**A Stable Blockchain Consensus Protocol for Wireless Blockchain Systems**

**Abstract-** Blockchain can solve security and trust challenges in wireless networks. Most previous studies of blockchain consensus protocols in wireless network rely on efficient and stable transmissions and correct leaders. Nevertheless, nodes in wireless blockchain have limited physical resources, unreliable channels, and varying bandwidths influenced jamming attacks or environments. In this paper, we propose a novel Byzantine fault-tolerant consensus protocol SWIB for blockchain in wireless networks, which do not rely on reliable leader-driven communications. SWIB selects a block proposer randomly to prevent adversary corrupt the block proposer, and uses threshold signature scheme as block proposal voting mechanism to reduce the communication overhead and computational overhead of blockchain system. Because only one block will be confirmed in per round, SWIB protocol can avoid the occurrence of conflicting blocks and blockchain forks. Moreover, it can guarantee security while tolerating at most faulty nodes among consensus nodes. Extensive simulation results show that SWIB is resistant to jamming attacks, double-spending attacks, and Sybil attacks.

# Introduction

With the rapid development of wireless communication and blockchain technologies, much work has been focused on applying blockchains to wireless applications, such as mobile edge computing [1], intelligent 5G technology [2], Internet of vehicles [3], etc. 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 significant challenge of security and trust. With salient properties of decentralization and persistence as well as traceability, blockchain provides 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, much study on wireless blockchain systems is to directly enable popular blockchain protocols deployed in the global Internet to wireless network environment. Consensus algorithms adopted by these blockchain protocols (e.g., Proof of Work [4]), complicated design (e.g., Proof of Stake [5]) usually cause massive resources consumption, and rely on reliable communications (e.g., Practical Byzantine Fault Tolerant [6]). However, these consensus algorithms are difficult, if not impossible at all, to be deployed in wireless networks with limited resources and time-varying channels. 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 applications of traditional blockchain consensus algorithms in wireless networks, which has motivated research on design of blockchain protocol for wireless networks.

Recently, some research on wireless blockchain systems leverage the characteristics of wireless networks to design efficient wireless blockchain consensus protocols. In order to adapt to the high dynamics of the mobile ad-hoc network, Jiao et al. [7] designed 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, Jiang et al. [8] proposed 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, Xu et al. [9] proposed an efficient and fair Proof-of-Communication consensus protocol in wireless blockchain system. And Zou et al. [10] proposed a fast consensus protocol for permissioned wireless blockchain system, which can achieve *k*-times consensus in unreliable and multi-access wireless environment. Besides, to overcome the interference of wireless broadcast communication, Xu et al. proposed 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, Xu et al. designed 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 stable wireless blockchain consensus protocol SWIB. This protocol is analogy with Proof-of-Stake consensus algorithm, which means nodes can achieve consensus without consuming massive resources for mining. SWIB adopts verifiable randomized election scheme and secure threshold signature scheme to ensure that all nodes in the wireless blockchain system can reach consensus in a random and steady manner. The SWIB protocol operates in a round by round fashion. In each round, a single block proposer is randomly and non-interactively selected according to nodes' probability. This probability depends on nodes' stability, which is a function of nodes' active time and the number of recent generated blocks by different nodes. In the block proposer election phase, each node only knows whether it becomes block proposer, but does not know who is actually the block proposer. However, each node can verify the legitimacy of the elected block proposer independently. Such design can greatly reduce resource cost of block proposer election and the corruption risk of adversary. In addition, we adopt threshold BLS (Boneh-Lynn-Shacham) signature scheme to improve the efficiency of reaching consensus in blockchain system. Such a signature scheme will greatly reduce the system overhead of consensus process. Another benefit of using such a signature scheme is decoupling of block proposer selection from block verification and finalization. In this way, block finalization can be achieved by any node who has obtained sufficient votes, not only rely on correct block proposer. Such design can improve the stability and efficiency of consensus process, and also reduce the risk of faulty consensus due to faulty nodes or unstable wireless channels.

Our contributions in this paper are summarized as follows:

* We propose a new blockchain consensus protocol SWIB, which combines verifiable random election with threshold BLS signature. It can ensure stable generation of blocks in wireless blockchain systems and reaching consensus in unreliable and unstable wireless networks.
* We propose a verifiable random election protocol, which is more suitable for wireless networks. The protocol can elect a high-quality node as block proposer in a randomized and verifiable way. 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 as vote mechanism to improve the efficiency of consensus process and decouple block proposer from consensus communication. In this way, even block proposer fails after broadcasting a new block, block verification and finalization can also be completed through a round of partial signatures exchanges. Additionally, our protocol satisfies strong consistency that can efficiently prevent blockchain forks.
* We analyze the system overhead of SWIB protocol to validate the efficiency of our protocol. Besides, we also prove that SWIB can guarantee persistence and liveness to wireless blockchain system if adversary controls less than 50% of voting power.
* Finally, extensive simulation results validate the correctness of our theoretical analytical results.

The rest of this paper is organized as follows. Section 2 introduces related work on state-of-the-art blockchain protocols, wireless consensus algorithms, and threshold BLS signature scheme. Models and assumption are presented in Section 3. In Section 4, we present the details of the stable wireless blockchain consensus protocol. Security analysis and performance analysis are conducted in Section 5. Extensive simulation results are presented in Section 6 for the performance evaluation and the conclusion of this paper is given 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 briefly introduce blockchain consensus protocols category in this section, more precise 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 these protocols can tolerate Byzantine fault and provide strong consistency. The very classical communication-based consensus protocol is practical Byzantine fault tolerant consensus protocol (PBFT) used in Hyperledger Fabric [19]. A block proposer is elected from all consensus nodes to propose a new block. And then the block proposer is responsible for driving communication of vote phase. The cost of adversary being malicious in communication-based consensus protocols should be small. Because consensus nodes not require to mortgage assets or consume resource in consensus process. However, the cooperation mechanism of these protocols can eliminate the influence of malicious behavior to ensure blockchain system security.

Communication-based blockchain consensus protocols have low scalability due to high communication complexity of consensus process. Some protocols are proposed to improve consensus performance to overcome the low scalability of these protocols. 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. Consensus nodes will change view when the faulty leader leads to the interruption of consensus process. Besides, communication-based blockchain consensus protocols usually require reliable message transmission model, as well as make use of all-to-all broadcast communications. Therefore, these blockchain consensus protocols are more suitable for scenarios with small network size and reliable communication model.

## 2.2 Consensus Protocols for Wireless Networks

Many consensus algorithms have been proposed to improve consensus efficiency in wireless networks. Leveraging 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 global consensus among nodes in clustered wireless network, 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 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 as the vote of block proposal, and broadcast it to other nodes. Unique full signature will be recovered by any node who collects a threshold of distinct partial signature shares of block proposal. In this way, even some messages loss or some nodes fail, block consensus can be achieved in blockchain system finally.

BLS signature scheme [30] utilizes cyclic group and bilinear mapping to construct aggregate signature, which used in multi-party signature and verification. 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 used to compute public key. Signature of message can be computed as, where can ensure the integrity of messageVerifiers can verify the signature by checking whether is valid.

Threshold BLS signature (TSS) scheme [29] is derived from BLS signature scheme, and work in non-interactive manner. 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. Participants can obtain other public key shares by exchange messages with each other. 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. The partial signature generation protocol will generate partial signature of each node, which is similar to BLS signature scheme. Besides, he full signature recovery protocol will recover unique full signature, i.e. a unique special value of Lagrange interpolation polynomial of partial signatures. The recovery of full signature can be finished without interacting with other participants. Besides, the verification algorithm of threshold BLS signature scheme uses the main public key to verify the validation of the complete signature, in which the partial signature verification is also same with BLS signature scheme.

SWIB protocol leverages -threshold BLS 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 a threshold of partial signature shares. Our protocol decreases the risk of single point failure by decoupling block proposer with consensus vote 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

## 3.1 Network Model

In this study, we consider a wireless network consisting of nodes, which are deployed in 2-dimentional Euclidean space. Let be the set of nodes. We assume any node can communicate with any other node in the network directly, which means the network under study forms a complete graph. In practice, such network can be formed by a group of Unmanned Aerial Vehicles or intelligent vehicles. The transceiver at each node works in a half-duplex manner, which can transmit or receive message, but not both at a time. The time in our protocol execution is divided into rounds, each of which contains constant slots. A slot is the time unit for nodes to transmit or receive a packet. It is assumed that the transmission between nodes in a slot are synchronized. We further assume that any node can join or leave the wireless blockchain network freely. Each node can generate its private-public key pair and main public key by running a secure distributed key generation protocol. Each node can obtain other nodes' public keys and identities by exchanging messages. The number of honest nodes is assumed to satisfy the requirement of threshold BLS signature scheme, because the security of SWIB protocol relies on security of the scheme.

## 3.2 Communication Model

In this work, nodes communicate with each other over wireless channels, which are assumed to follows the standard path loss model and Rayleigh fading model [32]. The path loss between and is given as

where is defined as the path loss of reference distance , and is the Euclidean distance between and , path loss exponent . Let be the channel gain from to , following the complex normal distribution with zero mean and variance (i.e., ). When a signal is transmitted from to with transmission power , the Signal-to-Interference-plus-Noise ratio of message transmission at a receive node :

where is the additive white Gaussian noise power. Let be the number of arbitrary nodes transmitting messages in the same slot with , the inter-local interference power between and can be derived as . Since we consider a directional message dissemination model, the interference can be negligibly small, e.g., for all . Thus, we can approximate the SINR as the Signal-to-Noise ratio

The absolute time is split into time slots of fixed time interval of each slot is calculated by

where is the maximum size of packet sent by nodes during the consensus process in bits, is the target Signal-to-Interference-plus-Noise ratio (SINR) of transmission from to , and denoted the bandwidth utilized for transmission. The communication between nodes is interrupted when SINR is below the given threshold . Since is exponentially distributed, the communication interruption probability between and is expressed as follows,

In SWIB, communication failure occurs in the blockchain network when receiver meets the communication interruption with more than nodes. Let be the set of consensus nodes in a round, and be the set of nodes that fail to transmit a message to the receiver during message transmission. The probability of communication failure due to communication interruption is given as

The security of SWIB not relies on the correct of block proposer. Since if block proposer is faulty, the rest of honest can still vote for an empty block and reach consensus eventually. We will analyze the performance of SWIB utilizing the communication interruption probability and the probability of communication failure.

## 3.3 Blockchain Basics

We assume that each node locally maintains a blockchain, which is a hash-chain of blocks. Each block contains a set of transactions, each of which consists of payer's information and payee's information as well as other necessary contents. Let represent blockchain, block, and transaction, respectively. The data structure of block and transaction is shown in Fig. 1. A block includes block header and block body. The body usually stores transaction meta data. Block header records blockchain version, block proposer, block height, previous hash, block hash, block full signature, and transactions hash, etc. A transaction contains payer's ID, payee's ID, service information, timestamp, payer's signature, payee's signature, etc.



**Fig. 1. Data structure of a block and transactions.**

## 3.4 Adversary

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

* Adversary can issue Sybil attacks, that is, creation of pseudo identities. The malicious nodes with pseudonym will not vote for valid block or generate valid block in consensus process, even transmit faulty messages to interrupt the consensus process;
* Adversary can launch jamming attacks to interfere with the message transmission of honest nodes. Without loss of generality, we assume that the capability of adversary is - bounded, i.e., in any interval of consecutive slots, adversary can jam no more than slots, where and 0< ≤ 1.

# The Stable Wireless Blockchain Consensus Protocol

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

## 4.1 Overview Architecture

This subsection introduces the preliminary design of SWIB protocol. 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 the block proposer for the current round; (2) verify the legality of new block, and run signature generation algorithm to vote for valid block; (3) run signature aggregation algorithm and signature recovery algorithm to finalize the block when generating the full signature.

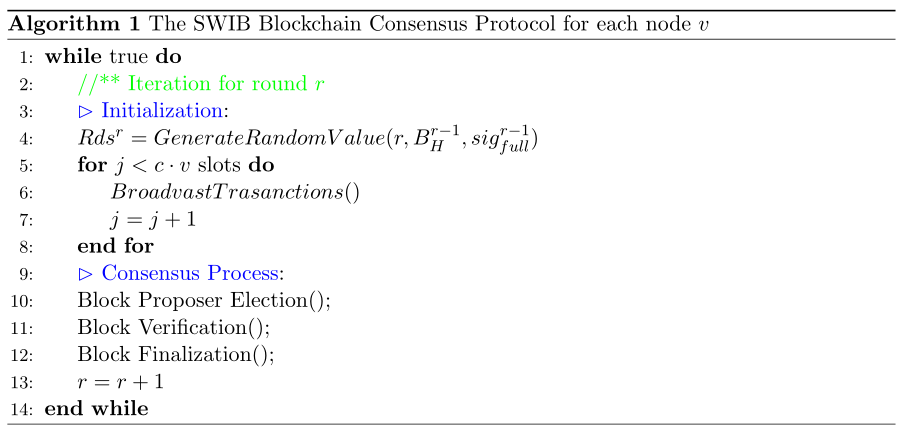
Our consensus protocol works in a round by round fashion. As shown in Fig. 2, we start with a secure randomness generation process. It is in general not secure to allow nodes to predict who will be the block proposer in the next round. Thus, we need a randomness source to ensure that nodes cannot obtain the information of next block proposer previously. In SWIB, consensus nodes can join the blockchain system by submitting Sybil-resistant-proof. Each node registers in the blockchain system by depositing certain amount of money, which will be stored in a virtual account. Only the depositor executing unpledged operation can take out the money. The protocol adopts a distributed randomness generation scheme, which ensures that all nodes can generate same randomness per round independently. According to the round randomness, a block proposer will be elected at the beginning of each round through block proposer election protocol. Then, the elected block proposer will generate a block and broadcast it to 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 the block through a signature recovery protocol.

We aim to propose a protocol such that hundreds of nodes can achieve consensus in a wireless network with unreliable and unstable channels. Fig. 3 shows the main components of our protocol.

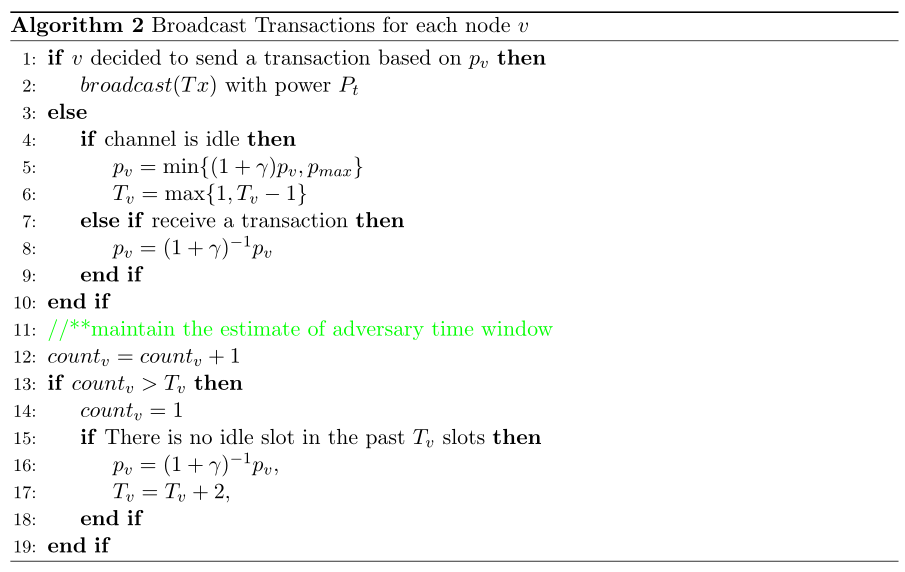


**Fig. 3. Main design components of SWIB.**

* Block proposer election: At the beginning of each round, a random number is independently generated by nodes via a distributed randomness generation scheme. Each of the nodes checks whether it becomes the block proposer for the current round through verifiable block proposer election protocol, which uses node's private key and the round random number as inputs. Besides, each node can verify the legality of block proposer according to its public key, proof, and the round random number. The elected block proposer will pack transactions from its transaction pool to generate a new block, and disseminate it to other consensus nodes in the wireless network.
* Block validation: Upon receipt of the proposed block, nodes will verify validity of the block. Each node will generate a partial signature of block hash if the result of the verification is true. Then, they will broadcast their partial signature shares to other nodes.
* Block finalization: Full signature will be recovered if a node receives and then aggregates enough partial signature shares. Then, the full signature will be broadcast to nodes. This full signature can be seen as the proof of block finalization. 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. Upon receipt or generation of a full signature, nodes will append the corresponding block into their local blockchain. After that, nodes will compute a new random value for the next round through distributed randomness generation protocol.



Algorithm 1 shows the process of a complete round in SWIB protocol. After generating a random value of new round, consensus nodes perform transaction collection. Note that the pseudo code of transaction broadcast operation presented in Algorithm 2, whose parameters are utilized to ensure jamming-resistant communications. The details of every stage will be given 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.

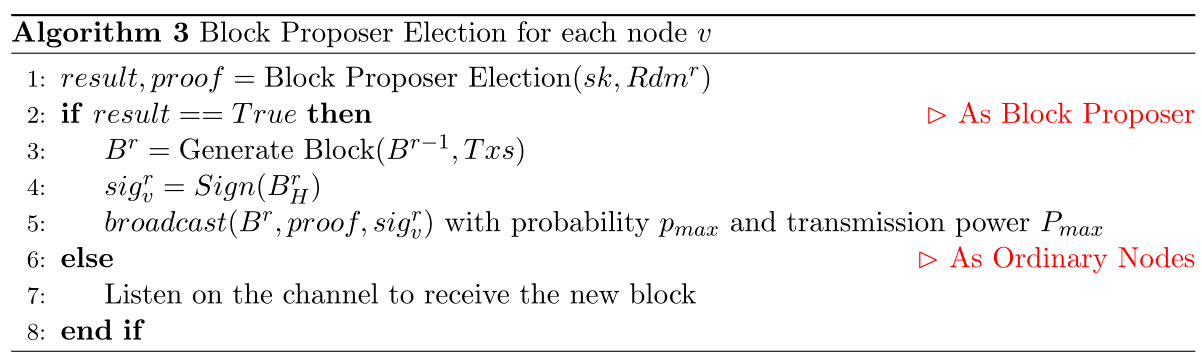


## 4.2 The SWIB Protocol

Before participating consensus, each node needs to obtain some information of the blockchain system. Each node will generate key pairs and the main public key through a distributed key generation algorithm. Besides, nodes also should request the identities and public keys of other consensus nodes as well as necessary blockchain history information. The first part of consensus process is block proposer election, which discussed in the following subsection in details.

### 4.2.1 Block Proposer Election and Block Generation

We present a verifiable block proposer election protocol, which is based on distributed randomness generation protocol and verifiable random function as well as threshold BLS signature scheme. Algorithm 3 gives the procedure for block proposer election at a node.



The block proposer election protocol adopts a random 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. The inputs for each selection include round number , block hash and the full signature of the previous round . Using the full signature as randomness source can ensure the uniqueness and immediacy of the inputs. As shown in Fig. 4, the randomness is performed as the normalized hash value of the above input combination:

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



**Fig. 4. Randomness generation of per round.**

The output of the distributed randomness generation protocol is unpredictable and unique. The recovery process of full signature for each round is unpreventable, provided that majority of the consensus nodes are correct as we assume. 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 known in advance, the full signature can only be revealed at the end of round. Thus, no one can predict the output of distributed randomness generation protocol in advance. The full signature cannot be tampered due to the security of threshold signature scheme, even a node first recovers full signature before others. Another benefit of distributed randomness generation protocol is that nodes can enter the following round non-interactively. Upon receiving or generating a valid full signature, consensus nodes will append the corresponding block into their local blockchain. Then, each node will start a new consensus process by computing the following round randomness.



**Fig. 5. The block proposer election for round .**

As shown in Fig. 5, all consensus nodes are assumed to have the same view on the list of nodes constituent of the wireless blockchain system. Each node executes a distributed key generation protocol to obtain a private-public key pair and main public key . All nodes will exchange the public keys before entering the consensus process. Thus, the public key list can be seen as the identities of nodes. To ensure a same view on the node list, the list is assumed to be sorted according to the hash values of public keys.

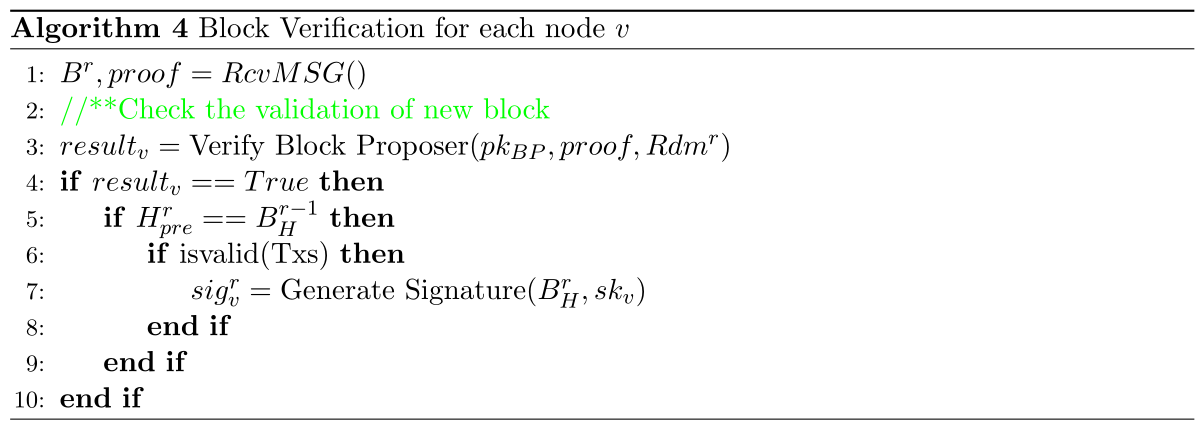
The block proposer for each round is elected according to nodes' elected probabilities and the value of the round. All honest consensus nodes can obtain the same value by executing the distributed randomness protocol with same inputs. In SWIB, the elected probability of a node is mainly determined by its stability, which is a relative concept. Let be the remaining active time of in the system, then the sum of all consensus nodes' remaining active time will be . The active time ratio of (denoted by ) is calculated as . The consensus ratio of (denoted by ) is calculated as , where is the number of blocks generated by in the latest blocks on the blockchain. When the blockchain length is smaller than , the consensus ratio of every node is set as zero. For the stability of (denoted by ), we have , where and are the weights of active time ratio and consensus ratio, respectively, and . According to nodes' stability values, we have the elected probability of each : . Our election protocol is designed to elect one block proposer for each round based on the elected probabilities of nodes and the value of the round.

The verifiable block proposer election protocol consists of a block proposer election algorithm and a block proposer verification algorithm. The block proposer election algorithm ensures that the election process is randomized and verifiable. To determine the block proposer for the current round, the election algorithm divides interval into consecutive intervals:. If , then the node whose public key is rank the will be the block proposer of round . Consensus nodes will independently check whether they are the block proposer of a round. After executing the election algorithm, the block proposer will obtain the election result and a proof while others will only get the election result and a common string. Therefore, each node can only know whether it is elected as block proposer by its private key and the value. Meanwhile, block proposer verification algorithm guarantees that other nodes can verify the legality of elected block proposer by its public key, value and the proof. Besides, the verifiable block proposer election protocol is built upon secure and robust threshold BLS signature scheme, which can tolerate any malicious nodes among the nodes. The secure threshold of the 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, which is elected as the block proposer, will pack transactions to generate a new block. The header of the block can be represented by a tuple , where is the round number, is the hash value of previous block, is the block hash, and is the identity of the current block proposer. Block body usually stores transaction metadata, which is a transaction list. The block proposer will broadcast the block and signature of the block hash to other nodes. In addition, the full signature can only be appended to this block when a node gathers enough partial signature shares and reconstructs the signature.

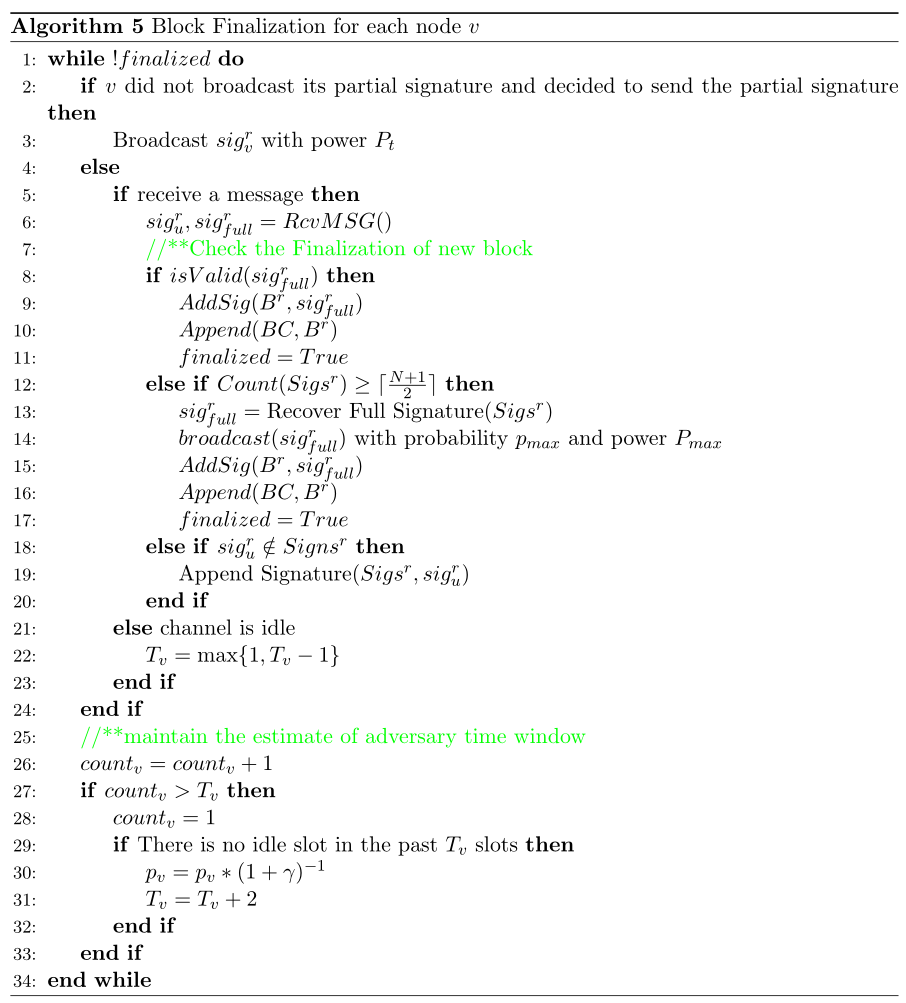
### 4.2.2 Block Verification and Finalization

The pseudo code of block verification and finalization is presented in Algorithm 4 and Algorithm 5. The block verification and finalization depend on secure threshold BLS signature scheme. 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. Blocks and the signatures are sent to consensus nodes through wireless network broadcasting.



In the block verification phase, a node needs to check the validation of the proposed block through the following components:

* Block proposer: The result of block proposer verification algorithm, which 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 shown in Fig. 6, if all the above conditions are satisfied, a node will then generate partial signature of the block hash and broadcast it to other nodes. In the block finalization phase, each node will gather partial signature shares of the block hash to recover the full signature. In SWIB, we can use the full signature as the proof of block finalization. The reconstruction of valid full signature proves that a threshold of nodes have signed block hash, which means that a sufficient number of nodes vote for the block validity. Therefore, it is feasible to use the full signature as the signal of block finalization. The stability of consensus process will be improved since full signature recovery can be done independently by any correct node. In this way, any node which 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. Moreover, since correct nodes can only vote once in a round, only one block will be confirmed in a complete round. In this case, SWIB satisfies strong consistency, which means that it can prevent the occurrence of blockchain fork.



**Fig. 6. Block verification and finalization at a node .**

### 4.2.3 Incentive and Punishment Mechanism

We design an incentive mechanism to encourage consensus nodes to generate signature. Both block verification and signature generation consume certain computational power of the corresponding nodes. Rational nodes are more willing to wait for receiving full signature, rather than consuming their computational power to verify a block, generate partial signature, and recover full signature. Therefore, an incentive mechanism is needed to motivate nodes to participate the consensus process actively. The total rewards in blockchain system are the submitted transaction fees of valid blocks. In our incentive mechanism, part of the transaction fees is rewarded to block proposer, and the rest of transaction fees will be averagely distributed among the nodes whose partial signature shares are used to recover the full signature.

Transaction fees will be shared by those nodes who have the smallest average timestamps of partial signature shares, which are used to recover the full signature. 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 , respectively. The average timestamps of the two sets are and . If , then consensus nodes whose partial signatures are in set will share the reward. This incentive mechanism not only encourages nodes to verify block and generate partial signature, but also motivates nodes to broadcast partial signature and full signature as soon as possible. The earlier the signature broadcasting, the higher the chance being rewarded. 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, the node usually generates invalid block or not generate any block before timeout; 2) node usually sends invalid signature or garbage messages in block verification and finalization phase. Nodes will be punished by reducing their lifetime when nodes are judged malicious. Reducing lifetime will decrease the stability of nodes, which will reduce the probability that obtains block generation reward. As a 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. Moreover, the transaction throughput of blockchain system will not significantly decrease since valid blocks always are generated by honest block proposers.

### 4.2.4 Synchronization Mechanism

In our protocol, we consider a more efficient and secure self-initiated mechanism. When a new node joins the 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 can request different blocks from different neighbors with high stability and small distance. Let be the communication interruption probability between and , then can succeed in receiving a block from with probability . The probability that a node receives consecutive blocks history successfully is . This mechanism reduces the overhead of node in blockchain history transmission. The consumed power of node transmitting a small number of blocks will be lower than that large number of blocks. Performance of system will decrease with the high frequency of communication failure when transmitter has low transmission power. Thus, the mechanism can effectively prevent a single node from communication failure due to consuming large power on the transmission of blockchain history. Meanwhile, synchronization mechanism will reduce the risk that node transmits error blockchain history to new node. Once a node transmits wrong blockchain history information, the receive node can immediately detect through blocks received from other nodes. In this case, node can request blocks from other nodes again to ensure the reality of blockchain history information.

Synchronization procedure can also happen when consensus node receives partial signature or full signature of block, which has not been received by the node. At the end of each round, honest node will request the latest consecutive blocks from multiple neighbors, which have high stability and near distance. The received blocks are added to the local blockchain of the node if it is not present in the local blockchain. After that, the node participates consensus process via generating the following round random number according to the full signature of latest block. This procedure ensures that blockchain system will not be stopped because the number of honest nodes succeed in verifying legality of block proposer not meets the security threshold. Thus, synchronization procedure is utilized to endure the security of SWIB.

# Protocol Analysis

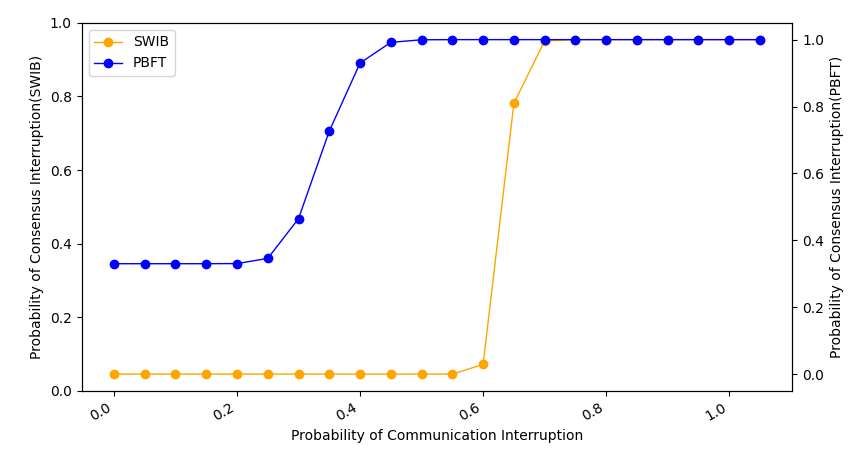
## 5.1 Performance Analysis

### 5.1.1 Consensus Interruption Analysis

In SWIB, block proposer is responsible for block generation. After a new block is generated, the block proposer sends the new block to other consensus nodes for voting to reach a consensus on the new block. Consensus nodes then broadcast partial signature. When a consensus node receives a sufficient number of partial signature shares, block finalization has been achieved via recovering a full signature. The full signature is seemed as a synchronous block update message, which is reported to other nodes by the consensus node. Our protocol provides a synchronization mechanism to solve the problem of blockchain inconsistency between different nodes.

Failed consensus process only occurs when no consensus node receives a sufficient number of partial signature shares in block finalization. This means that all consensus nodes fail to receive a sufficient number of responses from others. Therefore, the average probability of consensus interruption is calculated as

where is the average SINR communication interruption probability at .



### 5.1.2 Consensus Latency Analysis

In SWIB, a block is composed of block header and a block body. the block header mainly stores metadata for identifying blocks and hash root (e.g., Merkel root) calculated based on the transactions stored in block body. Let be the number of CPU cycles available for block proposer in a unit time, and be the number of CPU cycles required to calculate a hash value for a transaction. For a block containing transactions, the required number of hash operations to generate the hash root is . The number of CPU cycles required for a hash operation to generate the root is . Thus, the block packaging latency is expressed as

The data transmission of block consensus includes: 1) block proposer broadcasts the new block to all consensus nodes, and 2) consensus nodes broadcast partial signature shares to others, as well as 3) consensus node broadcast full signature to others. Let be the maximum time spent on generating a partial signature, and be the maximum time of generating a full signature. The block consensus latency at can be expressed as follows,

where represents the data size of block header, represents the data size of a transaction, and represent the data size of partial signature and full signature, respectively. Assuming that the partial signature broadcast by is the partial signature received by , the generation latency for a valid block is given as

When generating an empty block in a round, the data transmission of block consensus includes: 1) a node broadcasts the empty block to all consensus nodes, and 2) consensus nodes broadcast partial signature shares to others, as well as 3) consensus node broadcast full signature to others. The block consensus latency can be expressed as follows,

Let be the probability that elected block proposer is honest. According to the consensus interruption probability and consensus latency, we can compute the average block generation latency in our protocol as

.

where is consensus interruption probability, is the generation latency of valid block, and is the maximum valid block generation latency, is empty block consensus latency.

## 5.2 Security Analysis

We conduct a security analysis to show that SWIB 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.

The 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.

### 5.2.1 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 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.

### 5.2.2 Liveness Analysis

For liveness, we need correct nodes that have same view to prevent SWIB protocol from stalling. In other words, our protocol satisfies liveness of blockchain system with faulty nodes.

**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.

### 5.2.3 Random Generation Analysis

Our distributed randomness generation protocol is based on a robust threshold signature scheme. Small part of malicious nodes cannot hinder the generation of a valid full signature if more than consensus nodes are honest. With the robust randomness, adversary cannot corrupt the process of block proposer election. Therefore, verifiable block proposer election protocol that adopts randomness 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. In this case, honest nodes have higher chance to be block proposer. They will generate valid blocks, which can ensure the efficiency of transaction procession.

### 5.2.4 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.

### 5.2.5 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 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.

Consensus interruption occurs when no node receiving a sufficient partial signature in block finalization phase. In SWIB, utilizing an estimate of adversary time window ensures the finalization of block in wireless blockchain network under jamming attack. We assume that the capability of adversary is - bounded, i.e., in any interval of consecutive slots, adversary can jam no more than slots, where and 0< ≤ 1. Let be the communication interruption probability between node and node in a time slot. We assume that a message is sent repeatedly until successful delivery. Let be a random variable, which denotes the number of time slots required to deliver a message from to . The probability that a node receives partial signature should be

The probability of consensus interruption should be

Since

The required number of time slots to receive partial signature shares from nodes with probability is

**Theorem 3**. If adversary is -bounded at any time interval of length slots, and the number of faulty nodes is Block finalization of our protocol can succeed in slots w.h.p., where is the total number of consensus nodes in our protocol.

**Proof**. We assume that an adversary can jam slots of slots, and is the probability of propagate message between nodes within a slot. According to Lemma 1 and Lemma 2, a node can successfully transmit a partial signature to another node after slots with high probability. Therefore, in normal consensus process, nodes can gather partial signatures with high probability after slots. In this case, the full signature can be reconstructed to show the termination of consensus process in a round. Since adversary launching jamming attack, honest nodes can only communicate slots per slots. Thus, the transmission probability of honest nodes should be under jamming attacks. The probability of node can receive a partial signature from another node after slots is . As result, block finalization can be completed after slots w.h.p. When nodes failures happen, error message will be transmitted in block finalization process. The number of faulty nodes is bounded by . There are at most nodes sending error partial signature shares during the signature aggregation process. The consensus process will be finalized after slots, since aggregating at least partial signature shares to recover the full signature. Thus, the upper bound of consensus process should be with high probability.

# Simulation Results

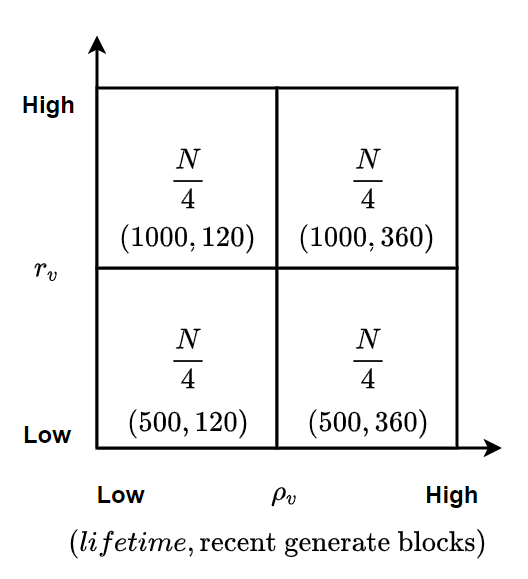
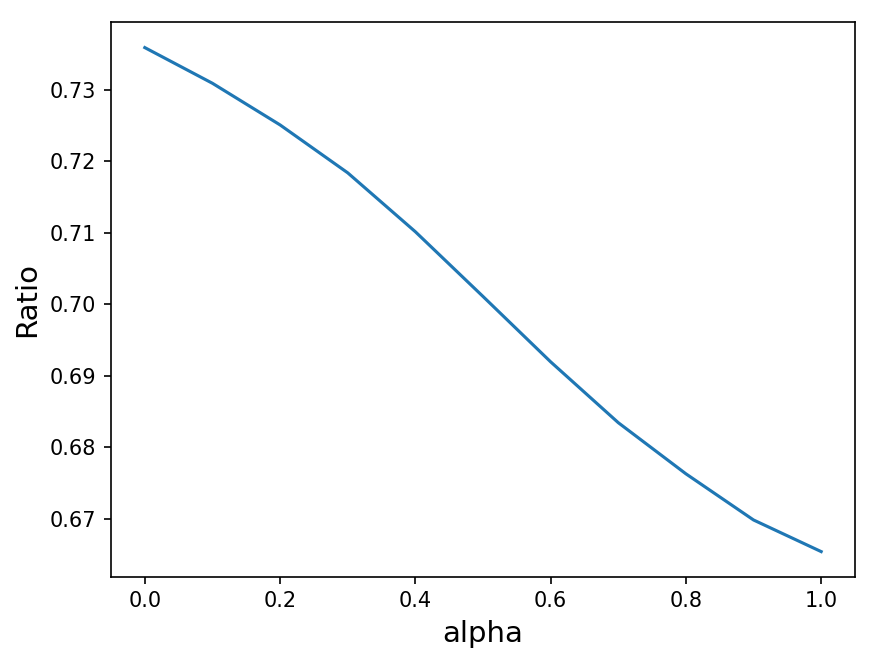
In this section, we conduct simulation experiments to evaluate the performance of SWIB protocol. The impacts of various parameters are investigated, including block size, network size and bandwidth. We evaluate the performance of our protocol through consensus latency and transaction throughput. The consensus latency is the running time of a round. The transaction throughput is the average number of processed transactions per second. All experiments are performed under Windows 10 operating system, and 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.

In SWIB protocol, verifiable block proposer election protocol randomly determines a block proposer through the stability of nodes. According to the defined 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 SWIB protocol. Besides, wireless nodes with lower stability have lower cost to be malicious. This indicates that nodes have higher probability to generate a valid block and maintain the latest blockchain 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 not 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 select high-quality nodes, we analyze the probability of selecting quality nodes as block proposer in 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 , and set that nodes with lower lifetime and number of latest generate blocks; nodes with lower lifetime and higher number of latest generate blocks; nodes with higher lifetime and lower number of latest generate blocks; nodes with higher lifetime and number of latest generate blocks. To investigate the impact of parameter , we measure the percentage of electing high-quality nodes as block proposer within 100 consensus times.

**Fi. Percentage of high-quality nodes elected as block proposer under different coefficients**

As shown in Fig 7, the percentage that electing high-quality node as block proposer in 100 consensus rounds linearly decreases with the increase value of . The experimental result shows that the number of recent generated blocks greatly affects the probability of selecting high-quality nodes, while the lifetime has relatively less influence. The maximum value of the percentage is less than 0.75 while the minimum value is larger than 0.65. When , the percentage of electing high-quality nodes as block proposer can approximately reach 0.7. We set the weight coefficient of stability to a = b = 0.5 in the subsequent performance experiments. These parameters can also ensure high probability that high-quality and stable node is selected as block proposer in block proposer election phase.

## 6.2 Block Size

We analyze how block size used in consensus protocol impacts the consensus latency, which is the time of taking to complete a round of SWIB protocol. And transaction throughput of blockchain system can be measured through used block size and resulting consensus latency. We run the consensus protocol for 10 consecutive rounds increase block size from 0.5 MB to 5 MB, and repeat the same experiments with four different fixed-size networks.



**Fig 8. The performance of SWIB vs. the block size**

As shown in Fig 8, when network bandwidth is limited to 35 Mbps, consensus latency will linearly increase when increasing the size of block. When block size is 4 MB, 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 1 MB, the protocol can achieve big average transaction throughput with small average consensus delay. In summary, 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 network sizes 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.



**Fig 9. The performance of SWIB vs. network size (number of nodes)**

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 1 MB. 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 partial signature shares, which will grow proportionally with the increase of network size. This indicates that all consensus nodes require to verify more partial signature shares, and the recovery of full signature requires larger threshold. Since the block size is fixed and consensus latency increases, the average transaction throughput will decrease. The result shows that average transaction throughput can upper 4000 TPS 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 and network size of 500 nodes. The results are shown in Fig 10, which plots the average consensus latency and average transaction throughput of 10 consecutive rounds of consensus protocol.

**Fig 10. The performance of SWIB vs. bandwidth**

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 linearly, which means that blockchain system 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 vote for valid block hash and aggregate partial signatures to generate full signature.

We analyze how Sybil percentage of Sybil nodes in consensus protocol impacts on 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 and block size of 1 MB.



**Fig 11. The performance of SWIB when confronting Sybil attacks**

The results shown in Fig 11 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 small, so that the average transaction throughput will not decrease greatly. Thus, average transaction throughput not significantly fluctuate with the increase of the percentage of Sybil nodes. As a result, our protocol can resist to Sybil attacks.

## 6.6 Jamming Attack

We present the performance of SWIB protocol when jammers with constraint of . This indicates that bursty jammers can jam consecutive slots at any interval length of slots. We analyze the impact of on consensus latency and 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.



**Fig 12. The performance of SWIB when confronting Jamming attacks**

As shown in the Fig 12, the consensus latency will decrease with the increasing of due to lower frequency of jamming attacks. The required time of partial signature shares aggregation will decline when increasing the value of . With low frequency of jamming attack, honest nodes can broadcast their partial signatures quickly. As result, the consensus latency will be decreased greatly. As block size is fixed and consensus latency is reduced, the average throughput significantly increases with the raise of value . Even adversary issues jamming attacks frequently, the throughput can also exceed 1000 TPS in network size of 700 nodes. These results indicate that the introduction of adversary time window estimation can address continues heavy contention of jamming attacks. In this case, our protocol can securely operate in blockchain system when adversary issuing jamming attack.

# Conclusion and Future Research

In this paper, we propose a stable wireless blockchain consensus protocol SWIB, which can ensure stable and secure consensus process by adopting verifiable random function and threshold signature scheme. We use 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 threshold BLS signature scheme, consensus nodes can verify and vote for block proposal in secure and efficient way. In addition, according to the characteristic of threshold BLS signature scheme, all honest 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 protocol decouples block proposer from block verification and finalization. Besides, our protocol can still work in wireless blockchain system under Sybil attacks and jamming attacks. Analysis and results show the efficiency and security properties of the SWIB protocol.

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 and F. Dressler, "BLOWN: A Blockchain Protocol for Single-Hop Wireless Networks under Adversarial SINR," in *IEEE Transactions on Mobile Computing*, doi: 10.1109/TMC.2022.3162117.

[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] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, “Proofs of space,” in Annual Cryptology Conference. Springer, 2015, pp. 585–605.

[16] K. Karantias, A. Kiayias, and D. Zindros. "Proof-of-Burn," in Financial Cryptography and Data Security, 2020, pp. 523-540, doi: 10.1007/978-3-030-51280-4\_28.

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

[18] F. Gai, B. Wang, W. Deng and W. Peng. "Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network." In Database Systems for Advanced Applications, 2018, pp. 666-681, doi: 10.1007/978-3-319-91458-9\_41.

[19] Hyperledger Fabric. [Online]. Available: https://cn.hyperledger. org/projects/fabric. 2019.

[20] Q. Wang, R. Li, S. Chen, and Y. Xiang. "Formal Security Analysis on dBFT Protocol of NEO."  [arXiv:2105.07459](https://arxiv.org/abs/2105.07459) [cs.CR], 2021

[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] E. Buchman. "Tendermint: Byzantine fault tolerance in the age of blockchains." Ph.D. thesis, The University of Guelph, Guelph, Ontario, Canada, June 2016.

[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.

[32] A. Goldsmith, Wireless Communications. Cambridge University Press, 1 ed., Aug. 2005.