**A Stable Committee-based 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 an innovative Byzantine fault-tolerant consensus protocol SWIB (**S**table **Wi**reless **B**lockchain) for blockchain in wireless networks, which do not rely on reliable communications. SWIB selects a block proposer randomly to prevent adversary corrupting the block proposer, and uses a threshold signature scheme as block proposal voting mechanism to improve the performance 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 attack, double-spending attack, and Sybil attack.

# Introduction

In recent years, both academia and industry have been attracted by intelligent 5G technology and Internet of Things (IoT). The development of 5G technology boosts the evolutions for wireless communication technology, which supports the development of Internet of Things applications. These Internet of Things applications include smart homes, public safety, smart city, autonomous driving, and telemedicine and so on [2, 3, 4]. In order to provide diverse services to improve social and economic developments, massive IoT devices are required to deployed in large-scale wireless networks. It is expected that the number of IoT connected devices will reach 1 trillion in 2035[1]. However, with the tremendous increase of the number of IoT devices, the arisen of data privacy and security restrict the IoT applications. Traditional IoT system can be out of services when the center server crashes or being attacked. Besides, the data in IoT devices will be uploaded to the traditional database, where the administrator can access it without the permission of data owner. With salient properties of decentralized and persistence as well as traceability, blockchain has been regarded as one of the most promising technologies to address security and privacy problems in IoT applications[5，6].

The consensus protocol is used to ensure data consistency and security of blockchain system. And IoT usually requires a consensus protocol to ensure that diverse IoT devices reach consensus on communication data. Therefore, the consensus protocols adopting in blockchain can be applied to the IoT. Although good prospects can be expected, the combination of blockchain and IoT faces some new challenges. Wireless communication is a common obstacle for combining blockchain and IoT due to the emergency of a large number of wireless devices in IoT. The physical limitation of wireless communications, such as channel fading, noise, interference and limited transmission power, will affect the communication between wireless nodes (i.e., wireless IoT devices), then affecting the efficiency of reaching consensus among nodes. Specifically, such impacts will be amplified while the size of network and the number as well as the density of nodes therein are large. Therefore, designing a blockchain consensus protocol for large-scale wireless network is necessary.

Currently, most blockchain consensus protocol are designed under the assumption of secure and reliable transmission. Majority blockchain consensus protocols usually require massive resources consumption (e.g., Proof-of-Work [7]) and complicated design (e.g., Proof-of-Stake [8]), or rely on reliable communications (e.g., Practical Byzantine Fault Tolerant [9]). However, most IoT devices have limited computing capability, which means that it is hard for IoT devices to conduct expensive computing. Meanwhile, PoW-based blockchain consensus protocols suffer from very poor performance with a consensus latency of tens of minutes, thus inefficient for delay-sensitive IoT applications. In contrast, PoS-based blockchain consensus protocols achieve high performance consensus without consuming massive computation power. However, the IoT devices faces nothing at stake problem, which may lead to IoT devices never achieve consensus. Therefore, these protocols require complicated design to ensure the security of consensus process. In addition, BFT-based blockchain consensus protocols can achieve consensus among nodes with low computational power and complexity. But, BFT-based protocols rely on all-to-all internode communications that result in exponentially increasing messages. And IoT networks are expected to involve numerous devices, which means that BFT-based protocols are not suitable for IoT due to low scalable. Meanwhile, BFT-based blockchain consensus protocol cannot satisfy the requirement of IoT applications because that the failure of leader will exponentially drop the performance of consensus.

Considering the challenges of time complexity and energy consumption, some scholars proposed blockchain consensus protocols for wireless network. To address the energy consumption problem in PoW-based consensus protocols, Xu et al. [10] proposed an efficient and fair Proof-of-Communication consensus protocol for single-hop wireless blockchain networks. And Jiang et al. [11] proposed a Sybil-proof-based Byzantine fault-tolerant consensus protocol, which can achieve real-time consensus in wireless networks. Besides, 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 and VRF sortition algorithm [12]. However, the multi-hop scenario was not taken into account in these studies. Xu et al. designed a fast fault-tolerant for multi-hop wireless blockchain network wChain [13]. To accelerate data aggregation, this protocol constructs communication spanner by the maximum independent set. But this protocol cannot tolerate Byzantine failure, which usually occurs in IoT networks. Therefore, it is significant to design a blockchain consensus protocol for IoT networks, which has tolerance on Byzantine failure and high performance on time complexity and energy-consuming issues.

In this paper, we proposed an energy-saving and efficient blockchain consensus protocol for multi-hop wireless networks. The main contributions of our work are summarized as follows:

* We present a consensus protocol for multi-hop wireless blockchain networks. Our consensus protocol improves consensus performance by using a threshold BLS (Boneh-Lynn-Shacham) signature scheme combined with a gossip broadcast communication method.
* We use the available energy and active time as the threshold to construct the consensus committee to guarantee the stability and security of consensus process. We propose a VRF-based leader election algorithm to reduce the corruption risk of adversary and improve the chance of generating valid block.
* We analyze the consensus success probability and expected consensus latency of our consensus protocol, and discuss the consensus security and attack resistance when 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. Models and assumptions are presented in Section 3. In Section 4, we present the details of the stable committee wireless blockchain consensus protocol. Security and performance analyses 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

To describe our propose protocol, we first introduce the state-of-the-art blockchain consensus protocols in this section. Then, the existing consensus protocols for wireless networks are given.

## 2.1 Blockchain Consensus Protocols

We briefly introduce blockchain consensus protocols in this subsection. Two kinds of consensus protocols have been widely utilized in blockchain system: Proof-of-Concept (PoX) and Byzantine fault tolerance (BFT) based.

In Proof-of-Concept consensus protocols, consensus nodes require to compete for block proposal right in each round by providing the proof of certain measurable resources. In Proof-of-Work (PoW) [7] consensus protocol, nodes have to find a hash value satisfying a difficult target to prove they complete large computational work and obtain block proposal right. Since the calculation of the target hash value is a time-consuming process, this protocol cannot provide instant consensus finality of blockchain protocol [14]. Actually, even though adversary controls computing power is less than 50% of total power, multi-blocks confirmation can only guarantee probabilistic consistency of PoW blockchain consensus protocol. Due to the large time-varying of generate a block, the block-confirmation latency of PoW blockchain protocol is large, and transaction throughput is limitation. In order to improve the performance of consensus protocol, some other Proof-of-X consensus protocols are proposed. Instead of computing power, Proof-of-Space consensus protocol [15] relies on the hard disk capacity of consensus nodes. In Proof-of-Space, nodes have to store huge data to get an opportunity of block proposal right. The block creation time interval of this protocol is about 4 minutes. In Proof-of-Burn consensus protocol [16], consensus nodes obtain block proposal chance by burning another “coin”, such as Bitcoin. In Proof-of-Elapsed-Time consensus protocol [17], block proposer is elected based on a random wait time. In order to reduce the high resource consumption of PoW, Proof-of-Stake (PoS) consensus protocol [8] is proposed to as an alternative. In PoS, nodes are elected as block proposer according to their deposited stakes. The more stakes of nodes, the higher probability to be block proposer. The first version of Casper [18] is a hybrid consensus of PoW and PoS, aiming to replace the PoW consensus algorithm with PoS consensus algorithm in Ethereum. In Proof-of-Reputation consensus protocol [19], consensus nodes with enough reputation can obtain the right to generate a new block. Proof-of-Activity consensus protocol [20] is a hybrid protocol that combines aspects of Proof-of-Work and Proof-of-Stake consensus protocols. Consensus nodes use Proof-of-Work to compete to generate a new block, and use Proof-of-Stake to validate/sign the new block.

In BFT-based blockchain consensus protocols, all consensus nodes reach consensus on block proposal by exchanging messages and performing local computation. These protocols can tolerate Byzantine fault and provide strong consistency. The first BFT-based consensus protocol is the practical Byzantine fault tolerant consensus protocol (PBFT) [9]. In this protocol, all nodes participate in the voting process in order to add the new block, and consensus is reached when more than two-thirds of all nodes agree upon the block. The PBFT consensus protocol achieves consensus among consensus nodes with quadratic message complexity. Thus, some BFT-based blockchain consensus protocols are proposed to improve scalability by reducing communication overhead. In NEO [21], partial nodes of delegated Byzantine fault tolerant consensus protocol are delegated to participant consensus process to reach agreement by voting on generated block. Reducing the number of consensus nodes can decrease the number of communication messages, and further increase the performance of system. In Zyzzyva [22], the modified BFT consensus algorithm SBFT employs threshold signatures to reduce communication overhead of consensus process. However, it still requires multi-rounds communications to aggregate signatures and terminate block confirmation. Besides, Byzcoin consensus protocol [23] leverages PoW for consensus group membership management and BFT protocol for block finalization. Reducing the number of consensus nodes can decrease the consensus message complexity, and improve the performance of system. Tendermint consensus protocol of Cosmos [24] adopts leader rotation mechanism that based on voting power of validators, and uses BFT consensus process to finalized one block. This protocol uses gossip protocol to improve the scalability of transaction propagation. Algorand consensus protocol [25] combines Byzantine agreement protocol with VRF committee election scheme to improve the scalability of blockchain consensus process. Nodes are chosen to become committee members to participant in BA protocol and achieve consensus on a set of transactions. However, most BFT-based blockchain consensus protocols require massive message passing to achieve consensus. Therefore, these blockchain protocols are suitable for scenarios with small network size.

## 2.2 Consensus Protocols for Wireless Networks

There has been extensive prior research on consensus protocol in distributed system with reliable or eventually reliable point-to-point channels. Actually, Assumption that majority components are connected is unavailable in the wireless networks. Santoro and Widmayer [25, 26] study consensus in the presence of unreliable communications, and show that consensus is impossible if more than of the possible messages sent in a round can be lost.

Some consensus protocols have been proposed to achieve consensus in wireless networks. Leveraging the natural superposition property of wireless multiple-access channels, Zheng et al. [27] propose an average consensus in clustered wireless sensor networks to achieve consensus. In order to efficiently achieve global consensus among nodes in clustered wireless network, Goldenbaum et al. [28] present an iterative gossip algorithm that is based on the superposition property of wireless channel. Newport and Robinson [29] propose fault-tolerant distributed consensus protocol to achieve the consensus of wireless systems through abstract MAC layer model. Asobes et al [30] present a consensus protocol for wireless networks with anonymous but reliable nodes. This protocol can tolerate unreliable communication and nodes crashes. Besides, Moniz et al. [31] 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 scholars proposed blockchain consensus protocols for wireless networks. Jiang et al. [11] propose a Sybil-proof-based Byzantine fault-tolerant consensus protocol, which can realize real-time consensus in wireless networks. To prevent Sybil attack and improve the efficiency of consensus, this protocol leverages the wireless transmitted signals to select nodes that can participate the final consensus reaching process. Xu et al. [10] propose an efficient and fair Proof-of-Communication consensus protocol in wireless blockchain system. This consensus protocol makes use of transmitted signal power to achieve consensus with the time complexity of . And Zou et al. [32] propose a fast consensus protocol that can achieve k-times consensus in unreliable and multi-access wireless environment. In this protocol, nodes achieve consensus on a proposal by leveraging the received signal strength. Besides, M. Xu et al. propose a jamming-attack-resistant and Sybil-attack-resistant blockchain consensus protocol BLOWN that is designed for wireless networks under adversarial SINR model [12]. This protocol combines the VRF-based Sortition algorithm with Proof-of-Channel consensus algorithm for efficient and secure leader election. In addition, Xu et al. design a fast fault-tolerant wireless blockchain protocol wChain [13], which can quickly aggregate data and reach consensus in multi-hop wireless communication networks.

# Models and Assumptions

In this section, we describe the network model, blockchain basics, threshold signature scheme and adversary assumptions under study.

## 3.1 Network Model

In this study, we consider a wireless network consisting of nodes, which are deployed in Euclidean space. Let be the set of nodes. We assume any node can directly communicate with each other in the network, which means the network under study forms a complete graph. In practice, such a 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 packets, but not both at a time. Time is divided into rounds, and each contains a fixed number of slots. A slot is the time for nodes to transmit a packet. Each node can generate its private-public key pair and main public key by running a secure distributed key generation protocol. The main public key is aggregated by a sufficient number of public keys. Each node can obtain other nodes' public keys and identities by exchanging messages. In this paper, we assume that the number of honest nodes satisfies the requirement of the threshold BLS signature scheme, which ensures the security of the proposed protocol.

## 3.2 Communication Model

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

where is defined as the path loss of the reference distance [32] , and is the Euclidean distance between , as well as is the 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-noise ratio at receiving node is

where is the additive white Gaussian noise power.

The absolute time is split into time slots of fixed time interval, which is calculated as follows:

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

In broadcast protocol, nodes communicate with other nodes within a single hop, which means that any pair of nodes can communicate through a wireless channel. We assume that each the transmission power used by each node should cover the wireless network area.

## 3.3 Blockchain Basics

We assume that each node locally maintains a local 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. Fig. 1 shows the data structure of a block and a transaction, respectively. A block includes block header and block body. The body usually stores transaction data. Block header records blockchain version, block proposer, block height, previous hash, block hash, block full signature, and transactions hash root, 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 Threshold Signature Scheme

Blockchain consensus protocols adopting threshold BLS signature scheme [29] can quickly and steadily achieve consensus in wireless networks with unreliable and unstable communication channels. Any nodes can independently prove that multiple nodes have verified the validation of a block proposal by aggregating several distinct partial signatures to construct a full signature. 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 nodes who collects a sufficient number of distinct partial signature shares of block proposal. In this way, even some messages loss or some participants fail, block consensus can be achieved in blockchain system finally.

In a threshold signature scheme [34], all participants use a share of the main private key as their private keys. Each participant can use its private key to generate partial signature on a message. A full signature of the message can only be constructed by a group of distinct partial signature shares. Participants can verify the full signature by the main public key. Thus, a group of participants can collectively construct a signature without any information about private key. For example, in a -threshold signature scheme, participants hold distinct private key shares of the main private key by executing a distributed key generation algorithm. Each participant can generate partial signature share by a partial signature generation algorithm. And any subset of more than distinct participants can issue a valid full signature through an interactive full signature generation algorithm, while as any subset or fewer participants can't. The full signature can be verified by any participant using the unique main public key.

BLS signature scheme [30] uses bilinear maps and Gap Diffie-Hellman groups where Computational Diffie-Hellman problem is hard, but Decision Diffie-Hellman problem is easy. BLS signature scheme consists of signature generation algorithm and signature verification algorithm. Let be a Gap Diffie-Hellman group with prime order and generator . And is 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. Threshold BLS signature scheme includes key generation algorithm, signature generation algorithm and verification algorithm. All participants work in non-interactive manner. The key generation algorithm adopts distributed key generation scheme [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 full signature. Discrete log-based distributed key generation scheme 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 scheme will randomly select a special value(the main private key) of -degree polynomial to generate main complete public key , which is usually used to verify full 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 method and a full signature recovery method. The partial signature generation method is similar to BLS signature scheme. The full signature recovery method will recover unique full signature, i.e., a unique special value of Lagrange interpolation polynomial of partial signature shares. Besides, the verification algorithm of threshold BLS signature scheme uses the main public key to verify the validation of the full signature, in which the partial signature verification is also same with BLS signature scheme. In this paper, we assume that the number of honest nodes satisfies the requirement of threshold BLS signature scheme, which ensures the security of the proposed protocol.

## 3.5 Adversary

We assume that adversary controls no more than of the total voting power. The malicious behaviors of adversary under consideration are as follows:

* Adversary can issue Sybil attacks, that is, creating pseudo identities. The malicious nodes with pseudonym will generate invalid block or not generate 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.

Table 1 lists major notations used later.



# The Stable Wireless Blockchain Consensus Protocol

In this section, we propose the stable wireless blockchain consensus protocol, named SWIB. We first give an overview architecture of the protocol, and then present a detailed protocol design of SWIB.

## 4.1 Architecture Overview

This subsection introduces the preliminary design of SWIB protocol. Next, we introduce how one round of block generation in SWIB works and how the consensus process transitions from round to round .



**Fig. 2.** An overview of how SWIB works for a round .At the beginning of the round, all consensus nodes are assumed to maintain a same replica of the blockchain. (1) Block proposer election algorithm is executed to randomly determine the block proposer for the current round; (2) verify legality of the 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.

SWIB works in a round by round fashion. Fig. 2 shows how SWIB works in a round *r*. It starts with a secure randomness generation process for block proposer election. It is in general not secure to allow nodes to predict who will be the block proposer in the next round. Thus, a randomness source is needed to ensure that no node can know who will be the next block proposer in advance. 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 a same randomness per round independently. According to the round randomness, a block proposer will be elected at the beginning of each round through the block proposer election algorithm. 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 generationmethod. Once aggregating a threshold of partial signature shares, any consensus node can recover the full signature to finalize the block through a signature recovery method.

We aim to propose a protocol such that hundreds of nodes can achieve consensus in a wireless network with unreliable and unstable channels.

The consensus process contains four important parts are given in the following:

* 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 algorithm, which uses node's private key and the round random number as inputs.
* Block Proposal: 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 can be recovered if a node receives and then aggregates enough partial signature shares. Then, the full signature will be broadcasted to all nodes. The 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 generate a new random value for the next round through distributed randomness generation scheme.



Algorithm 1 shows the process of a complete round in SWIB protocol. Before starting a new round, consensus nodes perform transaction broadcast. Note that the pseudo codes of broadcast operation in blockchain network 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, SWIB can still work smoothly in a wireless blockchain system.



## 4.2 The SWIB Protocol

Before participating consensus, each node will generate key pairs and the main public key through a distributed key generation algorithm. Besides, nodes should also request the identities and public keys of other consensus nodes as well as necessary blockchain history information. SWIB protocol contains four important parts: block proposer election, block generation, block verification and block finalization. In the following subsections, we detailed describe the four parts.

### 4.2.1 Block Proposer Election and Block Generation

We present a verifiable block proposer election algorithm, which is based on distributed randomness generation scheme and threshold BLS signature scheme. Algorithm 1 shows the procedures for block proposer election and block generation (see lines 10-14 therein).

The block proposer election protocol adopts a random seed to guarantee the security of election process. The distributed randomness generation scheme can enable all consensus nodes to jointly generate a round randomness in an 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. 3, 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. 3. Randomness generation of per round.**

The output of the distributed randomness generation scheme 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 scheme 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 scheme 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. 4. The block proposer election for round .**

After execution of the distributed key generation scheme, each node obtains a private-public key pair , to sign and verify messages. The public key is known by all nodes in the system. All nodes will exchange their public keys before entering the consensus process. 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. As illustrated in Fig. 4, all participants have same view of the public keys list.

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 scheme with the 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 : .

The random block proposer election algorithm ensures that the election process is randomized. To determine the block proposer for the current round, the election algorithm divides interval into consecutive intervals:. If , then the node whose public key ranks the will be the block proposer of round . Consensus nodes can independently check whether they are the block proposer of current round. Meanwhile, other nodes can verify the legality of elected block proposer according they maintained nodes list. Besides, the 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 in each round when the full signature for its preceding round is recovered.

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, as well as is the root of transactions. 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 block verification and finalization depend on secure threshold BLS signature scheme. The pseudo code of block verification and finalization is presented in Algorithm 1 (see lines 16-39). SWIB uses three important algorithms of threshold BLS signature scheme: a signature generation algorithm 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 algorithm to check both partial signature and full signature. Blocks and the signatures are sent to consensus nodes via wireless broadcast.

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 election algorithm, which uses current round randomness as input, should be same with the index of block proposer in its nodes list.
* Previous hash : The previous hash has to equal the hash of that block, which was confirmed during the previous round.
* Transactions: All transactions included in the proposed block should not conflict with previous confirmed transactions.

If all the above conditions are satisfied, a node will then generate partial signature of the block hash and broadcast it to other nodes. As shown in Fig. 5, each node can gather a sufficient number of partial signature shares of the block hash to recover the full signature. SWIB uses the full signature as the proof of block finalization. The reconstruction of valid full signature proves that a threshold of nodes sign on the block, 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 refuse to recover the full signature or broadcast valid full signature to other nodes, block finalization can still be reached when there are enough honest nodes. This design enables that block finalization will be steadily achieved in a 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. Therefore, SWIB satisfies strong consistency, which means that it can prevent the occurrence of blockchain fork.



**Fig. 5. Block verification and finalization at a node .**

### 4.2.3 Incentive and Punishment Mechanism

We design an incentive mechanism to encourage consensus nodes to participate the consensus process. Block proposer might be reluctant in block generation due to high power consumption for block packing and its broadcasting. Besides, 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 in 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.

Part of the transaction fees is distributed to block proposers, which make sure they will package transactions as much as possible and generate valid blocks.

The rest fees will be shared among those nodes whose partial signature used to recover the full signature that appended into the valid block in block finalization. Because the uniqueness of threshold BLS signature scheme, any different partial signature shares can recover the same full signature. Since the full signature for a block may be recovered by more than two distinct sets of partial signature shares due to message lost. In this case, the rest fees will be shared by the set of nodes who have the smallest average timestamps of partial signature shares that used to recover the full signature. In other words, we only reward a set of consensus nodes that have signed block hash quickly. 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. The earlier the signature broadcasting, the higher the chance being rewarded.

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 of current round, the node generates invalid block or not generate any block before timeout; 2) node sends invalid signature or garbage messages in block verification and finalization phase. Irrational nodes may harm others without benefiting themselves. A punishment mechanism is necessary to restrict the malicious behaviors of consensus nodes. In SWIB, the active time determines how long a node will work continuously in blockchain system. Hence, reducing the active time of a node is a good punishing measure if the node behaves maliciously.

Reducing the active time of malicious nodes can increase the security and performance of blockchain system. Reducing active time will decrease the stability of nodes, which further reduce the elected probability of nodes. In this way, there is lower chance to finalize an empty block in a round due to malicious nodes’ generation of invalid blocks or doing nothing before timeout. The more valid block finalized, the higher performance of blockchain system. Moreover, irrational nodes will be quickly expelled from the system if they initiate malicious behaviors frequently. As a result, the security of blockchain system will be improved. Therefore, the punishment mechanism can reduce the continuous impact of malicious behaviors and improve the willingness of nodes to behave honest. 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 the consensus process. The joining 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 design reduces the overhead of nodes in blockchain history transmission. For a consensus node, the consumed power for transmitting a small number of blocks will be lower than that for transmitting large number of blocks. The mechanism can effectively prevent a single node from communication interruption due to consuming large power on blockchain history transmission. Meanwhile, synchronization mechanism will reduce the risk that a node transmits error blockchain history to mislead new nodes. Once a node transmits error blockchain history information, the receive node can immediately detect through blocks that received from other nodes. In this case, the new node can request blocks from other trustworthy nodes to ensure the reality of blockchain history information.

Synchronization procedure can also happen when consensus node receives partial signature or full signature before receiving block. This node will request the processing block from its neighbors that have high stability and short distance. If the previous hash of the block not equal to the hash of the latest block on node's local blockchain, the node will request blockchain history from its neighbors. The received blocks are added to the node's local chain if it is absent in the blockchain. When the maintained chain is the latest blockchain, 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 not meets the security threshold. Thus, SWIB can utilize the synchronization mechanism to guarantee the security of SWIB.

# Protocol Analysis

In this section, we will analyze the performance and security of SWIB. We analyze the performance in terms of consensus success probability and consensus latency. Besides, we analyze the security from consensus protocol security, random generation security and attacks resistance.

## Performance Analysis

In this subsection, we analyze consensus success probability and consensus latency, which is an important metric to measure the performance of a blockchain system.

### 5.1.1 Consensus Success Probability Analysis

In SWIB, block proposer is responsible for block generation. After generating a new block, the block proposer sends the new block to other consensus nodes for its validation. Consensus nodes then generate and broadcast partial signatures when they successfully verify the block. When a consensus node receives a sufficient number of partial signature shares, the block finalization has been achieved via recovering a full signature. The full signature is seen as a synchronous block update message, which can be reported by any consensus node obtaining the full signature. Our protocol provides a synchronization mechanism to solve the problem of blockchain inconsistency between different nodes. However, consensus process may be failed if no consensus node receives the minimum required number of partial signature shares for recovering the full signature. This means no consensus node can generate/receive the full signature in consensus round.

In our considered network model, we have the following consensus success probability of SWIB.

**Proposition 1**. For a given round, the consensus success probability of our protocol is lower bounded as

where the function is defined as, for and . Moreover, and are the maximum SNR communication interruption probability and the minimum SNR communication interruption probability in the wireless network, respectively.

**Proof**. In SWIB, aggregation failure occurs when a node fails to receive more than partial signature from other nodes in block verification. Let be the set of consensus nodes in a round, and be the set of nodes that fail to transmit partial signature to the receiver in a given round. We denote the order of set as , i.e., . The aggregation failure probability of any node due to communication interruption is given as

where is the SNR communication interruption probability between and .

The security of SWIB not relies on the correctness of block proposer. When block proposer is faulty, the rest of honest nodes can vote for an empty block and reach consensus eventually. Thus, if all consensus nodes failed to aggregate enough partial signature shares, the consensus process in a round will be interrupted. The consensus success probability is calculated as

where and are the maximum SNR communication interruption probability and the minimum SNR communication interruption probability in the network, respectively. ■

In our protocol, consensus success probability is only related to network size and SNR communication interruption probability between nodes because the consensus security of SWIB does not rely on correct block proposer driving communications.

### 5.1.2 Consensus Latency Analysis

In SWIB, consensus latency is defined as the time interval from a block proposal to its finalization. In order to focus on the impact of wireless communication, we assume that the computing time is negligibly small. Then, the consensus latency in our protocol is obtain as

with a consensus success probability greater than or equal to for some . This equation comes true because a single consensus round of SWIB consist of at most turns of independent message dissemination opportunities. The term is the number of time slots required for the delivery of a message. A block is composed of block header and a block body. The block header mainly stores metadata for identifying blocks and has fixed size. Block body contains multiple transactions, which is assumed to take fixed size. Let be the number of bits that can be transmitted in a slot, and and respectively be block header size and transaction size. The maximum consensus latency of the SWIB without considering channel contention is expressed as

where is the number of slots required to transmit the maximum-sized block, and is the number of slots required to transmit a signature. We assume that the transmission probability of nodes is . In our considered network model, the consensus latency can be computed as follows:

**Proposition 2.** For a given consensus round, let represent the maximum number of time slots for transmitting a block, and be the number of time slots to transmit a signature. The transmission probability of each node is denoted as in the system. The required runtime of consensus process in SWIB is at least time slots, where is some target transmission probability.

**Proof**. In a wireless network, each node determines to transmit a message in a single time slot with probability . Then, the probability that only one node broadcasts a message in a slot is expressed as . Let be a random variable denoting the number of time slots required for a node to successfully transmit a message. We suppose that is the minimum value such that for target successful transmit probability . Then, the mean required number of time slots is for target transmit probability .

For a given round, block proposer needs time slots to transmit a block, and other consensus nodes require time slots to transmit a signature. In block verification, there is at most nodes required to transmit partial signature shares. Moreover, one consensus node is required to transmits full signature for block finalization. Thus, the maximum time slots for a consensus round without considering channel contention is . Therefore, the mean runtime of consensus process in wireless blockchain network is time slots, where is some target transmission probability. ■

We analyze the expected consensus latency in SWIB when considering the consensus success probability and faulty nodes. For some target transmission success probability , each node transmission probability is set to be . We assume that transmitting a valid block requires time slots, and transmitting an empty block requires time slots. Let be the consensus latency of a valid block in round, and be the latency of an empty block in round. The value is the secure threshold of our protocol. and are maximum consensus latency of valid block and empty block, respectively. Let denote the probability that elected block proposer is faulty in a network that has at most faulty nodes among nodes. According to above consensus success probability, we can calculate the expected consensus latency as follows

where is the total number of consensus rounds.

We set the number of required time slots transmitting a signature as , and the number of required time slots transmitting a valid block as . Let and be the number of required time slots transmitting maximum size of block and the number of required time slots transmitting an empty block, respectively. The expected consensus latency is expressed as

where is target transmission success probability, is the transmission probability of each node, is the fraction of faulty nodes, and is the consensus success probability.

## 5.2 Security Analysis

In this subsection, we first analyze the consensus security of SWIB. Then, we analyze the random generation to prove the security of randomness generation scheme. Moreover, we analyze the attacks resistance of SWIB.

### 5.2.1 Consensus 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 scheme when node apply to enter consensus process.

If there is a quorum of nodes to fulfill the requirement of threshold BLS signature scheme, our 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.

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.

### 5.2.2 Random Generation Analysis

Our distributed randomness generation scheme 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 algorithm 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 algorithm 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.

### 5.2.3 Attack Resistance Analysis

In this subsection, we analyze Sybil attacks resistance and Jamming attacks resistance of SWIB.

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. However, the elected probability will not increase due to adversary create multiple pseudonyms in random block proposer election algorithm. The elected probability of nodes is proportional with the stability of nodes. We assume that the stability of an adversary is , and the total stability of network is . If the adversary generates pseudonyms, whose stability respectively are . The sum elected probability of the pseudonyms is . This indicates that generating multiple pseudonyms will not increase the elected probability. As result, our protocol can resistant Sybil attacks in block election process.

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.

SWIB can normaly 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 attacks. Such design can ensure the liveness of blockchain system under adversary issuing jamming attacks, 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.

**Proposition 3**. Let be the maximum number of time slots for each completed consensus round, and be the transmission probability of each node in system. In SWIB, it requires at least time slots lead to against any - bounded adversary, where is some target transmission probability.

**Proof**. According Proposition 1, let be the probability that only one node transmit message in a slot and be the target success transmit probability. Then, the minimum number of time slots for each consensus round is calculated as for target transmit probability . Assuming that in any interval of consecutive slots, adversary can jam no more than slots, where and 0< ≤ 1. For each slot, the jamming probability is . Thus, a message can be transmitted successfully after at least time slots. A completed consensus process for each round requires time slots. Therefore, to ensure at least time slots with successful transmission probability , our protocol runs at least slots for each round under jamming attacks issued by any - bounded adversary. ■

Thus, to ensure the block finalization under jamming attacks, each node maintains an estimate of adversary time window to dynamically adjust round timeout. In this case, SWIB can against - bounded adversary issuing jamming attacks.

# 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.

We set the transmit power at each node for broadcast transmissions to be mW, and the noise power to be mW. The target transmission probability to be , and the target SNR threshold is assumed to be dB. We also fix the path loss exponent as , and bandwidth as MHz. In addition, all nodes are deployed into a network area with size , and the minimum distance between nodes is meter. From Friis equation, we assume and for the simulation we fix meter and meters, from the ISM (industrial scientific and medical) ratio bounds at 2.4GHz. Besides, the transmission probability of each node is set as  .

## 6.1 Weight Coefficient

In SWIB protocol, verifiable block proposer election algorithm randomly determines a block proposer through the stability of nodes. According to the defined formulation of stability , we believe that nodes with large active time 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 active time 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 active time 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 active time and number of latest generate blocks; nodes with lower active time and higher number of latest generate blocks; nodes with higher active time and lower number of latest generate blocks; nodes with higher active time and number of latest generate blocks. The parameters are shown in Fig. 6(a). To investigate the impact of active time ratio weight, we measure the percentage of electing high-quality nodes as block proposer within 100 consensus times.

 

(a) (b)

**Fig. 6. Percentage of high-quality under different coefficients**

As shown in Fig. 6(b), the percentage of high-quality node linearly increases with the increase value of active time ratio weight. Both active time ratio and consensus ratio influence the probability that electing high-quality node as block proposer. The maximum value of the percentage is less than 0.85 while the minimum value is larger than 0.7. When the active time ratio weight equals to 0.7, the percentage high-quality nodes can approximately reach 0.8. In this case, active time ratio is more effect than consensus ratio for stability. Consensus nodes that have high remaining active time would like to be honest than those with low remaining active time due to the incentive and punishment mechanism of SWIB. To ensure relative high probability that high-quality nodes are elected as block proposer, we set the active time ratio weight as 0.7 and consensus ratio weight as 0.3 in the subsequent performance experiments.

## 6.2 Consensus Interruption Comparison

We analyze how the consensus interruption probability is influenced by the communication interruption probability. To compare the consensus interruption probability of SWIB with PBFT, Fig. 7 shows the trend of consensus interruption probability with a varying communication interruption probability.

  (a) Fault-free Theoretical comparison (b) Fault-free Practical Comparison

**Fig. 7. Consensus interruption probability comparison.**

We choose PBFT as the baseline protocol and representative of a classical consensus protocol for Byzantine environment. Assuming that all nodes are honest, Fig. 7(a) shows the theoretical result that the consensus interruption probability in the two protocols remains steady first, then increases to 1.0 and stabilizes at this value. Consensus process in PBFT is interrupted with high probability when communication interruption probability is bigger than 0.4. However, consensus process in our protocol can work smoothly while the communication interruption probability is smaller than 0.5. Since our protocol decouples leader (also denote as block proposer) from communication driven process, our protocol is more stable than that in PBFT when nodes lost some messages. Experiment result shown in Fig. 7(b) further proves the correctness of theoretical analysis.

## 6.3 Performance

In this subsection, we discuss the performance of our protocol. Both block size and network size are factors that influence the performance of blockchain system, which can be measured via consensus latency and transaction throughput.

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

 

1. Average latency Vs. block size (b) Average throughput Vs. block size

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

As shown in Fig. 8(a), consensus latency linearly increases with the increase of block size. When block size is smaller than 1 MB, the average consensus latency is approximate second in network with 500 nodes. Experiment result shows that block transmission is the main bottlenecks of consensus protocol. In Fig. 8(b), the average throughput first increases rapidly, and finally tends to be flat. This indicates 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 large average transaction throughput with small average consensus delay. Therefore, we can handle a large block size to ensure higher transaction throughput within reasonable consensus latency.

We then 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 50 to 600, and replicated the similar experiments with three different block sizes.

 

1. Average latency Vs. network size (b) Average throughput Vs. network size

**Fig. 9. The performance of SWIB vs. network sizes**

As shown in above Fig. 9 (a), consensus latency increases with the number of consensus nodes, and achieving acceptable values even for network size of 600 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. Meanwhile, the increase of network size will lead to high frequency wireless channel competition, which eventually results in the increase of consensus latency. Each node in our protocol dynamically adjusts transmission probability by sensing channels state within the estimate adversary time window. This design ensures that consensus latency will not increase sharply with the increase of network size. Since the block size is fixed and consensus latency increases, the average transaction throughput will reduce. The experiment result of Fig. 9 (b) shows that average transaction throughput can reach approximate 2000 TPS for network size of 600 nodes and block size of 1 MB. Thus, we can handle node transmission probability and block size to guarantee high transaction throughput within reasonable consensus latency for different network scenario.

 

1. Average latency Vs. network size (b) Average throughput Vs. network size

**Fig. 10. The performance of SWIB vs. network sizes**

We compare the performance under fault-free scenario and the performance under different fractions of adversary power in SWIB and PBFT protocols. We run protocols for 10 consecutive rounds with network sizes ranging from 50 to 600 and fixed block size 1 MB. Fig. 10(a) shows that the average consensus latency needed to finalize a 1 MB block with different network size in SWIB and PBFT. Compared with PBFT, our protocol has higher scalability capabilities. Our protocol can achieve consensus within 1 second in wireless network of 600 nodes while the consensus latency will increase quickly in PBFT. Because SWIB allows to securely reach block consensus in a round of voting, which further reducing consensus latency while PBFT requires two rounds communications to reach consensus on a block.

Due to randomness of block proposer election method, the probability of electing a malicious block proposer in a round is related to the fraction of adversary power in network. In PBFT, view change happens when malicious leader proposed an invalid block or not proposed any block. In this case, all honest nodes require two round communications to determine new leader. In SWIB, honest nodes will discard invalid block and sign an empty block instead, then entering the next consensus round. Fig. 10 (b) shows a comparison of the performance when adversary behave malicious behaviors in network of 500 nodes. This experiment assume that all malicious nodes not send any messages in a consensus round. Under this assumption, the performance is affected in both our protocol and PBFT. Results show that our protocol can keep working with adversary power smaller than 50% while PBFT will fail to reach consensus when adversary power more than 33%. This indicates that our protocol has better fault tolerance than PBFT. In summary, experiment results show that SWIB performs high capability than PBFT in wireless block networks.

## 6.4 Attack Resistance

In sybil attacks, an adversary can control some malicious nodes that compete for being block proposer but refuse to generate valid blocks or not generate any blocks. In addition, malicious nodes will not vote for valid block 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.



1. Average latency Vs. fraction of Sybil nodes (b) Average throughput Vs. fraction of Sybil nodes

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

The experiment result in Fig. 11 (a) shows that consensus latency decreases with the increase of Sybil nodes percentage. Since Sybil nodes are not absent from block proposer process, empty block will be generated while Sybil nodes become block proposer. In this case, all honest nodes vote for the empty block after valid block verification and finalization timeout. Compare with propagating a valid block, propagating an empty block requires small number of time slots. Thus, the consensus latency will not significantly increase due to malicious block proposer in a round. Meanwhile, channel contentions between consensus nodes become light when the fraction of Sybil nodes increasing. This is because that Sybil nodes would not send messages during consensus process, or just send some error message with lower transmission probability. As result, the consensus latency declines when the percentage of Sybil nodes increases. With the increase of Sybil nodes percentage, the probability of generating empty blocks is increase. In this case, the average number of confirmed transactions will decrease. As shown in Fig. 11 (b), the average transaction throughput will decrease with the increase of percentage of Sybil nodes in network. Meanwhile, our protocol can achieve more than 2000 TPS in blockchain network with 500 nodes and block size of 1 MB. As a result, our protocol can resist to Sybil attacks, and achieve acceptable performance when adversary issuing Sybil attacks.

We present the performance of SWIB protocol when jammers with constraint of . This indicates that perpetual jammers can jam consecutive slots at any interval length of slots. We analyze the impact of on consensus latency and throughput. We test the performance of SWIB when -bounded jammers adopting different attack strategies. Jammers adopting random strategy can randomly jam slots at any interval of slots. Perpetual jammers can jam consecutive slots in each slots. We test consensus latency and average transaction throughput. We run our protocol for 10 consecutive rounds with value range from 0.1 to 1 and adversary time window of 100 time slots in the two attack strategies.

 

1. Average latency Vs. value (b) Average throughput Vs. value

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

As shown in the Fig. 12 (a), consensus latency decreases with the increase of under random jamming attacks while it keeps steady under perpetual jamming attacks. This indicates that random jamming attack is more power than perpetual jamming attack in SWIB. When adversary issues perpetual jamming attacks, the transmission probability of nodes reduced quickly by the maintained estimate of adversary time window . In this case, channel contentions in network will be reduced, and nodes can transmit a message successfully within small time slots. However, when jammers issue random jam attack, transmission probability of nodes will not reduce frequently due to the estimate of adversary time window  not increase rapidly. The required time of partial signature shares aggregation will decline when increasing the value of . Thus, the consensus latency decreases with the increase of . Moreover, Fig 12 (b) shows that the throughput of our protocol does not fluctuate significantly under perpetual jamming attacks. As block size is fixed and consensus latency is reduced, the average throughput under random jamming attacks increases with the improvement of value . The throughput can exceed 1000 TPS in network size of 500 nodes under -bounded adversary who launches random jamming attack. Therefore, our protocol can securely operate in blockchain system when adversary issuing jamming attack, and achieve acceptable transaction throughput within reasonable consensus latency.

# 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 algorithm to 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. Moreover, our protocol can still work efficiently 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. We should study the energy consumption in consensus process, and design energy efficiency consensus protocol in wireless network. 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] A. Salam. "Internet of things for sustainable community development: introduction and overview[M]." Internet of Things for Sustainable Community Development. Springer, Cham, 2020: 1-31.

[2] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1184-1195, April 2018, doi: 10.1109/JIOT.2018.2812239.

[3] A. K. Gupta and R. Johari, "IOT based Electrical Device Surveillance and Control System," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-5, doi: 10.1109/IoT-SIU.2019.8777342.

[4] N. Dlodlo, O. Gcaba and A. Smith, "Internet of things technologies in smart cities," 2016 IST-Africa Week Conference, 2016, pp. 1-7, doi: 10.1109/ISTAFRICA.2016.7530575.

[5] H. -N. Dai, Z. Zheng and Y. Zhang, "Blockchain for Internet of Things: A Survey," in IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8076-8094, Oct. 2019, doi: 10.1109/JIOT.2019.2920987.

[6] O. Onireti, L. Zhang and M. A. Imran, "On the Viable Area of Wireless Practical Byzantine Fault Tolerance (PBFT) Blockchain Networks," 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013778.

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

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

[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] A. Corso, “Performance analysis of proof-of-elapsed-time consensus in the sawtooth blockchain framework,” 2019.

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

[19] 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.

[20] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of activity: Extending bitcoin’s proof of work via proof of stake [extended abstract]y, in SIGMETRICS Performance Evaluation Review, 42(3): 34-37, 2014.

[21] 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

[22] 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.

[23] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, “Enhancing bitcoin security and performance with strong consistency via collective signing,” in 25th fUSENIXg Security Symposium (fUSENIXg Security 16), 2016, pp. 279–296.

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

[25] 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.

[25] N. Santoro and P. Widmayer. Time is not a healer. In Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science, pages 304–313. Springer-Verlag, 1989.

[26] N. Santoro and P. Widmayer. Distributed function evaluation in presence of transmission faults. Proc. Int. Symp. on Algorithms (SIGAL), pages 358–367, 1990.

[27] 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.

[28] 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.

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

[30] J. Aspnes, F. Fich, and E. Ruppert. "Relationships between broadcast and shared memory in reliable anonymous distributed systems." In 18th International Symposium on Distributed Computing, pages 260–274, 2004.

[31] 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.

[32] 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.

[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.

[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.

[33] B. M. Nejad, S. A. Attia, and J. Raisch, “Max-consensus in a max-plus algebraic setting: The case of fixed communication topologies,” in Proc. 22nd Int. Symp. Inf., Commun. Autom. Technol., 2009, pp. 1–7.

[34] A. Boldyreva. Threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, Proceedings of PKC 2003, volume 2567 of LNCS, pages 31–46. Springer-Verlag, Jan. 2003.