h.p.. Therefore, after slots, consensus node can gather partial signatures with high probability. In this case, this node can reconstruct full signature and confirm block. Since adversary launching jamming attack, honest nodes can only communicate slots per slots. Thus, a node with high probability can receive a partial signature from another node after slots. As result, after slots, a consensus node with high probability can confirm block by recovering the full signature.

## 5.2 Performance Analysis

In this subsection, we will analysis the performance of our protocol detailly. We will analysis the system computation overhead and communication overhead of put protocol to discuss the system overhead.

In stable wireless blockchain consensus protocol, a block is considered as valid if it obtained more than votes from consensus nodes. The probability that a node succeeds in receiving a message from another node is , then probability that a node succeeds in receiving at least messages is . Therefore, the probability that successfully generating a valid block in a round is , where is the number of nodes that generate or received full signature. Once a consensus node recovers the full signature, a block is confirmed to be valid and can be append to blockchain. We can calculate the probability of generating an empty block as .

We will study the system overhead of our protocol without considering malicious behaviors. Denote as the average time of generating a normal block. In our protocol, the time to generate a block includes: 1) time of generating block; 2) time of broadcasting block; 3) time of verifying block and signature; 4) time of generating signature; 5) time of broadcasting signature. Let be broadcast latency, and be the time of generating and verifying block, as well as be the time of generating and verifying signature. The average time of generating a valid block can be . The time of generating an empty block can be represented as .

### 5.2.1 Communication overhead Analysis

We measure the communication overhead through the required average communication times of generating a valid block.

In our protocol, the arrival rate of transactions is , and the number of arriving transactions whin time should be . The communication times of generating a valid block is calculated as . If there is no valid block is confirmed within , our protocol will finalize an empty block within time . The increase number of communications of generating an empty block is . Thus, the communication time of generating an empty block is represented as . Let be communication times of generating a valid block, and be communication times of generating an empty block. It is assumed that empty blocks have been generated before generating a valid block. In this case, the average communication times of generating a valid block is computed as

### 5.2.2 Computational overhead Analysis

We measure the computational overhead through the required average hash times of generating a valid block.

Let the computational power of nodes be , and be the time of generating and verifying an empty block. As we defined in earlier, is the time of generating and verifying block, as well as is the time of generating and verifying signature. The hash times of generating a valid block is ; and the hash times of generating an empty block is . Assuming that empty blocks have been finalized before generating a valid block, the average hash times to generate a valid block is calculated as

# Simulation Result

In this section, we conduct simulation experiments to evaluate our protocol. The impacts of various parameters are surveyed, including block size, network size and bandwidth. We evaluate the performance of our protocol through consensus latency and transaction throughput. The consensus latency is the time of running a round. The transaction throughput if the average number of confirmed transactions per second. All experiments are performed under Windows 10 operating system running on a machine with an Intel i7-10700F Core, 2.90GHz CPU, 8.00 GB RAM. All the reported results are the average of 10 runs.

## 6.1 Weight Coefficient

Before analyzing the performance of our protocol, we first require to determine the weight coefficient of nodes stability.

Block proposer election protocol randomly determines a block proposer through the stability of nodes. According to the formulation of stability , we believe that nodes with large lifetime ratio and consensus ratio are more trustworthy. Nodes that have invested much more deposits and generated some valid blocks are more willing to follow the protocol to obtain rewards of generating valid blocks. Besides, wireless nodes with lower stability have lower cost to be malicious, which indicates that nodes with higher stability have higher probability to generate a valid block and maintain real block history. However, only finalizing a valid block can guarantee the efficiency of processing transactions in blockchain system. Therefore, it is necessary to select a quality node to generate a valid block, even though the security of our consensus protocol node relies on the correctness of block proposer. Both lifetime ratio and consensus ratio of node can affect the elected probability of quality block proposer. In order to make sure the high probability of quality nodes, we analyze the probability of selecting quality nodes with different weight coefficients. Experiments show that both lifetime ratio and consensus ratio have significant impact on the election of block proposer.

We consider a blockchain system with network size , set that nodes with lower lifetime ratio and consensus ratio; nodes with lower lifetime ratio and higher consensus ratio; nodes with higher lifetime ratio and lower consensus ratio; nodes with higher lifetime ratio and consensus ratio. To investigate the impact of parameter , we measure the percentage of electing high-quality nodes as block proposer within 100 consensus times.



Fig 7

As shown in Fig 3, when , the percentage of electing a high-quality node as block proposer should be high. The experimental results show that the lifetime ratio and consensus ratio of nodes will affect the probability of selecting high-quality nodes, and the importance is similar. Therefore, the weight coefficients of the two factors should be as approximate as possible. In the subsequent performance experiments, the weight coefficient of stability is set to a = b = 0.5 to ensure that high-quality and stable block proposer is selected as soon as possible in block proposer election phase.

## 6.2 Block Size

Block size can affect consensus latency, which is the time of taking to complete a full round of our protocol. We can measure transaction throughput of blockchain system through used block size and resulting consensus latency.

We analyze how block size used in consensus protocol impacts the consensus latency. We run the consensus protocol for 10 consecutive rounds increase block size from 0.5MB to 5MB, and repeat the same experiments with four fixed-size networks. The network bandwidth limitation of all our experiments is set to 35 Mbps.



Fig 4

As shown in Fig 4, When network bandwidth is limited to 35 Mbps, the consensus latency will linearly increase when transmitting large blocks. When block size is 4MB, the average consensus latency is about . Precise result show that block transmission accounts for more than 90% of the overall confirmation time, indicating that block transmission is the main bottleneck of consensus protocol. In addition, with the increase of block size, the average throughput first increases rapidly, and finally tends to be flat. The results show that only increasing block size cannot improve the average transaction throughput indefinitely. When the block size is equal to 1MB, the protocol can achieve big average transaction throughput with small average consensus delay. The result shows that we can handle a large block size to ensure higher transaction throughput within reasonable consensus latency.

## 6.3 Number of Nodes

We analyze the impact of the number of consensus nodes on consensus latency and transaction throughput. We run the protocol for 10 consecutive rounds with network sizes ranging from 100 to 1000, and replicated the similar experiments with four different block sizes.



As shown in above figures, consensus latency will increase slowly with the number of consensus nodes, and achieving acceptable values even for network size of 1000 nodes and block size of 1MB. Due to the broadcast nature of wireless communication, block transmission time with same network bandwidth restriction will be similar in different network sizes. However, block finalization requires enough signatures, which will grow proportionally with the increase of network size. This indicates that all consensus nodes requires to verify more partial signatures, and the recovery of full signature requires larger threshold. Therefore, consensus nodes need more time to collect and verify partial signatures per round, and full signature aggregation need much more time too. Since the block size is fixed and consensus latency increases, the average transaction throughput will reduce. The result shows that average transaction throughput can peak up 4000TPS even for network size of 1000 nodes. Thus, our protocol can scale for large-size network.

## 6.4 Bandwidth

We analyze the impact of bandwidth on consensus latency and transaction throughput. We progressively increase the bandwidth from 10 Mbps up to 130 Mbps, and repeat the same experiments with four different block sizes. The results are shown in Fig 5, which plots the average consensus latency and average transaction throughput of 10 consecutive rounds of consensus protocol.

The consensus latency decreases with the increase of bandwidth, and finally trend to be flat. With the improvement of bandwidth, transmission rate will be fast in network. As the block transmission time accounts for 90% of consensus latency, the transmission time will decrease when increasing the bandwidth. As result, average transaction throughput will increase, which means that blockchain can process transaction faster if increasing bandwidth. Therefore, we can handle smaller bandwidth while still achieving as low consensus latency as possible.

## 6.5 Sybil Attack

In a sybil attack, an adversary can control some malicious nodes that compete for being block proposer but refuse to generate valid blocks or generate empty blocks. In addition, malicious nodes will not sign on valid block hash and not aggregate partial signatures to generate full signature.

We analyze how Sybil percentage of Sybil nodes in consensus protocol impacts consensus latency and average transaction throughput. We increase the percentage of Sybil nodes from 5% up to 49%, and repeat the same experiments with four different network sizes.



The results shown in figures show that the percentage of Sybil nodes does not impact consensus latency significantly. Since Sybil nodes are not absent from block proposer process, empty block will be generated when Sybil nodes become block proposer. In this case, consensus latency will not significantly increase because the transmission time of empty blocks is relatively small. Because the empty block does not contain any transactions and the consensus delay will be relatively low, so the average transaction throughput will not change greatly. Thus, average transaction throughput not fluctuate with the increase of the percentage of Sybil nodes. As a result, our protocol can resist to Sybil attack

## 6.6 Jamming Attack

We present the performance of our protocol when bursty jammers with constraint of . This indicates that jammers can jam consecutive slots at any interval length of slots. We analyze the impact of on consensus latency ang throughput. We run our protocol for 10 consecutive rounds with ranges from 0.1 to 0.5, and repeat same experiments with four different network sizes.



As shown in the Fig, the consensus latency will decrease with the increasing of due to lower frequency of jamming attack. When increasing the value of , the required time of full signature aggregation process will decrease. As block size is fixed and consensus latency is reduced, the average throughput significantly increases with the improvement of . The consensus latency increases quickly with small indicates that the introduction of adversary time window estimation can address continues heavy contention of jamming attack.

# Conclusion and Future Research

In this paper, we propose a stable wireless blockchain consensus protocol SWIB, which can ensure the stable consensus process leveraging the security of threshold signature scheme. We propose a new verifiable block proposer protocol, which can prevent the formation of forks and choose a more reliable block proposer to generate a valid block. Besides, by combining the nature of wireless communication protocol with a BLS threshold signature scheme, consensus nodes can verify and vote for the validity of block proposal in secure and efficient way. In addition, by using the characteristic of BLS threshold signature scheme, all consensus nodes can recover full signature to complete block finalization in distributed manner. Thus, block finalization not rely on correct block proposer, which means SWIB decouples block proposer from block consensus process. Besides, SWIB prevents Sybil attacks and jamming attacks. Analysis and simulation result show the efficiency and security properties of the SWIB protocol. Our protocol can also

In the future, we will investigate the multi-hop version of SWIB protocol in wireless ad hoc networks. What's more, it is necessary to investigate the impacts of mobility in ad hoc wireless network. Also, we will explore whether our protocol can mitigate other attacks such as long-term attacks, nothing-at-stake attacks, eclipse attacks, etc.

# References:

[1] J. Xu, S. Wang, A. Zhou and F. Yang, "Edgence: A blockchain-enabled edge-computing platform for intelligent IoT-based dApps," in China Communications, vol. 17, no. 4, pp. 78-87, April 2020, doi: 10.23919/JCC.2020.04.008.

[2] T. Maksymyuk, J. Gazda, L. Han and M. Jo, "Blockchain-Based Intelligent Network Management for 5G and Beyond," 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT), 2019, pp. 36-39, doi: 10.1109/AIACT.2019.8847762.

[3] R. Jabbar, N. Fetais, M. Kharbeche, M. Krichen, K. Barkaoui and M. Shinoy, "Blockchain for the Internet of Vehicles: How to Use Blockchain to Secure Vehicle-to-Everything (V2X) Communication and Payment?," in IEEE Sensors Journal, vol. 21, no. 14, pp. 15807-15823, 15 July15, 2021, doi: 10.1109/JSEN.2021.3062219.

[4] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf, 2008.

[5] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” in *Annual* *International Cryptology Conference*. Springer, 2017, pp. 357–388.

[6] M. Castro, B. Liskov. Practical Byzantine fault tolerance[C]. Proceedings of the 3rd Symposium on Operating Systems Design and Implementation(OSDI), 1999: 173-186.

[7] Z. Jiao, B. Zhang, L. Zhang, M. Liu, W. Gong and C. Li. A Blockchain-Based Computing Architecture for Mobile Ad Hoc Cloud, in IEEE Network, vol. 34, no. 4, pp. 140-149, July/August 2020.

[8] Z. Jiang, Z. Cao, B. Krishnamachari, S. Zhou and Z. Niu, "SENATE: A Permissionless Byzantine Consensus Protocol in Wireless Networks for Real-Time Internet-of-Things Applications," in IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6576-6588, July 2020.

[9] Q. Xu, Y. Zou, D. Yu, M. Xu, S. Shen, F. Li. Consensus in Wireless Blockchain System, in WASA, 2020.

[10] Y. Zou, M. Xu, J. Yu, F. Zhao and X. Cheng, "A Fast Consensus for Permissioned Wireless Blockchains," in IEEE Internet of Things Journal, 2021.

[11] M. Xu, F. Zhao, Y. Zou, C. Liu, X. Cheng, F. Dressler. BLOWN:A Blockchain Protocol for Single-Hop Wireless Networks under Adversarial SINR, in CoRR abs/2103.08361, 2021.

[12] M. Xu, C. Liu, Y. Zou, F. Zhao, J. Yu and X. Cheng, "wChain: A Fast Fault-Tolerant Blockchain Protocol for Multihop Wireless Networks," in IEEE Transactions on Wireless Communications, vol. 20, no. 10, pp. 6915-6926, Oct. 2021, doi: 10.1109/TWC.2021.3078639.

[13] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, “A survey of distributed consensus protocols for blockchain networks,” IEEE Commun.Surv. Tutorials, vol. 22, no. 2, pp. 1432–1465, 2020.

[14] M. Vukolic, ‘‘The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication,’’ in Proc. Int. Workshop Open Problems Netw. Secur., 2015, pp. 112–125.

[15] Burstcoin official website. https://www.burst-coin.org/. May. 2019.

[16] B. Wiki. Proof of burn. [Online]. Available: https://en.bitcoin.it/wiki/Proof\_of\_burn

[17] V. Buterin and V. Griffith, ‘‘Casper the friendly finality gadget,’’ 2017, arXiv:1710.09437. [Online]. Available: https://arxiv.org/abs/1710.09437

[18] Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network. https://link.springer.com/content/pdf/10.1007%2F978-3-319-91458-9\_41.pdf. Jan. 2019.

[19] Fabric official website. https://get.fabric.io/. Jan. 2019.

[20] Open Network for the Smart Economy. Accessed: Mar. 20, 2018. [Online]. Available:

[21] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, ‘‘Zyzzyva: Speculative byzantine fault tolerance,’’ ACM Trans. Comput. Syst., vol. 27, no. 4, pp. 1–39, 2010.

[22] J. Kwon. Tendermint: Consensus without mining.

https://tendermint.com/static/docs/tendermint.pdf (21 August 2021, date last accessed).

[23] Y.Gilad, R. Hemo, S. Micali, et al. Algorand: Scaling Byzantine agreements for cryptocurrencies[C]. In: Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28–31, 2017: 51–68.

[24] M. Zheng, M. Goldenbaum, S. Stańczak and H. Yu, "Fast average consensus in clustered wireless sensor networks by superposition gossiping," 2012 IEEE Wireless Communications and Networking Conference (WCNC), 2012, pp. 1982-1987, doi: 10.1109/WCNC.2012.6214113.

[25] M. Goldenbaum, H. Boche and S. Stańczak, "Nomographic gossiping for f-consensus", Proc. 10th Int. Symp. Model. Optimiz. Mobile Ad Hoc Wireless Netw., pp. 130-137, 2012.

[26] C. Newport and P. Robinson, “Fault-tolerant consensus with an  
abstract mac layer,” arXiv preprint arXiv:1810.02848, 2018.

[27] F. Molinari, N. Agrawal, S. Stańczak and J. Raisch, "Max-Consensus Over Fading Wireless Channels," in IEEE Transactions on Control of Network Systems, vol. 8, no. 2, pp. 791-802, June 2021.

[28] H. Moniz, N. F. Neves and M. Correia, "Byzantine Fault-Tolerant Consensus in Wireless Ad Hoc Networks," in IEEE Transactions on Mobile Computing, vol. 12, no. 12, pp. 2441-2454, Dec. 2013, doi: 10.1109/TMC.2012.225.

[29] A. Boldyreva. "Threshold signatures, multi signatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," in Proc. 6th Int. Workshop Theory Pract. Public Key Cryptogr., 2003, pp. 31–46.

[30] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing[C]". International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2001:514-532.

[31] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. "Secure distributed key generation for discrete-log based cryptosystems," in Proc.Int. Conf. Theory Appl. Cryptograph. Techn., vol. 1592, Aug. 2010,pp. 295–310.